

Shopdm Pay URL-Based Payment Integration

Version 0.2.1-beta

This documentation explains how to integrate Shopdm Pay's URL-based one-time payment system into your application or website. This method allows merchants to create a payment link by specifying the total amount, payment reason, and redirect customers to a site after payment.

How It Works

Merchants can create payment links by appending URL parameters to the Shopdm Pay payment URL. These parameters define the total amount and reason for payment. Optionally, merchants can also provide an `invoice_id` to associate the payment, custom data for other processing and a signature to secure the url. Merchants can configure a redirect URL where customers will be sent after completing the payment and a webhook to receive payment alerts.

Live payment link example

Unset

```
https://pay.shopdm.store/dom-software?total_xcd=1.00&reason=Order+201234
```

IMPORTANT NOTE: Unsigned urls can have their total or reason changed by a buyer, simply by changing the url. They are allowed, but they are meant for semi-technical environments where it is feasible to check the payment alerts from the merchant dashboard after payment. In fully automated environments, merchants should always use signed urls or trust only the webhook alerts.

TLDR: Integrate in 4 Steps, all on the frontend

1. **Generate Payment URL** Construct the payment URL by appending the parameters to the base URL. Ensure all parameters are URL-encoded. You can get your merchant handle from your merchant dashboard, or by asking the Shopdm Pay team.

NOTE: The code below is all you'll need to do on the frontend. It will work if pasted into an index.html file.

JavaScript

```
<script>
/***** STEP 1: calculate values *****/
const cart = {
  '1':{id: '1', price: 10, quantity: 3, itemName:'chicken'},
  '2':{id: '2', price: 5, quantity: 4, itemName:'fries'},
}
/** calculating in cents to avoid floating point errors */
const totalXcdCents = Object.values(cart).reduce((total, item) =>
{
  total += Math.round(item.price * 100) * item.quantity;
  return total;
}, 0);
/** convert back to dollars */
const totalXcd = (totalXcdCents / 100).toFixed(2);
/***** STEP 2: sign values & generate link *****/
async function generatePaymentLink() {
  try {
    const signingApiUrl
    =
    "https://us-central1-shop-dm-dev.cloudfunctions.net/api/v1/pay/generate-signature"
    const payload = {
      total_xcd: totalXcd,
      reason: "Order123",
      invoice_id: "123",
      custom: "additional info",
      redirect: true,

      //any value for redirect will trigger a redirect, including
      //false. Omit the key to turn off redirects.
      webhook: true //any value will trigger the
      //webhook, including false, omit the key to turn off webhooks
    }
  }
}
```

```

    }
    const response = await fetch(signingApiUrl, {
      method: "POST",
      headers: {"Content-Type": "application/json"},
      body: JSON.stringify({payload: payload})
    })
    if (!response.ok) {
      throw new Error("Failed to fetch signature")
    }
    const {signature} = await response.json()
    const merchantHandle = "dom-software"
    const paymentUrl =
      `https://pay-dm-dev.web.app/${merchantHandle}?total_xcd=${payload.total_xcd}&reason=${payload.reason}&invoice_id=${payload.invoice_id}&custom=${payload.custom}&signature=${signature}&redirect=${payload.redirect}&webhook=${payload.webhook}`;
    console.log("Generated payment url:", paymentUrl);
  } catch (error) {
    console.error("Error generating payment link:", error);
  }
}
window.onload = generatePaymentLink;
</script>

```

2. **Embed the Link** Add the generated payment URL to a button or link on your website:

```

Unset
<a
href="https://pay-dm-dev.web.app/dom-software?total_xcd=50.00&reason=Order123&invoice_id=123&custom=additional%20info&signature=9daf0e9c8f62b1328296beaa73e50e464e69999c549fda6f8e9ba5d458996f1a&redirect=true&webhook=true"
target="_blank"
>

```

Pay Now

3. **Handle Redirects (optional):** Pass `redirect=true` in the url. To set up the redirect, provide your redirect url to the Shopdm Pay team via `pay@shopdm.store`. If a `redirectUrl` is provided, customers will be redirected to this URL. Ensure the redirect URL is configured to handle the return from Shopdm Pay (e.g., display a success message or update order status). You can use the `{INVOICE_ID}` placeholder to handle your customer experience e.g. https://www.testshop.com/order/{INVOICE_ID}/payment-success
4. **Handle Webhooks (optional):** Pass `webhook=true` in the url. To set up the webhook, provide your webhook url to the Shopdm Pay team via `pay@shopdm.store`. If a `webhookUrl` is provided, it will receive a payload after every successful payment made with `webhook=true` in the url. More details below.

