

**Low Fees.  
Pay Per Use.  
Viral Growth.**

**Welcome to Dropp** 

As a Dropp Merchant, you have limitless possibilities to attract new customers.

Let's Get Started

# Let's Go

<b>Start Here</b>	<b>3</b>
<i>Create Profile</i>	3
<b>Navigating this Guide</b>	<b>4</b>
<b>Concepts</b>	<b>5</b>
<i>Merchant Portal</i>	5
<i>Digital Key – Public/Private Key pair</i>	5
<i>Wallet</i>	5
<i>Signature</i>	5
<i>Invoice</i>	5
<i>Distribution</i>	6
<i>Promise To Pay Protocol</i>	6
<i>Pre-signed links</i>	6
<i>Offers</i>	6
<i>Recurring Payments</i>	7
<i>Currencies Supported</i>	7
<b>Integration Options</b>	<b>8</b>
<i>Option 1: Full Service Merchant Implementation</i>	9
Tasks	9
How it works	9
<i>Option 2: No code Option</i>	11
Tasks	11
How it works	11
<b>Tutorials</b>	<b>12</b>
<i>Signup and setup Merchant profile</i>	12
<i>Add 'Pay with Dropp' button</i>	14
<i>Callback Service and Posting payment request to Dropp</i>	16
<i>Add 'Pay with Dropp' button with pre-signed link for no-code option</i>	17
<i>Testing in a sandbox environment</i>	18
Turn on Test mode in Merchant Portal	18
Configure your service to use the sandbox environment	19
Configure your wallet to use the sandbox environment	19
<b>Help</b>	<b>21</b>

# Start Here

## Create Profile

First, you need to sign up for a merchant account and create your merchant profile. Creating a merchant account is free. If you haven't done so already, head over to the Merchant Portal at <https://merchant.portal.dropp.cc/> to sign up.

Creating a profile is quick and simple. All you need is the basic information about your business. Also, all merchants must go through the KYC process and this usually takes 1-2 business days. So, let's get the process going first. [Sign up and set up Merchant profile](#) tutorial walks you through the process.

While that is in progress, come back here to integrate your online storefront with Dropp payment platform.

If you are new to Dropp, you can learn about the Dropp payment platform at <https://dropp.cc/>.

# Navigating this Guide

This guide is organized into the sections shown. Feel free to jump around & refer to relevant sections.

## Concepts

Key concepts to be aware of.

## Tutorials

How-to guides to get you going.

## Integration Options

Ways to integrate and start accepting Dropp payments.

## Help

How to get help if you are stuck.

# Concepts

## Merchant Portal

Merchant Portal is your dashboard. You can manage and view all aspects of your merchant account from the portal.

Link to Merchant Portal is: <https://merchant.portal.dropp.cc/>

## Digital Key – Public/Private Key pair

The keys are part of a public-key cryptography system used to perform secure digital transactions. These keys are used to digitally sign the transactions with Dropp. You need to keep the private key private. Dropp doesn't maintain your private key.

## Wallet

Wallet is a mobile app with a supplementary browser extension available for desktop. Each consumer will need a Dropp wallet. They can download it from the mobile App store or Play store. They can fund their wallet using a bank account or credit card for fiat currencies or digital transfers for the supported digital currencies such as HBAR and USDC. The wallet securely stores payment information and enables the consumers to pay for services using Dropp.

## Signature

Payment transaction details are digitally signed using Private Key to prove authenticity and to protect its integrity. Signature and the signed data are verified using the Public Key by the receiving party.

## Invoice

Invoice data contains details such as product or service reference, price and the currency. Unlike traditional invoice, Dropp's invoice data does not contain customer identifying information. Dropp is designed to provide complete privacy for the consumers.

## Distribution

If you need to split the revenue with other parties such as your affiliates and partners, you'd do so by including the distribution details along with other invoice data. All you need is the account ID and the amount for each party.

## Promise To Pay Protocol

Promise To Pay is Dropp's patent pending message protocol that encapsulates invoice data of the payment transaction signed by the merchant and their customer. Also called P2P in short, Promise To Pay, utilizes ED25519 public key infrastructure to sign the digital transactions. While it helps to understand the P2P protocol, a lot of the functionality has been abstracted out with our friendly SDKs. So, you don't have to work directly with the internals of the protocol.

## Pre-signed links

This is a no-code option to accept payment with Dropp, best suited for donations and tipping. See [Integration Options Option 2: No code Option](#) for details.

