Hashpay
TECHNICAL PUBLICATIONS SUITE

• BUKU PANDUAN RESMI MITRA · A4 RELEASE

# Spesifikasi Integrasi & Referensi API Lengkap Pengembang

Buku panduan teknis rekayasa piranti lunak: autentikasi API, pembuatan order server-to-server, webhook HMAC-SHA256, siklus transaksi, settlement, kode status, dan checklist go-live.

**INFORMASI KONTROL DOKUMEN**

| | |
|---|---|
| NOMOR DOKUMEN | KLASIFIKASI DOKUMEN |
| HP-DOC-2026-INT-V2.4 | Spesifikasi Publik Terverifikasi |
| VERSI & STATUS | TANGGAL PUBLIKASI & BAHASA |
| v2.4 · Production Release | Oktober 2026 · ID |

© 2026 Hashpay Inc. All Rights Reserved · docs.hashpay.id

Zero Floating-Point Financial Standard

STRUKTUR DOKUMEN

# Daftar Isi Lengkap

**MEMULAI**

1.0 Ikhtisar — Bab 1

2.0 Autentikasi — Bab 2

**IMPLEMENTASI**

3.0 Integrasi backend — Bab 3

4.0 Integrasi frontend — Bab 4

5.0 Webhook — Bab 5

**SIKLUS & SETTLEMENT**

6.0 Siklus hidup transaksi — Bab 6

7.0 Settlement & payout — Bab 7

8.0 Error & status code — Bab 8

**PENGUJIAN & PELUNCURAN**

9.0 Sandbox & faucet — Bab 9

10.0 Go live (produksi) — Bab 10

BAB 1.0 · MEMULAI

# Panduan Integrasi

> *Panduan komprehensif bagi developer untuk mengintegrasikan gateway pembayaran kripto Hashpay: pembuatan order dari backend, alur hosted checkout dan direct API, deep linking mobile webview, penanganan webhook, serta peluncuran ke produksi.*

## Pilih jalur integrasi Anda

Hashpay menyediakan dua jalur integrasi utama yang disesuaikan dengan arsitektur sistem dan kebutuhan antarmuka bisnis Anda:

| JALUR INTEGRASI | COCOK UNTUK | TINGKAT KOMPLEKSITAS |
|---|---|---|
| Hosted Payment Page (Webview) | Website e-commerce, aplikasi mobile (Flutter, React Native, iOS, Android), dan terminal POS. Hashpay menangani pemilihan token, koneksi dompet Web3, kunci kurs, dan QR dinamis secara mandiri. | 1 panggilan API backend + redirect atau modal webview |
| Direct API (UI Kustom) | Merchant yang membutuhkan tampilan checkout 100% white-label di dalam aplikasi mereka sendiri. Anda mendesain UI secara bebas, sementara Hashpay menyediakan perhitungan kurs dan verifikasi on-chain. | 4 panggilan API backend + implementasi UI kustom |

## Matriks lingkungan & Base URL

Hashpay menerapkan isolasi ketat antara lingkungan Sandbox (Testnet) dan Produksi (Mainnet). Pastikan server Anda menggunakan base URL dan kredensial yang tepat:

| PARAMETER | SANDBOX (TESTNET) | PRODUKSI (MAINNET) |
|---|---|---|
| Base API URL | `https://api.hashpay.id/v2/testnet` | `https://api.hashpay.id/v2/mainnet` |
| URL Hosted Checkout | `https://webview.hashpay.id/sandbox` | `https://webview.hashpay.id` |
| Jaringan Blockchain | Ethereum (Sepolia), BNB Chain (Testnet), Tron (Shasta), Solana (Devnet) | Ethereum, BNB Chain, Tron, Solana |
| Aset Settlement | USDT & USDC (Faucet Testnet) | USDT & USDC (Aset Asli Mainnet) |
| Kredensial API | Secret Key Testnet ( `sec_test_...` ) | Secret Key Mainnet ( `sec_live_...` ) |

## Tahapan integrasi end-to-end

1. **Pembuatan Order & Kuotasi Kurs** — Server Anda memverifikasi batas nominal transaksi, membuat order via `POST /merchant/fiat/order`, dan menerima URL sesi hosted checkout (`data.url`) beserta scoped key.
2. **Presentasi Antarmuka Checkout** — Sajikan layar pembayaran ke pelanggan: buka halaman hosted via redirect browser, iframe, atau webview mobile; atau tampilkan QR kustom via Direct API.
3. **Pemantauan Blockchain & Kunci Kurs** — Hashpay mengunci nilai tukar kripto real-time selama 15 menit. Node watcher mendeteksi transfer masuk di blockchain dalam 2 hingga 5 detik.
4. **Notifikasi Webhook & Pemenuhan Pesanan** — Setelah mencapai konfirmasi blok yang diperlukan, Hashpay mengirimkan notifikasi HTTP POST bertanda tangan kriptografis (`transaction.success`) ke server Anda. Server Anda memvalidasi signature dan memproses barang pelanggan.
5. **Rekonsiliasi & Settlement** — Dana hasil penjualan disapu secara otomatis ke dompet settlement bisnis Anda atau dicairkan langsung ke rekening bank lokal di Indonesia.

> ⭐ **ATURAN EMAS INTEGRASI PEMBAYARAN**
>
> Selalu jadikan notifikasi webhook server-ke-server sebagai satu-satunya sumber kebenaran status pembayaran. Jangan pernah menyerahkan barang pesanan hanya berdasarkan kembalinya browser pelanggan, karena sinyal sisi klien dapat dipalsukan atau terputus koneksi.

BAB 2.0 · MEMULAI

# Autentikasi

> *Hashpay punya tiga gaya autentikasi: header api-key untuk endpoint transaksi, merchant JWT untuk dashboard, dan order-scoped key untuk hosted checkout.*

## Header api-key (endpoint transaksi)

Panggilan server-ke-server ke endpoint order, rate, QR, dan token memakai **secret key** di header `api-key`.