Supported URL Parameters

Parameter	Required	Description	Note
<code>total_xcd</code>	Yes	The total payment amount in XCD currency. Acceptable format: Must be a positive number, with optional decimal point e.g. 100, 0.1, 25.19	
<code>reason</code>	No	The reason for the payment (e.g., "Order 1234"). Acceptable format: letters, numbers, spaces, commas, periods, underscores and hyphens. Max length 255 chars.	If not provided, the customer can specify it. Sanitized to remove disallowed characters
<code>invoice_id</code>	No	A unique identifier you can use to associate the payment with an object in	Sanitized to remove disallowed characters. Can be passed into

		<p>your database</p> <p>Acceptable format: letters, numbers, spaces, commas, periods, underscores and hyphens. Max length 255 chars.</p>	<p>your returnUrl if you use the "{INVOICE}" placeholder</p>
custom	No	<p>A general field to pass any other information relevant to your payment</p> <p>Acceptable format: letters, numbers, spaces, commas, periods, underscores and hyphens. Max length 255 chars.</p>	<p>Sanitized to remove disallowed characters</p>
redirect	No	<p>If not provided, no redirect will occur, even if a returnUrl is configured for the payee's account.</p> <p>If any value is provided, it will trigger a redirect.</p> <p>This is to allow payees to control whether they want to display receipts or redirect the customer.</p>	<p>// <input checked="" type="checkbox"/> Enables redirect redirect: true</p> <p>// <input checked="" type="checkbox"/> Also enables redirect redirect: "false"</p> <p>// <input checked="" type="checkbox"/> Do NOT do this to disable redirect redirect: false // still triggers redirect!</p> <p>// <input checked="" type="checkbox"/> Correct way to disable redirect // do not include the redirect key</p>
webhook	No	<p>If not provided, the webhook will not be called, even if a webhookUrl is configured for the payee's account.</p> <p>If any value is provided, the webhook will be called.</p> <p>This is to allow payees to control whether they want to hit their webhook or</p>	<p>// <input checked="" type="checkbox"/> Enables webhook webhook: true</p> <p>// <input checked="" type="checkbox"/> Also enables webhook webhook: "false"</p> <p>// <input checked="" type="checkbox"/> Do NOT do this to disable webhook webhook: false // still triggers webhook!</p> <p>// <input checked="" type="checkbox"/> Correct way to</p>

		only process the payment	disable webhook // do not include the webhook key
signature	No (but recommended)	Validates that the buyer has not tampered with the url. Generated via our API	When provided, we will verify that the user has not changed the total_xcd, reason, invoice_id, or custom and if they have, an error message will be displayed

Base Payment URL

Unset

DEVELOPMENT

https://pay-dm-dev.web.app/{merchant_handle_id}

PRODUCTION

https://pay.shopdm.store/{merchant_handle_id}

Signature Generation API

Unset

DEVELOPMENT

<https://us-central1-shop-dm-dev.cloudfunctions.net/api/v1/pay/generate-signature>

PRODUCTION

<https://us-central1-shop-dm.cloudfunctions.net/api/v1/pay/generate-signature>

Webhook Documentation

Overview

The `payment.successful` webhook is triggered when a payment is successfully processed. Merchants can listen to this webhook to update their systems in real-time upon payment completion.