## Offers

Offer discounts to your customers. You can create offers from the Merchant Portal and you can configure

- Target customers: existing customers, new or all customers
- Discount: fixed price discount (e.g. \$2 off) or a percentage (10% off)
- Expiration date

Once you have created an offer, it will be visible to your customers in their wallet app/extension.

## Recurring Payments

Recurring payments support subscription and micro subscription models. In addition to traditional subscription models, you can use Dropp's recurring payments to offer unbundled media consumption and more.

For example, instead of charging customers for a book or a video upfront, you can charge them per page read or per 5 minutes of video watched. This opens additional possibilities and opportunities for you and your customers. Customers can now freely try without having to pay upfront for something they may not enjoy – they pay only for what they consumed.

## Currencies Supported

Dropp platform currently supports USD and AED fiat currencies as well as USDC and HBAR digital currencies. You can accept payments in any or all these currencies. Similarly, your customers can pay in these currencies. We regularly introduce additional currencies into the platform. If currency of your choice isn't supported yet, please get in touch with us.

# Integration Options

There are 2 ways to integrate and start accepting payments using Dropp. Each caters to a different use case. You can use either or both options depending on your varying needs.

	Option 1	Option 2
Best Suited Use Cases	<p>Pay Per Use Paywalls for media, blogs and such.</p> <p>Subscriptions and recurring payments.</p>	<p>Donations, Tipping.</p> <p>Cases where no post-payment processing is needed to fulfil goods or services.</p>
Example	<p>Charge customers to read just one article instead of having them sign up for a monthly subscription.</p> <p>Pay per page of the book read, or for every 5 minutes of the video watched.</p>	<p>Accept charitable donations.*</p> <p>Accept fan tipping.</p>
Merchant Requirements	<p>Implement service to perform server-side processing using our SDK.</p>	<p>This is a no-code option. No service implementation needed - but supports only cases like those noted above.</p>

\* Accepting charitable donations using Dropp:  
Dropp is designed to provide complete privacy for the consumers. Therefore, customer details are not available from Dropp if you need to provide your customers with a donation receipt for tax deduction purposes.



## Option 1: Full Service Merchant Implementation

In this option, you'd need to setup a backend service with a callback URL, that would then call Dropp payment services to post payments. This is our recommended option.

### Tasks

1. [Sign up and set up Merchant profile](#)
2. [Add 'Pay with Dropp' button](#)
3. [Callback Service and Posting payment request to Dropp](#)
4. Enable the content/service for the customer or initiate product dispatch.

### How it works

Add a 'Pay with Dropp' button to your product or service listing in your online storefront. This is a one-time set up.

↓  
Customer clicks on the 'Pay with Dropp' button.

↓  
Automatically opens the customer wallet app.

↓  
Customer verifies the purchase information and clicks the 'Pay' button in their wallet app.

↓  
That invokes merchant's (your) service for the callback URL.

↓  
In your service, you:

Verify purchase information (aka Invoice) received from customer wallet.

↓  
Prepare a request to Dropp with your (Merchant) information and the purchase information (Invoice) received from the customer wallet.

↓  
Digitally sign the request.

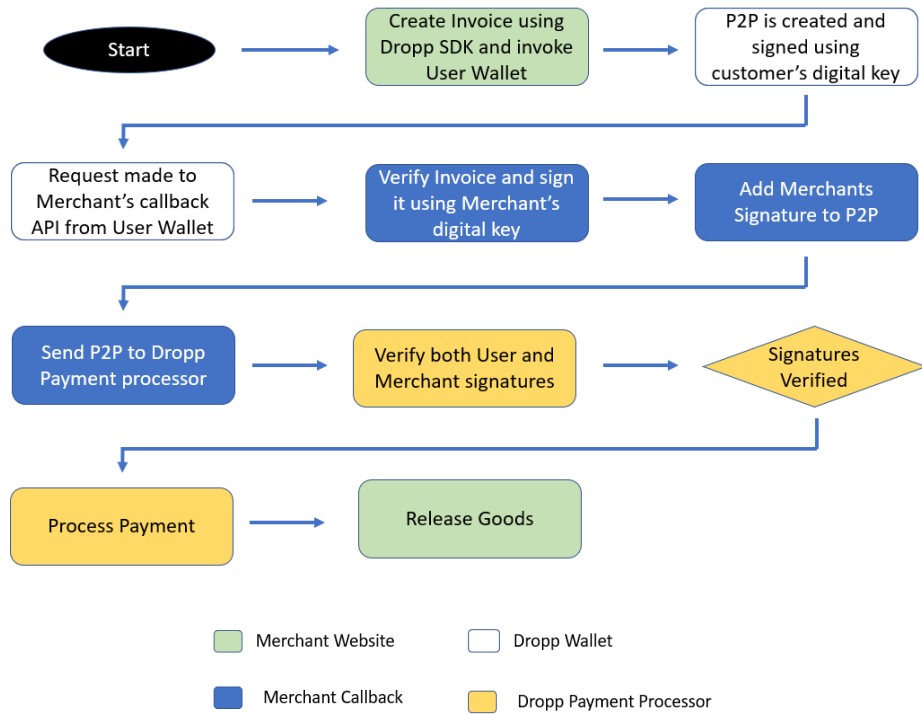
↓  
Call Dropp Payment API and send this request.