Header — CODE

```
curl -X POST "https://api.hashpay.id/v2/mainnet/merchant/fiat/order" \
  -H "api-key: $HASHPAY_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{ "merchant_id": "10001", "order_no": "INV-2026-0001", "amount": "150000" }'
```

## Merchant JWT (dashboard & manajemen)

Endpoint profil, dashboard, disbursement, payment-link, dan webhook-config memakai `Authorization: Bearer <access_token>` dari `POST /v2/auth/login`.

| ENDPOINT | AUTH |
|---|---|
| `/orderLimits`, `/merchant/fiat/order`, `/checkOrderRate`, `/pgqr/fiat/create`, `/listToken`, `/merchant/settlement` | Header `api-key` |
| `/merchant`, `/merchant/dashboard`, `/merchant/transaction` | Merchant JWT |
| `/merchant/disbursement`, `/merchant/payment-link` | Merchant JWT |
| `/rate`, `/faucet`, `/order/payment-link` | Public |

## Order-scoped key (hosted checkout)

Halaman pembayaran hosted memakai key berumur pendek per order (`pl_…`, TTL 30 menit) yang tertanam di `url` checkout. Jangan pernah menaruh secret key asli di client.

> 💡 **CATATAN PENTING**
>
> Endpoint seperti `getOrder` dan `checkOrderRate` menerima api-key asli atau order-scoped key yang cocok.

BAB 3.0 · IMPLEMENTASI

# Integrasi Back-End

> *Panduan lengkap integrasi sisi server: validasi limit transaksi, penerbitan tagihan order, penguncian kurs kripto, pembuatan QR dinamis, dan rekonsiliasi status transaksi.*

## Autentikasi & pemetaan kredensial

Setiap pemanggilan server-ke-server ke endpoint transaksi Hashpay wajib menyertakan Secret Key Anda di dalam header `api-key`. Jangan pernah mengekspos Secret Key di kode sisi klien, bundel aplikasi mobile, atau repositori publik:

| PARAMETER | HEADER HTTP / FIELD | KETERANGAN & SUMBER |
|---|---|---|
| Secret Key | `api-key: <SECRET_KEY_ANDA>` | Kunci rahasia merchant Anda. Ditemukan di Dashboard → Pengaturan → Kunci API. |
| Merchant ID | `merchant_id: "10001"` | Nomor identitas merchant publik Anda. Ditemukan di Dashboard → Pengaturan → Kunci API. |
| Nomor Order | `order_no` / `transactionID` | ID transaksi unik dari sistem Anda (contoh: `INV-2026-0001`). Gunakan nilai yang konsisten di setiap langkah. |
| Nominal Tagihan | `amount: "150000"` | Jumlah tagihan dalam Rupiah (IDR) dalam format teks integer (tanpa titik atau desimal). |
| Kontrak Token | `tokenAddress` | Alamat smart contract stablecoin yang dipilih, diambil dari `GET /listToken`. |
| Jaringan Blockchain | `network` | Pilihan jaringan blockchain: `ETHEREUM`, `BNB Chain`, `TRON`, atau `SOLANA`. |

## Langkah 0 — Periksa batas limit transaksi (Pre-flight)

`GET` `/orderLimits?merchant_id={merchant_id}`

Sebelum menampilkan opsi pembayaran atau menerbitkan order, tanyakan batas limit nominal toko Anda. Hashpay menghitung batas minimum transaksi secara dinamis agar nilai belanja cukup untuk menutup biaya MDR dan biaya gas jaringan blockchain:

Request Cek Batas Order — BASH

```
curl -X GET "https://api.hashpay.id/v2/mainnet/orderLimits?merchant_id=10001" \
  -H "api-key: sec_live_9f83b27a8e..."
```

Respons Cek Batas Order JSON


```
{
  "status_code": 200,
  "message": "success",
  "data": {
    "merchant_id": "10001",
    "currency": "IDR",
    "min_order_amount": "10000.00",
    "max_order_amount": "500000000.00",
    "mdr_fee_percent": "0.50"
  }
}
```


> 💡 **CATATAN PENTING**
>
> Selalu validasi bahwa total belanja pembeli tidak kurang dari `min_order_amount`. Pengiriman order di bawah batas minimal akan langsung ditolak sistem dengan error `400 Bad Request`.

## Langkah 1 — Buat order baru (Untuk kedua jalur)

`POST` `/merchant/fiat/order`

Titik masuk utama untuk **Jalur A (Hosted Page)** maupun **Jalur B (Direct API)**. Perintah ini mendaftarkan order baru di Hashpay dan menerbitkan sesi checkout serta kredensial per-order:

Pembuatan Order (cURL) BASH

```
curl -X POST "https://api.hashpay.id/v2/mainnet/merchant/fiat/order" \
  -H "api-key: sec_live_9f83b27a8e..." \
  -H "Content-Type: application/json" \
  -d '{
    "merchant_id": "10001",
    "order_no": "INV-2026-0001",
    "amount": "150000",
    "name": "Budi Santoso",
    "email": "budi@example.com",
    "phone": "081234567890",
    "products": [
      { "name": "Biji Kopi Spesialti 250g", "price": 150000, "quantity": 1 }
    ]
  }'
```

**Pembuatan Order (Node.js / TypeScript)** TYPESCRIPT

```typescript
import axios from "axios";

interface CreateOrderResponse {
  status_code: number;
  message: string;
  data: {
    request: {
      order_no: string;
      merchant_id: string;
      amount: string;
      currency: string;
      status: string;
    };
    url: string;
    key: string;
    expired_at: number;
  };
}

async function createHashpayOrder(orderNo: string, amountIdr: number, customerEmail: string) {
  const response = await axios.post<CreateOrderResponse>(
    "https://api.hashpay.id/v2/mainnet/merchant/fiat/order",
    {
      merchant_id: process.env.HASHPAY_MERCHANT_ID,
      order_no: orderNo,
      amount: amountIdr.toString(),
      email: customerEmail,
    },
    {
      headers: {
        "api-key": process.env.HASHPAY_SECRET_KEY,
        "Content-Type": "application/json",
      },
    }
  );

  return response.data.data;
}
```