Example Payload

Unset

```
{
  "id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "event": "payment.successful",
  "created_at": 1700000000,
  "api_version": "0.2.1-beta",
  "merchant_id": "76f6ca55-cd77-48d4-8a12-e0e5eb2e5d11",
  "data": {
    "object_type": "oneTimePayment",
    "object_id": "dc760ceb-30e5-421f-9a26-3c3a3e0485ce",
    "reference": "INV-2024-001",
    "currency": "xcd",
    "amount_in_currency": 50.00,
    "amount_xcd": 50.00,
    "reason": "Order 12345"
  },
  "fee_data": {
    "customer_fee_xcd": 1.55,
    "merchant_fee_xcd": 1.50,
    "net_amount_xcd": 48.50,
    "amount_paid_by_customer_xcd": 51.55
  },
  "metadata": {
    "invoice_id": "INV-2024-001",
    "custom": "userId 1f0b38c3-3db7-4a31-966b-e0f68ed0f038"
  },
  "live": true
}
```

Fields

Each webhook request includes the following fields:

Field Name	Type	Description
id	string (UUID)	Unique identifier for the webhook event.
event	string	The type of event. In this case, "payment.successful".
created_at	integer (timestamp)	Unix timestamp representing when the event was created.
api_version	string	API version used for this webhook.
merchant_id	string (UUID)	Unique identifier for the merchant receiving the webhook.
live	boolean	true if the event occurred in the live environment, false for test/sandbox.
data (Object)	object	Contains payment-related information.
└ object_type	string	The type of payment object ("oneTimePayment").
└ object_id	string (UUID)	Unique identifier for the payment object.
└ reference	string	A reference ID for the transaction (e.g., invoice number).
└ currency	string	The currency code in lowercase (e.g., "xcd").
└ amount_in_currency	number	Payment amount in the original currency.
└ amount_xcd	number	Payment amount converted to XCD.
└ reason	string	Reason or description of the payment.
fee_data (Object)	object	Contains breakdown of fees and net amount.
└ customer_fee_xcd	number	Fee paid by the customer in XCD.
└ merchant_fee_xcd	number	Fee deducted from the merchant in XCD.
└ net_amount_xcd	number	Net amount received by the merchant after fees.

<code>└ amount_paid_by_customer_xcd</code>	number	Total amount paid by the customer, including fees.
metadata (Object)	object	Additional metadata related to the payment.
<code>└ invoice_id</code>	string	Invoice ID associated with the payment.
<code>└ custom</code>	string	Any custom metadata, e.g., user identifiers.
<code>└ redirect</code>	string	The value passed in the url for the redirect key. If truthy, it signals that a redirect was triggered
<code>└ webhook</code>	string	The value passed in the url for the webhook key. It determines whether the webhook will be fired, so it should always be truthy when it hits the webhook

Headers

Each webhook request includes the following headers:

Header Name	Description
<code>Content-Type</code>	Always <code>application/json</code> .
<code>X-Shopdm-Signature</code>	HMAC-SHA256 signature of the request body, used for verification.

Verifying the Webhook Signature

To ensure the webhook request is authentic, verify the `X-Shopdm-Signature` header. The signature is computed using HMAC-SHA256 with your webhook secret.

Steps to Verify the Signature

1. Retrieve the `X-Shopdm-Signature` from the headers.
2. Compute the HMAC-SHA256 hash using your webhook secret and the raw request body.
3. Compare the computed hash with the `X-Shopdm-Signature`. If they match, the request is authentic.

Example Code for Backend Signature Verification (Node.js)

JavaScript

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

function verifySignature(req, webhookSecret) {
  const signature = req.headers['x-shopdm-signature'];
  const payload = req.body
  //Sort keys to ensure deterministic serialization
  const sortedPayload = JSON.stringify(
    payload,
    Object.keys(payload).sort()
  );

  const expectedSignature = crypto
    .createHmac('sha256', webhookSecret)
    .update(sortedPayload, 'utf8')
    .digest('hex');

