openbcp DOCS

USDT · BNB SMART CHAIN · HOSTED CHECKOUT

Developer Documentation

Accept USDT payments on BNB Smart Chain. Hosted checkout, fully managed infrastructure, transparent per-transaction fees.

<$0.01
BSC network fee, any amount

~9s
3 on-chain confirmations

5 min
to first payment

CONTENTS

What's inside

01 Overview

Your backend calls the openbcp API to create a payment invoice. openbcp returns a `payment_url` — a hosted checkout page on openbcp.com. Redirect your customer there. openbcp handles the QR code, countdown timer, live confirmation, and fee disclosure. After 3 on-chain confirmations openbcp fires a signed webhook to your server.

BSC transaction fees are under **$0.01** regardless of amount. A small platform fee is added at invoice creation — fully disclosed to the customer on the checkout page.

02 How it works

1. Your server calls **POST /api/v1/payment** → receives `payment_url`
2. Redirect customer to `payment_url` — openbcp hosts the checkout
3. Customer scans QR or copies address, sends USDT on BSC
4. openbcp POSTs a signed webhook to your `callback_url`

The hosted checkout QR code encodes an **EIP-681** URI — wallets like Trust Wallet and MetaMask Mobile auto-fill the token, recipient, and amount when scanned.

03 Quick Start

Get your first payment running in under 5 minutes.

1 · GET YOUR API KEY

Go to **Dashboard → Integration**. Copy your `X-openbcp-Api-Key`. Keep it secret — only use it server-side.

2 · CREATE A PAYMENT AND REDIRECT

PYTHON COPY

```python
import requests

resp = requests.post(
    "https://openbcp.com/api/v1/payment",
    headers={"X-openbcp-Api-Key": "your-api-key"},
    json={
        "amount_usdt": "29.99",
        "external_id": "order-123",
        "callback_url": "https://yoursite.com/webhooks/openbcp",
    }
)
data = resp.json()

# Redirect the customer - openbcp handles everything from here
return redirect(data["payment_url"])
```

3 · RECEIVE THE WEBHOOK

PYTHON COPY

```python
import hmac, hashlib

@app.post("/webhooks/openbcp")
def webhook():
    sig = request.headers["X-openbcp-Signature"]
    ts  = request.headers["X-openbcp-Timestamp"]
    body = request.get_data()
    msg = ts.encode() + b"." + body

    expected = hmac.new(
        YOUR_API_KEY.encode(), msg, hashlib.sha256
    ).hexdigest()

    if not hmac.compare_digest(sig, expected):
        return "Unauthorized", 401

    data = request.json()
    if data["status"] in ("PAID", "OVERPAID"):
        fulfill_order(data["external_id"])
    elif data["status"] in ("EXPIRED", "CANCELLED"):
        cancel_order(data["external_id"])
    return "", 202
```

04 Authentication

All API requests must include your API key in the `X-openbcp-Api-Key` header.

HTTP COPY

```http
X-openbcp-Api-Key: your-api-key-here
```

> ! Never expose your API key in frontend JavaScript, mobile apps, or public repositories. It must **only** be used server-side.

Regenerate your key under **Integration → API Key → Regenerate**. Old keys stop working immediately.

05 Hosted Checkout

openbcp hosts the payment page at `https://openbcp.com/pay/<invoice_id>`. You never build checkout UI — just redirect.

WHY HOSTED?

Trust	Customers see openbcp.com in the browser — a known payment processor, not a random merchant domain
Transparency	The platform fee is always shown. Amount can't be faked.
Smart QR	Encodes EIP-681 URI — wallets auto-fill token, address, and amount
No frontend	QR code, countdown, live polling, paid/expired states — all handled

THE FLOW

FLOW COPY

```
# 1. Your server creates the invoice
POST /api/v1/payment → { payment_url: "https://openbcp.com/pay/42", ... }

# 2. Redirect customer
window.location.href = payment_url

# 3. openbcp.com/pay/42 shows:
#    - Amount: 30.02 USDT (29.99 product + 0.03 platform fee)
#    - QR code (EIP-681, pre-fills wallet)
#    - Copy address button
#    - Live countdown + auto-confirm on payment

# 4. On confirmation → webhook fires to your callback_url
```

06 Create Payment

POST `/api/v1/payment`

Creates an invoice. A platform fee is automatically added to the customer-facing amount. Returns a `payment_url` — redirect the customer there.