**Pembuatan Order (PHP / Laravel)** PHP

```php
use Illuminate\Support\Facades\Http;

$response = Http::withHeaders([
    'api-key' => config('services.hashpay.secret_key'),
    'Content-Type' => 'application/json',
])->post('https://api.hashpay.id/v2/mainnet/merchant/fiat/order', [
    'merchant_id' => config('services.hashpay.merchant_id'),
    'order_no'    => 'INV-2026-0001',
    'amount'      => '150000',
    'name'        => 'Budi Santoso',
    'email'       => 'budi@example.com',
]);

if ($response->successful()) {
    $data = $response->json('data');
    // Untuk Jalur A: redirect pembeli langsung ke $data['url']
    return redirect($data['url']);
}
```

**Payload Respons Pembuatan Order** JSON


```json
{
  "status_code": 201,
  "message": "success",
  "data": {
    "request": {
      "order_no": "INV-2026-0001",
      "merchant_id": "10001",
      "amount": "150000",
      "currency": "IDR",
      "status": "Created"
    },
    "url": "https://webview.hashpay.id/sandbox?key=pl_eyJhbGciOi...",
    "key": "pl_eyJhbGciOi...",
    "expired_at": 1821387856
  }
}
```


⭐ **DEVELOPER JALUR A: SELESAI DI SINI!**

Jika Anda memilih **Jalur A (Hosted Checkout)**, Anda tidak perlu memanggil langkah 2 hingga 4. Cukup buka `data.url` di browser pembeli atau sematkan di webview aplikasi Anda. Hashpay menangani pemilihan token, koneksi wallet, penguncian kurs, dan QR secara otomatis. Lanjutkan langsung ke **Integrasi Front-End**.

## Jalur B — Direct API (Membangun UI checkout sendiri)

Bila aplikasi Anda merender komponen UI pembayaran sendiri di dalam aplikasi native, jalankan langkah 2, 3, dan 4 pada server Anda:

## Langkah 2 — Ambil daftar token & wallet yang aktif

`GET` `/listToken?merchant_id={merchant_id}`

Mengembalikan seluruh pilihan stablecoin (USDT, USDC), jaringan blockchain (Ethereum, BNB Chain, Tron, Solana), alamat kontrak, desimal, dan opsi wallet yang aktif untuk toko Anda:

**Request Ambil Daftar Token** BASH

```bash
curl -X GET "https://api.hashpay.id/v2/mainnet/listToken?merchant_id=10001" \
  -H "api-key: sec_live_9f83b27a8e..."
```

**Respons Daftar Token (Ringkasan)** JSON


```json
{
  "status_code": 200,
  "message": "success",
  "data": [
    {
      "walletType": "METAMASK",
      "walletIcon": "https://assets.hashpay.id/wallets/metamask.png",
      "tokenDetails": [
        {
          "name": "USDT",
          "network": "BNB Chain",
          "tokenAddress": "0x55d398326f99059fF775485246999027B3197955",
          "icon": "https://assets.hashpay.id/tokens/usdt.png",
          "decimals": 18,
          "chainId": 56
        },
        {
          "name": "USDT",
          "network": "TRON",
          "tokenAddress": "TR7NHqjeKQxGTCi8q8ZY4pL8otSzgjLj6t",
          "icon": "https://assets.hashpay.id/tokens/usdt.png",
          "decimals": 6,
          "chainId": 728126428
        }
      ]
    }
  ]
}
```


## Langkah 3 — Kunci nilai kurs & hitung rincian fee

`POST` `/merchant/fiat/checkOrderRate`

Mengunci nilai tukar kripto real-time selama 15 menit dan menghitung nominal persis aset kripto yang harus dibayar. Kirimkan `tokenAddress` dan `network` yang dipilih dari Langkah 2, serta masukkan `order_no` sebagai `transactionID`:

**Request Kunci Kurs** BASH

```bash
curl -X POST "https://api.hashpay.id/v2/mainnet/merchant/fiat/checkOrderRate" \
  -H "api-key: sec_live_9f83b27a8e..." \
  -H "Content-Type: application/json" \
  -d '{
    "merchant_id": "10001",
    "transactionID": "INV-2026-0001",
    "tokenAddress": "0x55d398326f99059fF775485246999027B3197955",
    "network": "BNB Chain"
  }'
```

**Respons Kunci Kurs** JSON


```json
{
  "status_code": 200,
  "message": "success",
  "data": {
    "order_no": "INV-2026-0001",
    "merchant_id": "10001",
    "status": "Created",
    "currency": "IDR",
    "amount": "150000",
    "amount_crypto": "9.200000",
    "crypto_rate": "16304.34",
    "is_direct_deposit": false,
    "merchant_fees": [
      {
        "name": "MDR Platform Fee",
        "fee_type": "PERCENT",
        "fee_value": "0.50",
        "fee_amount": "750.00",
        "fee_amount_crypto": "0.046000",
        "is_included": true
      }
    ],
    "total_fee_amount_merchant": "750.00",
    "total_fee_amount_crypto_merchant": "0.046000",
    "merchant_received_amount": "149250.00",
    "merchant_received_amount_crypto": "9.154000",
    "fees": [],
    "total_fee_amount": "0.00",
    "total_fee_amount_crypto": "0.000000",
    "total_amount": "150000.00",
    "total_amount_crypto": "9.200000"
  }
}
```


### 💡 MEMAHAMI DIRECT DEPOSIT VS ALAMAT HD

Bila `is_direct_deposit: true`, pembayaran menggunakan deposit pool bersama dengan akhiran 3 digit kode unik (misal 9.200470 USDT) untuk memetakan transfer pelanggan ke order Anda. Bila `is_direct_deposit: false`, sistem membuat alamat HD baru yang eksklusif hanya untuk order ini.