↓  
Successful return from Dropp Payment API indicates that the payment went through and the customer has been charged for Invoice amount.

↓  
Enable the service for the customer or initiate product dispatch.

↓  
This completes the purchase.

### Pictorial representation of the flow



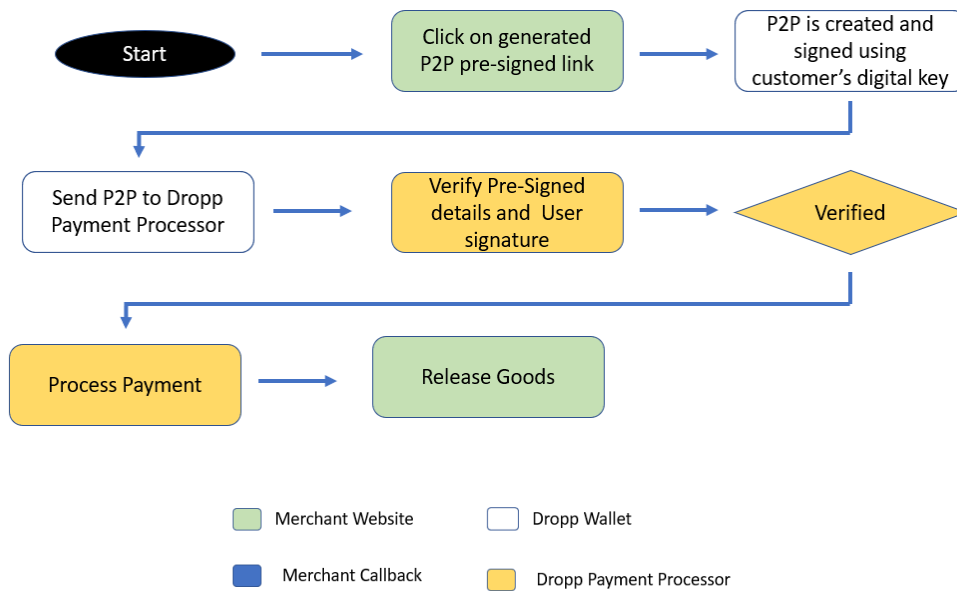
## Option 2: No code Option

No code option is for merchants without a backend server. From the merchant portal, you'd generate a pre-signed link per product or service you offer. Add this link to your online storefront or email campaign. When your customers click the link, their wallet app will open. They will then initiate the payment through the wallet app and interact directly with Dropp payment services.

### Tasks

1. [Sign up and set up Merchant profile](#)
2. [Add 'Pay with Dropp' button with pre-signed link for no-code option](#)