REQUEST HEADERS

HEADERS COPY

```
X-openbcp-Api-Key: your-api-key
Content-Type: application/json
```

REQUEST BODY

FIELD	TYPE	DESCRIPTION
`amount_usdt`	string REQUIRED	Your product price in USDT. Pass as a string — e.g. "29.99". The platform fee is added on top.
`external_id`	string	Your internal order / reference ID. Returned in webhooks.
`callback_url`	string	HTTPS URL that receives signed webhook POSTs on status change.
`metadata`	object	Arbitrary JSON dict stored with the invoice.
`amount_fiat`	number	Original fiat amount (informational only).
`fiat_currency`	string	ISO 4217 code, e.g. "USD".

RESPONSE

201 CREATED COPY

```
{
  "status": "success",
  "payment_url": "https://openbcp.com/pay/42",    // redirect customer here
  "invoice_id": 42,
  "amount_usdt": "30.02",            // gross - what customer pays
  "merchant_amount_usdt": "29.99",   // your product price
  "platform_fee_usdt": "0.03",       // openbcp platform fee
  "deposit_address": "0xAbCd...1234",
  "network": "BEP20",
  "token": "USDT",
  "expires_at": "2026-06-09T12:00:00Z"
}
```

Each invoice gets a fresh BSC deposit address. Don't create duplicate invoices for the same order — reuse the existing `invoice_id` if the customer needs to retry.

07 Check Payment Status

`GET` `/api/v1/payment/{invoice_id}/status`

Public endpoint — no API key needed. Returns current status and transaction list.

200 OK COPY

```
{
  "invoice_id": 42,
  "payment_status": "PAID",
  "amount_usdt": "30.02",
  "expires_at": "2026-06-09T12:00:00Z",
  "transactions": [
    {
      "txid": "0xabc...",
      "amount_usdt": "30.02",
      "confirmations": 3,
      "required_confirmations": 3,
      "status": "CONFIRMED"
    }
  ]
}
```

STATUS VALUES

VALUE	MEANING
UNPAID	Waiting — no transaction detected yet
PARTIAL	Transaction detected but below required amount
PAID	Exact or sufficient amount confirmed on-chain
OVERPAID	More than requested amount confirmed
EXPIRED	Invoice timed out with no payment — webhook fired
CANCELLED	Manually cancelled by the customer — webhook fired

There is also an authenticated endpoint that returns the full invoice including `external_id` and `created_at`: `GET /api/v1/payment/{invoice_id}` requires `X-openbcp-Api-Key`.

08 Error Codes

All errors return JSON with `"status": "error"` and a `"message"` field.

HTTP	CAUSE
400	Missing or invalid fields in request body
401	Missing or invalid X-openbcp-Api-Key header
404	Invoice not found
503	Blockchain node unavailable — retry after a short delay
500	Internal error — retry with exponential backoff

09 Receiving Webhook Events

openbcp sends a signed POST to your `callback_url` on every terminal status change — not just on payment success.

EVENTS THAT TRIGGER A WEBHOOK

STATUS	WHEN	PAID	TXID
PAID	Exact / sufficient amount confirmed (3 blocks, ~9s)	true	on-chain hash
OVERPAID	More than requested amount confirmed	true	on-chain hash
PARTIAL	Transaction detected but below required amount	false	on-chain hash
EXPIRED	Invoice timed out with no payment	false	null
CANCELLED	Customer manually cancelled the order	false	null

PAYLOAD

PAYLOAD COPY

```
{
  "invoice_id": 42,
  "external_id": "order-123",
  "status": "PAID",          // PAID | OVERPAID | PARTIAL | EXPIRED | CANCELLED
  "amount_usdt": "30.000000",
  "deposit_address": "0x...",
  "txid": "0xabc123...",      // null for EXPIRED / CANCELLED
  "paid": true,              // false for PARTIAL / EXPIRED / CANCELLED
  "paid_at": "2026-06-08T14:32:11Z"
}
```

Your handler must return **HTTP 202** to acknowledge delivery. If your endpoint does not return 202, openbcp retries with exponential backoff up to the configured maximum. Use `invoice_id + status` as the deduplication key.