## Langkah 4 — Terbitkan QR on-chain & alamat deposit

`POST` `/pgqr/fiat/create`

Menghasilkan data QR dinamis yang siap dipindai dompet kripto. Mengembalikan URI standar EIP-681 (`qrValue`) dan alamat tujuan blockchain yang valid:

**Request Pembuatan QR Dinamis** BASH

```
curl -X POST "https://api.hashpay.id/v2/mainnet/pgqr/fiat/create" \
  -H "api-key: sec_live_9f83b27a8e..." \
  -H "Content-Type: application/json" \
  -d '{
    "merchant_id": "10001",
    "transactionID": "INV-2026-0001",
    "tokenAddress": "0x55d398326f99059fF775485246999027B3197955",
    "network": "BNB Chain",
    "walletType": "METAMASK"
  }'
```

**Respons Pembuatan QR Dinamis** JSON


```
{
  "status_code": 200,
  "message": "success",
  "data": {
    "qrValue": "ethereum:0x55d398326f99059fF775485246999027B3197955@56/transfer?
address=0x2AC2223419f32a6Aab57D9d4cb3210C97Bc62d51&uint256=9200000000000000000",
    "request": {
      "order_no": "INV-2026-0001",
      "merchantID": "10001",
      "tokenName": "USDT",
      "network": "BNB Chain",
      "walletType": "METAMASK",
      "destinationAddress": "0x2AC2223419f32a6Aab57D9d4cb3210C97Bc62d51",
      "amount": "9.200000",
      "totalAmount": "9.200000",
      "totalAmountFiat": "150000.00",
      "uniqueCode": "0",
      "expiredAt": 1821388756,
      "is_direct_deposit": false
    }
  }
}
```


> ⚠ **KEWAJIBAN NOMINAL PERSIS**
>
> Pelanggan harus mengirimkan nominal `request.totalAmount` secara persis ke `request.destinationAddress`. Jika pelanggan mengirim nominal kurang, sistem akan menandai transaksi sebagai Underpaid dan webhook sukses tidak akan terpicu otomatis.

## Langkah 5 — Polling status & rekonsiliasi (Fallback)

`POST` **/merchant/fiat/getOrder**

Meskipun webhook adalah jalur pengiriman utama untuk notifikasi pembayaran lunas, gunakan `getOrder` sebagai cadangan rekonsiliasi berkala (misal bila server webhook Anda sempat down atau ketika memverifikasi komplain pelanggan secara real-time):

**Request Cek Status Order** BASH

```bash
curl -X POST "https://api.hashpay.id/v2/mainnet/merchant/fiat/getOrder" \
  -H "api-key: sec_live_9f83b27a8e..." \
  -H "Content-Type: application/json" \
  -d '{
    "merchant_id": "10001",
    "order_no": "INV-2026-0001"
  }'
```

**Respons Cek Status Order** JSON

```json
{
  "status_code": 200,
  "message": "success",
  "data": {
    "order_no": "INV-2026-0001",
    "merchant_id": "10001",
    "merchant_name": "Toko Sejahtera Bersama",
    "currency": "IDR",
    "amount": "150000",
    "status": "Success",
    "paid_amount": "9.200000",
    "total_amount": "9.200000",
    "total_amount_fiat": "150000.00",
    "tx_hash": "0x3a7e28b14c995f088127394c8b66da982341209b0a1d48293c834017834bcdef",
    "link_explorer":
"https://bscscan.com/tx/0x3a7e28b14c995f088127394c8b66da982341209b0a1d48293c834017834bcdef",
    "expired_at": 1821388756,
    "createdAt": "2026-10-23T10: 15:00Z",
    "updatedAt": "2026-10-23T10: 16:42Z"
  }
}
```

BAB 4.0 · IMPLEMENTASI

# Integrasi Front-End

> *Menyajikan pengalaman checkout kepada pembeli: redirect browser, modal iframe responsif, JavaScript SDK gaya Midtrans (snap.js), serta penanganan deep linking webview pada aplikasi mobile.*

## Ikhtisar opsi antarmuka pembayaran

Saat order dibuat, server Anda menerima parameter `data.url` yang memuat token sesi terenkripsi. Tentukan opsi antarmuka yang paling cocok untuk aplikasi Anda:

| OPSI | POLA ANTARMUKA | REKOMENDASI PENGGUNAAN |
|---|---|---|
| Opsi A | Full Redirect Browser | Website toko online e-commerce multi-halaman dan keranjang belanja standar. |
| Opsi B | Responsive Iframe Embed | Aplikasi web modern (SPA) yang membutuhkan modal pembayaran di halaman yang sama. |
| Opsi C | JavaScript SDK (`snap.js`) | Website yang membutuhkan modal popup dengan callback lifecycle JavaScript. |
| Opsi D | Mobile In-App WebView | Aplikasi native iOS, Android, Flutter, dan React Native. |

## Opsi A — Full redirect browser (Rekomendasi untuk web)

Arahkan browser pembeli langsung ke `data.url`. Anda dapat menambahkan parameter `&redirectUrl=` untuk menentukan URL tujuan pada website toko Anda setelah pembayaran selesai:

**Redirect Browser di Sisi Klien** — JAVASCRIPT

```javascript
// Ambil data.url dari endpoint backend pembuatan order Anda
const checkoutUrl = data.url + "&redirectUrl=" + encodeURIComponent("https://tokosaya.com/orders/INV-2026-0001/selesai");

// Arahkan pelanggan ke halaman checkout Hashpay
window.location.assign(checkoutUrl);
```

## Opsi B — Responsive iframe embed

Sematkan halaman checkout langsung di dalam layout website atau dialog modal kustom Anda. Pastikan menyertakan atribut `allow="clipboard-write"` agar fitur salin alamat deposit berfungsi lancar:

Markup Modal Iframe HTML

```
<iframe
  src="https://webview.hashpay.id/?key=pl_eyJhbGciOi...&redirectUrl=https%3A%2F%2Ftokosaya.com%2Fselesai"
  style="width: 100%; height: 740px; border: 0; border-radius: 16px; overflow: hidden;"
  allow="clipboard-write"
></iframe>
```

## Opsi C — JavaScript SDK (snap.js)

Gunakan pustaka klien ringan kami bergaya Midtrans Snap. Script mengekspos fungsi `window.pay` untuk produksi dan `window.paytest` untuk sandbox:

