TikiPay Payment API - Complete Developer Guide

Basic-to-advanced integration guide for creating hosted payments, redirecting customers, verifying payment status, processing webhooks, handling errors and deploying safely.

1. Integration architecture

Recommended architecture: Browser/mobile app -> Your backend -> TikiPay API. Keep the Brand API Key on your backend. Never expose it in public browser JavaScript, HTML or an APK.

2. Full payment flow

Step	What to do
1	Create your local order as PENDING.
2	Call POST /api/payment/create from your backend.
3	Save the returned invoice_id with your local order.
4	Redirect the customer to payment_url.
5	Receive the browser return and/or webhook.
6	Call POST /api/payment/verify from your backend.
7	Fulfill only when status is COMPLETED.
8	Make fulfillment idempotent so retries cannot deliver twice.
9	Log invoice ID, transaction ID, amount, method and status.

3. Authentication

Use the Brand API Key for website/server Create and Verify calls. Preferred header: **API-KEY: YOUR_BRAND_API_KEY**. Device Keys are for supported device/SMS connector workflows, not for website payment API calls.

4. Create Invoice

POST /api/payment/create

Required core fields: amount and success_url. Recommended fields: cus_name, cus_email, cancel_url, webhook_url and metadata. Store your local order ID inside metadata.

```
curl -X POST "{{BASE_URL}}/api/payment/create" \
-H "Content-Type: application/json" \
-H "API-KEY: YOUR_BRAND_API_KEY" \
-d '{
"cus_name": "John Doe",
"cus_email": "john@example.com",
"amount": 1200,
"success_url": "https://merchant.example/payment/success",
"cancel_url": "https://merchant.example/payment/cancel",
"webhook_url": "https://merchant.example/api/tikipay/webhook",
"metadata": {"order_id":"ORDER-1001"}
}'
```

Create success response

```
{
"status": 1,
"message": "Payment Link",
"payment_url": "{{BASE_URL}}/api/execute/SESSION_ID",
"invoice_id": "SESSION_ID"
}
```

5. Redirect customer

After Create succeeds, save invoice_id and redirect the browser to payment_url. Do not put the Brand API Key in the redirect or frontend.

6. Verify Payment

POST /api/payment/verify

Send invoice_id (recommended) or transaction_id. Verify from your backend before marking the local order paid.

```
curl -X POST "{{BASE_URL}}/api/payment/verify" \
-H "Content-Type: application/json" \
-H "API-KEY: YOUR_BRAND_API_KEY" \
-d '{
"invoice_id": "SESSION_ID"
}'
```

Verify response

```
{
"cus_name": "John Doe",
"cus_email": "john@example.com",
"amount": 1200,
"invoice_id": "SESSION_ID",
"transaction_id": "TXN123456789",
"metadata": {"order_id":"ORDER-1001"},
"payment_method": "bkash",
"status": "COMPLETED"
}
```

7. Payment statuses

Status	Merchant action
COMPLETED	Payment verified. Mark paid and fulfill exactly once.
PENDING	Keep the order open. Re-check Verify later.
ERROR	Do not fulfill. Handle failure/cancel flow.

8. Webhook handling

For the native API, TikiPay can POST paymentMethod, transactionId, paymentAmount, paymentFee, invoice_id and status to the webhook_url supplied during Create. Treat the webhook as a notification, then call Verify before fulfillment. Webhooks may be retried, so handlers must be idempotent.

9. HTTP errors and recovery

HTTP / condition	Meaning / action
400	Invalid parameters. Check amount, URLs, invoice ID and JSON.
401	Invalid API key. Check the Brand API Key and auth header.
404	Transaction not found. Check the reference and account.

500	Server processing error. Do not fulfill; log and retry safely.
Timeout	Final state is unknown. Re-run Verify using the same invoice ID.

10. Language examples available on the live docs page

The live TikiPay developer documentation includes request examples for cURL, server-side JavaScript, PHP, Laravel, Python and Node.js for both Create and Verify.

11. Production checklist

Use HTTPS. Keep API keys server-side. Validate request data. Store invoice IDs. Verify before fulfillment. Make fulfillment idempotent. Log responses. Test COMPLETED, PENDING, invalid-key, invalid-parameter and timeout scenarios before going live.