10 Verifying Signatures

Every webhook includes two headers:

`X-openbcp-Signature`

HMAC-SHA256 of `{timestamp}.{raw_body}`, keyed with your API key

`X-openbcp-Timestamp`

Unix timestamp of when the event was sent (reject if >5 min old)

Always verify the signature before processing the payload. Compute the HMAC on the **raw request bytes** — not re-serialized JSON.

PYTHON COPY

```python
import hmac, hashlib, time

def verify_openbcp_signature(raw_body: bytes, sig: str,
                             timestamp: str, api_key: str) -> bool:
    if abs(time.time() - int(timestamp)) > 300:
        return False  # reject stale webhooks
    msg = timestamp.encode() + b"." + raw_body
    expected = hmac.new(api_key.encode(), msg, hashlib.sha256).hexdigest()
    return hmac.compare_digest(expected, sig)
```

NODE.JS COPY

```javascript
const crypto = require('crypto');

function verifySignature(rawBody, sig, timestamp, apiKey) {
  if (Math.abs(Date.now() / 1000 - parseInt(timestamp)) > 300) return false;
  const msg = Buffer.concat([Buffer.from(timestamp + '.'), rawBody]);
  const expected = crypto
    .createHmac('sha256', apiKey)
    .update(msg)
    .digest('hex');
  return crypto.timingSafeEqual(
    Buffer.from(expected),
    Buffer.from(sig)
  );
}
```

11 Python SDK

A thin typed wrapper around the REST API for Python backends. Includes webhook verification.

INSTALL COPY

```
pip install git+https://github.com/usdt-paygate/paygate-python.git
```

USAGE

PYTHON COPY

```python
from paygate import PayGateClient, verify_webhook

client = PayGateClient(
    base_url="https://openbcp.com",
    api_key="your-api-key",
)

# Create invoice - redirect customer to payment_url
invoice = client.create_payment(
    "29.99",
    external_id="order-123",
    callback_url="https://yoursite.com/webhooks/openbcp",
)
return redirect(invoice.payment_url)  # openbcp hosts the checkout

# In your webhook handler
if verify_webhook(raw_body, sig_header, "your-api-key"):
    if payload["status"] in ("PAID", "OVERPAID"):
        fulfill_order(payload["external_id"])
    elif payload["status"] in ("EXPIRED", "CANCELLED"):
        cancel_order(payload["external_id"])
```

INVOICE FIELDS

FIELD	TYPE	DESCRIPTION
`payment_url`	str	Hosted checkout URL — redirect the customer here
`invoice_id`	int	Numeric invoice identifier
`amount_usdt`	Decimal	Gross amount customer pays (includes fee)
`merchant_amount_usdt`	Decimal	Your product price (what you receive)
`platform_fee_usdt`	Decimal	openbcp platform fee
`payment_status`	str	UNPAID / PARTIAL / PAID / OVERPAID / EXPIRED / CANCELLED
`expires_at`	datetime	Invoice expiry (UTC)
`transactions`	list	On-chain transaction objects

12 Node.js SDK

Zero-dependency TypeScript SDK for Node 18+. Uses native fetch — no extra packages required.

INSTALL COPY

```
npm install github:usdt-paygate/paygate-node
```

USAGE

TYPESCRIPT COPY

```
import { PayGateClient, verifyWebhook, WebhookVerificationError } from '@openbcp/
paygate';

const client = new PayGateClient({
  baseUrl: 'https://openbcp.com',
  apiKey: 'your-api-key',
});

// Create invoice - redirect customer to payment_url
const invoice = await client.createPayment('29.99', {
  external_id: 'order-123',
  callback_url: 'https://yoursite.com/webhooks/openbcp',
  success_url: 'https://yoursite.com/order/confirmed',
});
res.redirect(invoice.payment_url);

// In your Express webhook handler
app.post('/webhooks/openbcp', async (req, res) => {
  try {
    verifyWebhook(
      req.body,
      req.headers['x-openbcp-signature'],
      req.headers['x-openbcp-timestamp'],
      process.env.PAYGATE_API_KEY,
    );
  } catch (e) {
    if (e instanceof WebhookVerificationError) return res.sendStatus(400);
  }
  const data = JSON.parse(req.body);
  if (['PAID', 'OVERPAID'].includes(data.status)) await
fulfillOrder(data.external_id);
  else if (['EXPIRED', 'CANCELLED'].includes(data.status)) await
cancelOrder(data.external_id);
  res.sendStatus(202);
});
```

API REFERENCE

METHOD	DESCRIPTION
`createPayment(amount, options?)`	Create invoice. Returns Invoice with payment_url.
`getPayment(invoiceId)`	Full invoice including transactions. Requires auth.
`getPaymentStatus(invoiceId)`	Public status endpoint — no auth, CORS-enabled.
`pollUntilPaid(invoiceId, options?)`	Async poll loop — resolves when paid, rejects on EXPIRED/CANCELLED.
`verifyWebhook(payload, sig, ts, key)`	Throws WebhookVerificationError on invalid signature.