Integrasi Snap.js HTML

```
<!-- Pasang script Hashpay Snap di tag <head> halaman Anda -->
<script src="https://webview.hashpay.id/snap/snap.js"></script>

<script>
  function openHashpayModal(orderKey) {
    // Gunakan window.paytest() untuk Sandbox; window.pay() untuk Produksi
    window.pay(
      { key: orderKey },
      {
        onSuccess: function (result) {
          console.log("Pembayaran terdeteksi sukses:", result);
          window.location.href = "/pembayaran-berhasil?order=" + result.order_no;
        },
        onPending: function (result) {
          console.log("Transaksi sedang menunggu konfirmasi on-chain:", result);
        },
        onError: function (error) {
          console.error("Pembayaran gagal atau dibatalkan:", error);
          alert("Pembayaran gagal diselesaikan. Silakan coba kembali.");
        },
        onClose: function () {
          console.log("Pelanggan menutup modal checkout.");
        }
      }
    );
  }
</script>
```

## Opsi D — Webview aplikasi mobile & penanganan deep link

Saat menyematkan webview checkout di dalam aplikasi native mobile (Flutter, React Native, Android Kotlin, atau iOS Swift), tombol koneksi wallet seperti MetaMask atau Trust Wallet akan memicu skema URL khusus seperti `ethereum:`, `solana:`, `wc:`, `metamask:`, dan `trust:`. Webview default mobile akan menganggap skema tersebut sebagai error. Anda WAJIB mencegat skema tersebut dan membukanya di aplikasi dompet pengguna:

**Implementasi Flutter WebView (webview_flutter)** DART

```dart
import 'package:flutter/material.dart';
import 'package:webview_flutter/webview_flutter.dart';
import 'package:url_launcher/url_launcher.dart';

class HashpayCheckoutScreen extends StatefulWidget {
  final String checkoutUrl;
  const HashpayCheckoutScreen({Key? key, required this.checkoutUrl}) : super(key: key);

  @override
  State<HashpayCheckoutScreen> createState() => _HashpayCheckoutScreenState();
}

class _HashpayCheckoutScreenState extends State<HashpayCheckoutScreen> {
  late final WebViewController _controller;

  @override
  void initState() {
    super.initState();
    _controller = WebViewController()
      ..setJavaScriptMode(JavaScriptMode.unrestricted)
      ..setNavigationDelegate(
        NavigationDelegate(
          onNavigationRequest: (NavigationRequest request) async {
            final uri = Uri.parse(request.url);
            // Cegat deep link non-HTTP untuk wallet Web3 (MetaMask, Trust, dll.)
            if (!request.url.startsWith('http://') && !request.url.startsWith('https://')) {
              if (await canLaunchUrl(uri)) {
                await launchUrl(uri, mode: LaunchMode.externalApplication);
                return NavigationDecision.prevent;
              }
            }
            return NavigationDecision.navigate;
          },
        ),
      )
      ..loadRequest(Uri.parse(widget.checkoutUrl));
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Pembayaran Kripto')),
      body: WebViewWidget(controller: _controller),
    );
  }
}
```

**Implementasi React Native WebView (react-native-webview)** TYPESCRIPT

```typescript
import React from 'react';
import { Linking } from 'react-native';
import { WebView } from 'react-native-webview';

export function HashpayCheckoutModal({ checkoutUrl, onPaymentClosed }) {
  return (
    <WebView
      source={{ uri: checkoutUrl }}
      javaScriptEnabled={true}
      domStorageEnabled={true}
      onShouldStartLoadWithRequest={(request) => {
        // Cegat skema deep link dompet Web3
        if (!request.url.startsWith('http://') && !request.url.startsWith('https://')) {
          Linking.openURL(request.url).catch((err) =>
            console.warn('Gagal membuka aplikasi dompet kripto:', err)
          );
          return false; // Cegah crash webview internal
        }
        return true;
      }}
    />
  );
}
```

**Implementasi Android Kotlin Native (WebViewClient)** KOTLIN

```kotlin
webView.webViewClient = object : WebViewClient() {
    override fun shouldOverrideUrlLoading(view: WebView?, request: WebResourceRequest?): Boolean {
        val url = request?.url?.toString() ?: return false
        if (!url.startsWith("http://") && !url.startsWith("https://")) {
            try {
                val intent = Intent(Intent.ACTION_VIEW, Uri.parse(url))
                view?.context?.startActivity(intent)
                return true
            } catch (e: ActivityNotFoundException) {
                Toast.makeText(view?.context, "Aplikasi dompet kripto tidak ditemukan",
Toast.LENGTH_SHORT).show()
                return true
            }
        }
        return false
    }
}
```

🔒 **SELALU KONFIRMASI MELALUI WEBHOOK SISI SERVER**

Callback sisi klien (seperti `onSuccess` atau redirect URL) murni berfungsi untuk mengarahkan navigasi antarmuka pembeli. Jangan pernah mengirimkan barang fisik atau mengaktifkan paket langganan sebelum server Anda memvalidasi tanda tangan HMAC-SHA256 pada event webhook `transaction.success`.

BAB 5.0 · IMPLEMENTASI

# Webhook

> *Hashpay mengirimkan notifikasi HTTP POST bertanda tangan ke URL webhook Anda setiap kali transaksi mencapai status terminal. Webhook adalah satu-satunya sumber kebenaran utama untuk rekonsiliasi pembayaran.*

## Tipe event webhook

Hashpay mengirimkan payload event real-time yang mewakili status penting dalam siklus hidup pembayaran pelanggan:

| NAMA EVENT | KONDISI TERPICU | TINDAKAN YANG HARUS DILAKUKAN MERCHANT |
|---|---|---|
| `transaction.success` | Transfer pelanggan terkonfirmasi penuh di blockchain dan saldo toko telah dikreditkan. | Tandai pesanan LUNAS; kirimkan produk digital atau proses pengiriman barang fisik. |
| `transaction.expired` | Masa kunci kurs 15 menit telah habis tanpa adanya penerimaan dana. | Tandai pesanan EXPIRED; minta pelanggan membuat kuotasi pembayaran baru. |
| `transaction.invalid` | Transfer on-chain terdeteksi namun tidak valid (misal: kurang bayar, token salah, atau rantai keliru). | Catat hash transaksi dan serahkan ke tim CS toko untuk penanganan manual. |
| `webhook.test` | Simulasi webhook yang dipicu secara manual dari Dashboard (Pengaturan → Kunci API → Kirim Event Uji). | Pastikan logika verifikasi signature dan handler HTTP 200 server Anda bekerja sempurna. |