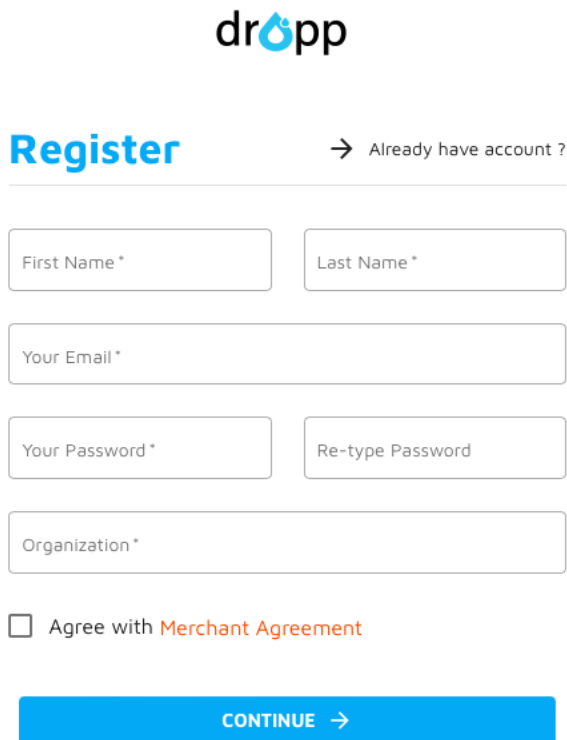
### How it works



# Tutorials

## Signup and setup Merchant profile

### Signup



The screenshot shows the 'Register' form for Dropp. At the top left is the 'dropp' logo. To the right of the logo is the text 'Register' in blue, followed by a link '→ Already have account?'. Below this is a horizontal line. The form consists of several input fields: 'First Name \*' and 'Last Name \*' (two separate boxes), 'Your Email \*' (one wide box), 'Your Password \*' and 'Re-type Password' (two separate boxes), and 'Organization \*' (one wide box). Below the fields is a checkbox labeled 'Agree with Merchant Agreement'. At the bottom is a blue button with the text 'CONTINUE →'.

Visit Dropp Merchant Portal at <https://merchant.portal.dropp.cc/> and signup for an account.

After initial sign up, you will receive an email to verify your account.

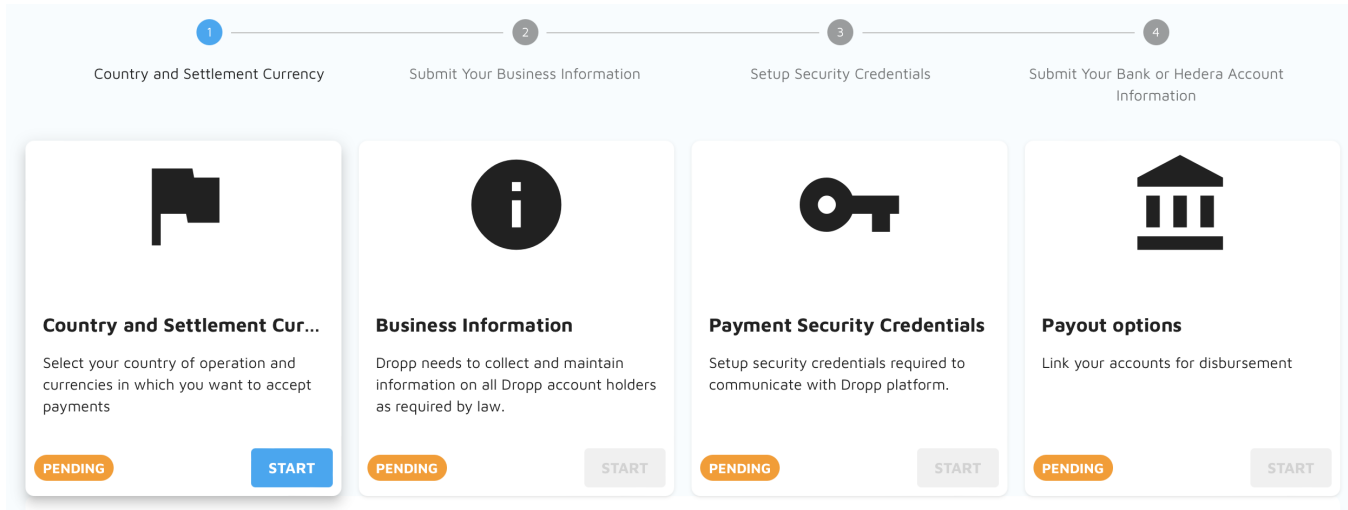
Click the link in the email to verify and activate your Dropp Merchant Account.

Once activated, you can login to the Merchant Portal and set up your merchant profile.

## Setup Profile

Make your selection and provide required information about your business in each of these 4 simple steps.

Each step is enabled as the previous one completes.



### Country and Settlement Currency

Which currencies can your customers pay with?  
Dropp supports USD and AED fiat currencies & HBAR and USDC digital currencies. We continue to add support for more currencies into the platform.

### Business Information

Provide your key business information, location, Government issued business registration ID, website and contact information.

### Payment Security Credentials

Dropp will generate a unique public/private key pair for you. Keep them secure. You need the private key to sign all payment communications. Private is confidential and Dropp does not keep a copy.

### Payout Options

Link your payout bank account. Payouts are made daily with support for real time payments coming soon.

## Add 'Pay with Dropp' button

You can also generate this code from Merchant Portal's Payment Link section and copy/paste into your site or email campaign.