GitHub: `usdt-paygate/paygate-node`

13 Go SDK

Standard-library-only Go client with typed structs, context support, and HMAC webhook verification.

INSTALL COPY

```
go get github.com/usdt-paygate/paygate-go
```

USAGE

GO COPY

```go
import paygate "github.com/usdt-paygate/paygate-go"

client := paygate.NewClient("https://openbcp.com", "your-api-key")

// Create invoice
successURL := "https://yoursite.com/order/confirmed"
invoice, err := client.CreatePayment(ctx, paygate.CreatePaymentRequest{
    AmountUSDT:  "29.99",
    ExternalID:  &[]string{"order-123"}[0],
    CallbackURL: &[]string{"https://yoursite.com/webhooks/openbcp"}[0],
    SuccessURL:  &successURL,
})
// Redirect customer to *invoice.PaymentURL

// In your webhook handler
func webhookHandler(w http.ResponseWriter, r *http.Request) {
    body, _ := io.ReadAll(r.Body)
    err := paygate.VerifyWebhook(
        body,
        r.Header.Get("X-openbcp-Signature"),
        r.Header.Get("X-openbcp-Timestamp"),
        os.Getenv("PAYGATE_API_KEY"),
    )
    if err != nil { w.WriteHeader(http.StatusBadRequest); return }

    var payload map[string]interface{}
    json.Unmarshal(body, &payload)
    switch payload["status"] {
    case "PAID", "OVERPAID": fulfillOrder(payload["external_id"])
    case "EXPIRED", "CANCELLED": cancelOrder(payload["external_id"])
    }
    w.WriteHeader(http.StatusAccepted)
}
```

API REFERENCE

METHOD	DESCRIPTION
`NewClient(baseURL, apiKey)`	Create client with 20s default timeout.
`CreatePayment(ctx, req)`	Create invoice. Returns *Invoice with PaymentURL.
`GetPayment(ctx, invoiceID)`	Full invoice including transactions. Requires auth.
`GetPaymentStatus(ctx, invoiceID)`	Public status endpoint — no auth.
`PollUntilPaid(ctx, invoiceID, interval, partial)`	Blocks until paid; ctx cancellation stops the loop.

METHOD	DESCRIPTION
`VerifyWebhook(payload, sig, ts, key)`	Returns *WebhookVerificationError on failure.

GitHub: `usdt-paygate/paygate-go`

14 MCP — AI Assistant Integration

Use openbcp directly from Claude Desktop or any MCP-compatible AI. Tell Claude to create a payment in plain English — it returns the `payment_url` you can share instantly.

SETUP

Add to your Claude Desktop config `~/Library/Application Support/Claude/claude_desktop_config.json` on macOS:

CLAUDE_DESKTOP_CONFIG.JSON COPY

```
{
  "mcpServers": {
    "openbcp": {
      "command": "uvx",
      "args": [
        "--from",
        "git+https://github.com/usdt-paygate/paygate-mcp.git",
        "paygate-mcp"
      ],
      "env": {
        "PAYGATE_URL": "https://openbcp.com",
        "PAYGATE_API_KEY": "your-api-key"
      }
    }
  }
}
```

Restart Claude Desktop. Then just type: `"Create a $50 USDT payment for order 123"` — Claude returns the payment_url immediately.

AVAILABLE TOOLS

TOOL	DESCRIPTION
`create_payment`	Create invoice, returns payment_url to share with customer

TOOL	DESCRIPTION
`get_payment`	Full invoice details with all on-chain transactions
`check_payment_status`	Quick status check with a plain-English summary

openbcp DOCS

Accept USDT on BNB Smart Chain in minutes.

openbcp.com/docs · API v1 · This document mirrors the official online documentation.