## Header HTTP yang dikirimkan pada setiap webhook

Setiap pengiriman webhook memuat header kriptografis standar untuk verifikasi keaslian pengirim:

| HEADER | KETERANGAN & FORMAT |
| --- | --- |
| `X-Hashpay-Event` | Nama event yang dikirim (contoh: `transaction.success`). |
| `X-Hashpay-Timestamp` | Waktu Unix epoch dalam detik saat notifikasi dibuat. |
| `X-Hashpay-Signature` | Tanda tangan HMAC-SHA256 dalam format `sha256=<hex_digest>`. |
| `X-Hashpay-Signature-Version` | Versi algoritma signature (`v2`). |
| `X-Hashpay-Idempotency-Key` | Kunci idempoten unik (berdasarkan tx hash atau ID pengiriman) untuk mencegah pemrosesan ganda. |

## Contoh payload JSON event

Berikut adalah contoh struktur payload JSON asli untuk masing-masing tipe event:

**1. Payload transaction.success** JSON

```
{
  "event": "transaction.success",
  "data": {
    "order_no": "INV-2026-0001",
    "merchant_id": "10001",
    "status": "Success",
    "token": "USDT",
    "network": "BNB Chain",
    "paid_amount": "9.200000",
    "final_amount": "9.154000",
    "tx_hash": "0x3a7e28b14c995f088127394c8b66da982341209b0a1d48293c834017834bcdef",
    "link_explorer":
"https://bscscan.com/tx/0x3a7e28b14c995f088127394c8b66da982341209b0a1d48293c834017834bcdef",
    "manual_review": false
  }
}
```

**2. Payload transaction.expired** JSON

```
{
  "event": "transaction.expired",
  "data": {
    "order_no": "INV-2026-0002",
    "merchant_id": "10001",
    "status": "Expired",
    "expired_at": 1821388756
  }
}
```

**3. Payload transaction.invalid** JSON

```
{
  "event": "transaction.invalid",
  "data": {
    "order_no": "INV-2026-0003",
    "merchant_id": "10001",
    "status": "Invalid",
    "token": "USDT",
    "network": "BNB Chain",
    "paid_amount": "5.000000",
    "tx_hash": "0x89ab12cd34ef567890123456789abcdef0123456789abcdef0123456789abcde",
    "link_explorer": "https://bscscan.com/tx/0x89ab...",
    "invalid_message": "Underpayment detected: expected 9.200000 USDT but received 5.000000 USDT",
    "manual_review": true
  }
}
```

**4. Payload webhook.test** JSON

```
{
  "event": "webhook.test",
  "test": true,
  "message": "This is a test event from the Hashpay dashboard.",
  "data": {
    "order_no": "INV-TEST-9999",
    "merchant_id": "10001",
    "status": "Success"
  }
}
```

## Verifikasi tanda tangan webhook (HMAC-SHA256)

Tanda tangan dihitung dari gabungan teks `<timestamp>.<raw_body>` menggunakan **Webhook Signing Secret** toko Anda sebagai kunci HMAC. Lakukan verifikasi di server sebelum membaca isi payload:

**Verifikasi Tanda Tangan di Node.js / Express** TYPESCRIPT

```typescript
import crypto from "node:crypto";
import { Request, Response } from "express";

export function verifyHashpayWebhook(req: Request, res: Response) {
  const signature = req.headers["x-hashpay-signature"] as string;
  const timestamp = req.headers["x-hashpay-timestamp"] as string;
  const secret = process.env.HASHPAY_SIGNING_SECRET!;

  // 1. Proteksi replay attack: tolak timestamp yang berusia lebih dari 5 menit (300 detik)
  const age = Math.abs(Math.floor(Date.now() / 1000) - Number(timestamp));
  if (!Number.isFinite(age) || age > 300) {
    return res.status(400).send("Timestamp kedaluwarsa atau invalid");
  }

  // 2. Hitung HMAC-SHA256 dari timestamp + "." + raw body buffer
  const expected = "sha256=" + crypto
    .createHmac("sha256", secret)
    .update(Buffer.concat([Buffer.from(timestamp + "."), req.body]))
    .digest("hex");

  // 3. Komparasi konstan waktu (timing-safe) untuk mencegah serangan timing attack
  const sigBuffer = Buffer.from(signature ?? "");
  const expBuffer = Buffer.from(expected);
  const isValid = sigBuffer.length === expBuffer.length && crypto.timingSafeEqual(sigBuffer, expBuffer);

  if (!isValid) {
    return res.status(401).send("Tanda tangan webhook invalid");
  }

  // 4. Proses data pesanan
  const payload = JSON.parse(req.body.toString());
  if (payload.event === "transaction.success") {
    console.log("Pesanan berhasil dibayar:", payload.data.order_no);
    // Update status pesanan di database toko Anda...
  }

  // 5. Segera balas dengan status 200 OK
  return res.status(200).json({ received: true });
}
```

**Verifikasi Tanda Tangan di PHP / Laravel** PHP

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HashpayWebhookController extends Controller
{
    public function handle(Request $request)
    {
        $signature = $request->header('X-Hashpay-Signature');
        $timestamp = $request->header('X-Hashpay-Timestamp');
        $secret    = config('services.hashpay.signing_secret');

        // 1. Tolak timestamp yang lebih dari 300 detik
        if (abs(time() - (int)$timestamp) > 300) {
            return response()->json(['error' => 'Timestamp kedaluwarsa'], 400);
        }

        // 2. Hitung HMAC dari raw request body
        $rawBody  = $request->getContent();
        $expected = 'sha256=' . hash_hmac('sha256', $timestamp . '.' . $rawBody, $secret);

        // 3. Bandingkan dengan hash_equals (timing-safe)
        if (!hash_equals($expected, (string)$signature)) {
            return response()->json(['error' => 'Signature tidak cocok'], 401);
        }