  return signature === expectedSignature;
}
```

Handling the Webhook

1. Listen for incoming webhook requests.
2. Verify the signature using the method above.
3. Process the payment details from the `data` and `fee_data` objects.
4. Store relevant information in your system.
5. Return a `200 OK` response if processed successfully.

Security Recommendations

- Validate incoming webhooks by verifying the signature.
- Ensure `invoice_id` matches known transactions.
- Ensure `amount_xcd` matches the total on your invoice object / passed in the url param.
- Ensure your webhook function can handle being called more than once (idempotent), to avoid duplicate processing. E.g. by marking your invoice object as processed: true and checking for that before doing any processing.

This webhook allows merchants to keep their systems updated in real-time when a payment is successfully completed.

Other Configurations

These configurations will affect your integration but they are not provided as URL parameters.

Supported URL Parameters

Parameter	Required	Description	Note
Redirect url	No	<p>The user will be redirected to this link after paying.</p> <p>The invoice_id you provide can be embedded in this link by using the variable {INVOICE_ID} e.g.</p> <p>www.merchant-site.com/{INVOICE_ID}/payment-success. You can use this to do order specific logic after the redirect.</p> <p>Acceptable format: Must be a valid url at a domain controlled by the merchant.</p>	
Webhook Url	No	<p>If you provide Shopdm Pay with your webhook api. We will call it after every successful payment.</p> <p>Acceptable format: Must be a valid url able to receive a POST.</p>	
Merchant Secret	No (Only if using Webhook)	<p>If you provide us with a webhook, we will provide you with a Merchant Secret to validate incoming requests.</p> <p>You will use it to generate signatures from the incoming requests. Because only you and Shopdm Pay know your secret, you can be sure that the alerts are coming from us.</p>	<p>Do not put this into frontend code, keep it only on your secure backend. Avoid committing this in your version control system, instead use environment variables.</p>

Security Features

Input Sanitization

All input parameters are sanitized on the backend to prevent:

- Cross-Site Scripting (XSS)
- SQL Injection
- Malicious payloads

Tampering Protection

Our API includes a signature-based verification mechanism to avoid users from tampering with your payment links. Sign your parameters on the frontend before generating your link, and include the signature in your link.

Secure Webhook alerts

Secure backend webhook alerts allow merchants to process your orders after successful payment. All alerts are signed with your merchant secret, so you can easily verify that the alert came from us. Just re-sign the webhook body and compare it with the signature in the header `X-Shopdm-Signature` (sample code above).

Risks to protect against

1. **Phishing attacks:**
 - a. **Description:** If an attacker compromises your site, they can redirect your buyer to an imitation site and steal their details.
 - b. **Mitigation:**
 - i. Protect yourself from XSS attacks that can compromise your site e.g. by sanitizing input fields and rendering content safely.
 - ii. Maintaining secure passwords and hardened servers regarding your code base.
 - iii. Monitoring your post-purchase alerts from Shopdm Pay via your webhooks, email and alerts on your Merchant dashboard.
2. **Url tampering:**
 - a. **Description:** An attacker or your buyer can change the total on the link you generate and pay you a different amount that you expect.
 - b. **Mitigation:**
 - i. Provide a signature with your url. Sign your values using our signing API.
3. **Webhook abuse:**

- a. **Description:** An attacker can hit your webhook with a fake alert and cause your system to change the status of your order to paid, even though they have not paid.
 - b. **Mitigation:**
 - i. Verify the signature sent to your webhook in the `X-Shopdm-Signature` header. To do this, Regenerate a signature on your server using your Merchant Secret and compare it to the signature from the header. **Note:** never transmit your Merchant Secret.
-

Testing Your Integration

1. **Use Test Parameters** Test with the development version of the payment URL or with minimal amounts to ensure correct behavior. Example:

Unset

```
https://pay-dm-dev.web.app/your-merchant-handle?total_xcd=1&reason=Test%20Order
```

2. **Verify Signing Behavior** Confirm that you can successfully sign your parameters and that you see no errors when you visit your payment url.
 3. **Verify Redirect Behavior** Confirm that the redirect works as expected by completing a test payment and ensuring the user lands on the intended page.
 4. **Verify Webhook Behavior** Confirm that the webhook works as expected by completing a test payment, verifying the signature and processing your order in your database.
-

Upcoming Features

- To be configurable from admin panel
 - Redirect url
 - Merchant webhook

For further assistance, contact Shopdm Pay support at davidson@shopdm.store.