Add the following JS library on your web page.

```
<script src="https://merchant.portal.dropp.cc/dropp-sdk/dropp.min.js" type="text/javascript"></script>
```

Add code like the below to your listing to add a Pay with Dropp button. Add this to each of your items.

```
<a class="dropp-payment" data-amount="[selling price]" data-callback-url="[merchant callback Url]" data-currency="[price currency]" data-description="[product description]" data-merchant-id="[your account id]" data-reference="[your internal unique product reference. e.g. sku]" data-thumbnail="[url for the product thumbnail image]" data-title="[product title]" data-type="[product type for your reference]">
  <span>Pay with </span>
</a>
```

It doesn't have to be an anchor tag (<a>). It can be a <div> or another element as well. The full HTML code for a sample page listing a single product will look like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Awesome Store with Dropp Payment</title>
  <script src="https://merchant.portal.dropp.cc/dropp-sdk/dropp.min.js" type="text/javascript"></script>
</head>
<body>


<H3>Floating Man $5.25</H3>
<a class="dropp-payment" data-amount="5.25" data-callback-url="https://droppmerchant.example.com/sample/callback" data-currency="USD" data-description="Man floating on the sea during sunset" data-merchant-id="0.0.12345" data-reference="M8409762" data-thumbnail="https://droppmerchant.example.com/media/man.png" data-title="Floating Man" data-type="Photograph">
  <span>
    Pay with
    
  </span>
</a>
</body>
</html>
```

Notice the various data fields in the code above. The data-callback-url field is the hook back to your service. Customer wallet calls back the URL provided in data-callback-url once the customer clicks Pay in their wallet/extension. To test the callback locally, you can set the callback URL to service running locally - <http://localhost:.../>

The above HTML code will render thus:



**Floating Man \$5.25**

Pay with 

## Callback Service and Posting payment request to Dropp

We encourage creating your own, but a sample working code is available.

NOTE: This code is provided as-is and is **not** battle tested. We provide this for your reference.

```
@GetMapping("/sample/callback")
public String callbackFromWallet(@RequestParam String p2p) throws DroppClientException {

    // P2P Object from JSON
    // Use the PromiseToPayData object to
    // validate invoice data and any other relevant details.
    PromiseToPayData p2pData = getP2PData(p2p);

    // Perform any necessary validation.
    // Simple, sample validation included below.
    // Add more validations as needed for your case.
    validate(p2pData);

    // Create Dropp Payment Request, Sign and Submit for payment.
    PaymentResponse response = signAndSubmit(p2pData);

    if(response.getResponseCode() == SUCCESS_CODE){
        saveToDbAndEnableServiceForPayer(p2pData);
    }else{
        notifyPaymentFailure();
    }
    return response.toString();
}

private PromiseToPayData getP2PData(String p2pJson) {
    if (StringUtils.isBlank(p2pJson)) {
        return null;
    }
    try {
        return objectMapper.readValue(p2pJson, PromiseToPayData.class);
    } catch (JsonProcessingException e) {
        log.error("Error parsing p2p Json", e);
    }
    return null;
}

private PaymentResponse signAndSubmit(PromiseToPayData p2pData) throws DroppClientException {

    // #1. Create Dropp Payment Request, Sign.
    DroppPaymentRequest paymentRequest =
        DroppPaymentRequest.newDroppPaymentRequest(DroppClient.instance(droppEnvironment))
            .withP2p(p2pData)
            .signByMerchant(privateKey);

    // paymentRequest.getP2p() contains merchant signed P2P if you want to inspect.

    // #2. Submit for payment
    PaymentResponse response = paymentRequest.submitPayment();