        // 4. Proses order
        $payload = $request->json()->all();
        if ($payload['event'] === 'transaction.success') {
            $orderNo = $payload['data']['order_no'];
            // Update order di database...
        }

        return response()->json(['status' => 'success'], 200);
    }
}
```

Verifikasi Tanda Tangan di Python (FastAPI / Flask) PYTHON

```python
import hmac
import hashlib
import time

def verify_hashpay_webhook(raw_body: bytes, timestamp_str: str, signature: str, secret: str) -> bool:
    try:
        timestamp = int(timestamp_str)
    except (ValueError, TypeError):
        return False

    # 1. Pengecekan batas waktu anti-replay (jendela 5 menit)
    if abs(time.time() - timestamp) > 300:
        return False

    # 2. Hitung HMAC-SHA256
    signed_payload = str(timestamp).encode("utf-8") + b"." + raw_body
    expected_hex = hmac.new(
        secret.encode("utf-8"),
        signed_payload,
        hashlib.sha256
    ).hexdigest()
    expected_signature = f"sha256={expected_hex}"

    # 3. Komparasi timing-safe
    return hmac.compare_digest(expected_signature, signature or "")
```

## Retry pengiriman & praktik terbaik idempoten

- **Balas Cepat dengan HTTP 200** — Kembalikan status HTTP `200` atau `204` dalam waktu kurang dari 3 detik. Tugas berat seperti pembuatan resi pengiriman atau email otomatis sebaiknya dimasukkan ke queue latar belakang.
- **Retry Otomatis dengan Exponential Backoff** — Bila server merchant mengalami gangguan (5xx) atau timeout, sistem Hashpay akan mengulang pengiriman hingga 3 kali dengan jeda waktu yang meningkat.
- **Tangani Event Secara Idempoten** — Fluktuasi koneksi internet dapat memicu pengiriman ganda untuk event yang sama. Catat `data.order_no` dan `X-Hashpay-Idempotency-Key` di database toko Anda. Jika order sudah diproses sebelumnya, balas dengan HTTP 200 tanpa memproses ganda.
- **Wajib Gunakan Raw Request Bytes** — Selalu hitung HMAC menggunakan raw request bytes murni dari socket jaringan. Melakukan parse JSON lalu men-serialize ulang ke string akan mengubah urutan kunci atau spasi, yang menyebabkan verifikasi signature gagal.

BAB 6.0 · SIKLUS & SETTLEMENT

# Siklus Hidup & Status Transaksi

> *Pahami siklus status yang dilewati oleh setiap order pembayaran dari tahap inisiasi hingga penyelesaian on-chain final, beserta webhook event terminal.*

## Transisi status transaksi

Setiap transaksi pelanggan bergerak mengikuti mesin status yang terdefinisi:

| STATUS | KLASIFIKASI | ARTI & KONDISI |
|---|---|---|
| `Created` | Antara (Intermediate) | Order berhasil dibuat di database; sesi checkout aktif menunggu pembayaran pelanggan. |
| `Pending` | Antara (Intermediate) | Transfer dana terdeteksi di blockchain; node watcher sedang menunggu jumlah konfirmasi blok yang cukup. |
| `Success` | Terminal (Final) | Pembayaran terkonfirmasi penuh on-chain. Saldo toko bertambah; memicu webhook `transaction.success`. |
| `Expired` | Terminal (Final) | Batas waktu kunci kurs 15 menit habis tanpa ada transfer masuk; memicu webhook `transaction.expired`. |
| `Failed` | Terminal (Final) | Eksekusi transfer di blockchain gagal atau direvert oleh jaringan. |
| `Cancelled` | Terminal (Final) | Order dibatalkan secara eksplisit sebelum pembayaran diselesaikan. |
| `Invalid` | Terminal (Exception) | Dana terdeteksi namun tidak cocok (misal: kurang bayar atau token salah); masuk antrean review manual ops. |

> 💡 **STATUS ANTARA VS STATUS TERMINAL**
>
> Hanya status terminal (`Success`, `Expired`, `Invalid`, `Failed`) yang mengirimkan notifikasi webhook ke server Anda. Status antara (`Created`, `Pending`) dapat dipantau langsung pada antarmuka checkout atau dicek via `POST /merchant/fiat/getOrder`.

## Penanganan anomali & review manual tim operasional

Saat pelanggan salah mengirim token atau mentransfer nominal yang kurang dari total tagihan, transaksi ditandai `Invalid` dengan tanda `manual_review: true`. Tim teknis Hashpay akan mengevaluasi kasus tersebut:

- Jika pelanggan melunasi sisa tagihan, tim ops dapat menyetujui pelunasan manual, yang akan memicu pengiriman webhook susulan `transaction.success` bertanda `manual_review: true`.
- Jika pesanan dibatalkan, dana on-chain akan dikembalikan langsung ke alamat wallet pengirim.

BAB 7.0 · SIKLUS & SETTLEMENT

# Settlement & Penarikan Dana

> *Bagaimana dana yang diterima di alamat deposit blockchain disapu, dikumpulkan, dan dicairkan ke dompet settlement atau rekening bank lokal di Indonesia.*

## Arsitektur penyapuan deposit otomatis (Sweep)

Saat pelanggan membayar tagihan, dana masuk ke alamat deposit on-chain unik. Worker latar belakang Hashpay secara rutin memantau saldo:

1. **Pencatatan Saldo Seketika** — Begitu transfer terkonfirmasi di blockchain, saldo aktif toko Anda ( `balance` ) langsung bertambah seketika. Anda tidak perlu menunggu proses sweep selesai untuk mengirimkan pesanan ke pelanggan.
2. **Penyapuan & Konsolidasi Otomatis** — Worker sistem secara berkala mengumpulkan dana dari alamat-alamat per-order ke dompet penampung utama toko Anda berdasarkan perhitungan biaya gas yang optimal.

## Pemicu sweep manual (api-key)

`POST` `/merchant/settlement`

Merchant dapat memicu proses penyapuan saldo on-demand secara instan untuk seluruh alamat deposit yang telah terbayar:

Trigger Sweep Manual — BASH

```
curl -X POST "https://api.hashpay.id/v2/mainnet/merchant/settlement" \
  -H "api-key: sec_live_9f83b27a8e..."
```

## Pencairan dana ke bank lokal & dompet kripto (Disbursement)

Untuk mencairkan keuntungan toko ke rekening bank lokal (BCA, Mandiri, BRI, BNI) atau dompet kripto eksternal, gunakan endpoint penarikan yang diotentikasi dengan token Merchant JWT:

- **Hitung Kuotasi & Biaya Transfer** — `GET /v2/mainnet/merchant/disbursement/calculate?amount=10000000&channel=BANK` mengembalikan kuotasi kurs konversi dan biaya transfer bank.
- **Kirim Pengajuan Penarikan** — `POST /v2/mainnet/merchant/disbursement` membuat instruksi pencairan dengan otorisasi 2FA.

BAB 8.0 · SIKLUS & SETTLEMENT

# Error & Kode Status

> *Standar envelope JSON error, daftar status HTTP, dan panduan penanganan retry pada endpoint API Hashpay.*

## Envelope JSON error terstandarisasi

Setiap kegagalan pemanggilan API Hashpay mengembalikan envelope JSON standar yang memuat `status_code`, `message`, dan `data: null`:

Contoh Envelope Error — JSON

```
{
  "status_code": 400,
  "message": "amount is below the minimum order limit of 10000 IDR",
  "data": null
}
```

## Daftar kode status HTTP

| KODE STATUS | KATEGORI | PENYEBAB & SOLUSI |
|---|---|---|
| `400` | Bad Request | Kesalahan validasi parameter (misal: nominal belanja di bawah batas minimal order). Periksa kembali isi body request. |
| `401` | Unauthorized | Header `api-key` tidak ada atau salah, token JWT kedaluwarsa, atau signature keliru. Periksa kredensial di dashboard. |
| `403` | Forbidden | Akun Anda tidak memiliki izin untuk mengakses resource atau fitur yang diminta. |
| `404` | Not Found | Nomor order, ID merchant, atau resource tidak ditemukan di sistem. |
| `429` | Too Many Requests | Batas pemanggilan rate limit terlampaui. Terapkan jeda (backoff) sebelum mencoba kembali. |
| `500` | Internal Server Error | Kendala sementara pada server. Coba beberapa saat lagi dengan jeda exponential backoff. |

⭐ **CATATAN PENTING**

Saat mengimplementasikan retry otomatis untuk error 429 atau 5xx, selalu terapkan jeda waktu bertingkat (exponential backoff) agar server tidak membebani gateway.

BAB 9.0 · PENGUJIAN & PELUNCURAN

# Sandbox & faucet

> *Uji alur penuh di testnet dengan token gratis sebelum menyentuh dana nyata. Formatnya identik dengan produksi.*

## Base URL sandbox

Gunakan `https://api.hashpay.id/v2/testnet`. Order, token, dan webhook sandbox terisolasi penuh dari produksi.

## Dapatkan token uji

Faucet mencetak token testnet dan gas ke alamat yang Anda kendalikan.

| | |
|---|---|
| GET | /faucet |
| POST | /faucet |
| GET | /faucet/status?address= |

💡 **CATATAN PENTING**

Faucet hanya untuk testnet dan dibatasi rate. Pakai tokennya untuk membayar order sandbox end-to-end.

## Simulasikan pembayaran

1. Buat order testnet dan buka checkout sandbox ( `/sandbox` ).
2. Kirim `totalAmount` persis dari wallet faucet ke alamat deposit.
3. Lihat order berubah jadi `Success` dan terima webhook `transaction.success`.

⚠️ **CATATAN PENTING**

Selalu kirim `totalAmount` persis (termasuk kode unik 3 digit). Kurang bayar ditandai `Underpaid`.

BAB 10.0 · PENGUJIAN & PELUNCURAN

# Go Live (Checklist Produksi)

> *Checklist persiapan akhir dan verifikasi peluncuran sebelum mengaktifkan integrasi Hashpay untuk transaksi mainnet nyata.*

## Peralihan konfigurasi ke lingkungan produksi

1. **Ubah Base API URL** — Arahkan client API backend Anda dari `https://api.hashpay.id/v2/testnet` ke `https://api.hashpay.id/v2/mainnet`.
2. **Pasang Kunci Rahasia Produksi** — Ganti kredensial testnet dengan `api-key` dan `merchant_id` produksi resmi dari dashboard toko Anda.
3. **Atur URL Notifikasi Webhook Produksi** — Masukkan endpoint webhook HTTPS publik toko Anda (contoh: `https://tokosaya.com/api/hashpay-webhook`) di menu **Pengaturan** → **Kunci API & Webhook**.
4. **Pastikan Domain Hosted Checkout Benar** — Pastikan URL checkout mengarah ke `https://webview.hashpay.id/` (tanpa imbuhan `/sandbox`).
5. **Daftarkan Rekening Bank atau Dompet Kripto** — Pastikan nomor rekening bank lokal atau alamat wallet penampung sudah disetujui di menu **Pengaturan** → **Rekening Penarikan**.

## Checklist keamanan sebelum peluncuran

- Secret Key disimpan ketat pada environment variable server / secret manager; tidak pernah terekspos di kode browser publik atau GitHub.
- Tanda tangan webhook diverifikasi secara kriptografis terhadap header `X-Hashpay-Signature` menggunakan Webhook Signing Secret produksi.
- Handler webhook bersifat idempoten dan mengabaikan event duplikat berdasarkan `X-Hashpay-Idempotency-Key` atau `order_no`.
- Proteksi anti-replay aktif: request dengan `X-Hashpay-Timestamp` yang berjarak lebih dari 300 detik otomatis ditolak.
- Endpoint webhook membalas dengan HTTP `200` atau `204` dalam waktu maksimal 3 detik, dan memproses order di latar belakang (queue).
- Worker rekonsiliasi cadangan aktif untuk memanggil `POST /merchant/fiat/getOrder` secara berkala.

> 🔒 **TRANSAKSI KRIPTO BERSIFAT FINAL**
>
> Transaksi on-chain tidak dapat ditarik kembali (irreversible). Pastikan seluruh mekanisme verifikasi signature dan pencatatan buku besar telah diuji tuntas di sandbox sebelum melayani pelanggan umum.