    // response.responseCode of 0 denotes success

    return response;
}
```

Your callback service is called via HTTP GET Method, with the query parameter named p2p as shown above in the sample code.



## Add 'Pay with Dropp' button with pre-signed link for no-code option

You can generate the pre-signed links from Merchant Portal's Payment Link > Pre-signed Payment Link section.

Fill in the information and click CREATE. You can then copy/page the generated link or the code.

When the payment is successful, users are directed to 'Success URL' specified. And if payments fail, they are directed to 'Failure URL'. The success and failure URLs would typically be a static page on your website.

### Generate Pre-Signed Link With Dropp

Product Title \*

Product Description \*

Product Type \*

Amount  Range

Currency \*  
USD

Product Image URL \*

Success URL

Success Message

Failure URL

Purchase Expiration

Reference \*

Purchase URL

Share URL

Live Account  Test Account

**CANCEL** **CREATE**

### Generated Links

**COPY PAYMENT LINK**

**COPY PAYMENT LINK CODE**

## Testing in a sandbox environment

You can use our sandbox environment to test your setup end-to-end before going live or anytime you need testing.

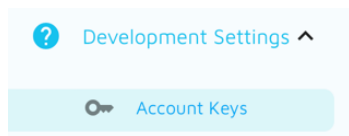
You need to do 3 things to get this going.

1. Turn on Test mode in Merchant Portal
2. Configure your callback backend service to connect to our sandbox environment
3. Configure wallet/browser extension to connect to our sandbox environment. Use this to simulate your customer interaction.

When you are done testing, switch back #1 and 2 to the main/production environment. (You can discard #3 or switch that back to the main/production environment as well).

## Turn on Test mode in Merchant Portal

A test merchant account is automatically created for you when you signed up as a merchant. You can find this in Merchant Portal with this toggle available on the left navigation menu. Toggle it to switch between main vs. test data.



Development Settings > Account Keys section in the portal will provide you with the account id and the keys. When in test mode, these will reflect the id and keys of the test account. Use this account id and keys (Promise-To-Pay Signing Keys) during your test to sign your data as usual.

This test account is unique to you and is not shared across merchants.

## Configure your service to use the sandbox environment

In the callback service code shared earlier in the document, you might have noticed this code to create an instance of `DropClient`. Pass in the environment you'd like to connect to. To connect to the sandbox, pass `Environment.SANDBOX`. For the live system, this would be `Environment.PROD`.

```
// #1. Create Dropp Payment Request, Sign.  
DropPaymentRequest paymentRequest =  
    DropPaymentRequest.newDropPaymentRequest(DroppClient.instance(droppEnvironment))  
        .withP2p(p2pData)  
        .signByMerchant(privateKey);
```

Make sure you connect to the correct environment wherever you are creating an instance of `DropClient` in your code. You can also create just one instance of `DropClient` and reuse.

## Configure your wallet to use the sandbox environment

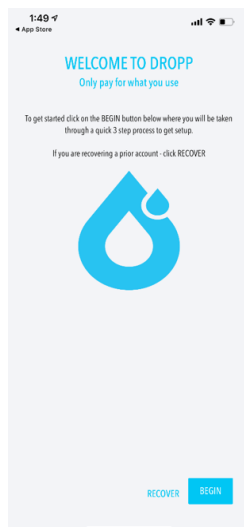
1. First download and install mobile wallet as well as browser extension.



Scan or click the QR code to download the wallet.

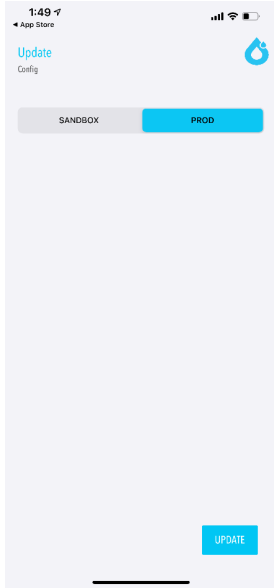
On a mobile phone, this will take you to the device's respective app store.

On the desktop, this will take you to the browser extension store.



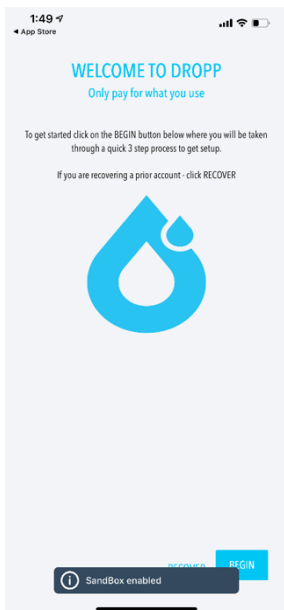
2. Open the wallet.

Press and hold the Dropp icon in the middle for 6-10 seconds.



3. This will present the environment selection screen.

Select the Sandbox tab to configure the wallet for testing.



4. When prompted, Quit and reopen the wallet.

You are now setup for Sandbox

5. Click BEGIN to create a test Dropp Wallet account and follow the instructions on the wallet.

# Help

We are here if you need help or have questions. We can have a dedicated Engineering staff guide you through the onboarding every step of the way.



<https://dropp.cc>



[info@dropp.cc](mailto:info@dropp.cc)



<https://twitter.com/droppcc>

**We can't wait to see  
What you develop with Dropp.**

