Docs > Integrations > Point of sale systems

POS Foundation

POS Foundation provides a proof-of-concept open source framework for building point-of-sale (POS) checkout interfaces that serve BigCommerce merchants with physical locations. It provides a template for custom POS solutions that use secure, pre-certified EMV card readers. POS Foundation's default integration with Stripe Terminal can significantly accelerate development time.

POS Foundation scaffolds manual connector apps that use store-level API accounts. You do not need to create an app profile or use ngrok. The app will not appear in the store control panel and is only accessible to computers running on the same local network as the app's server. In contrast to Subscription Foundation-derived apps, POS Foundation apps do not create their own dedicated sales channels. If you're interested in contributing to this package, see Contributing.

Software requirements

- Node.js 18.0.0+
- The npm package manager
- A MongoDB database instance; this guide uses MongoDB Cloud

Hardware requirements

- A Stripe Terminal-enabled card reader
- A card reader-compatible device to run the POS system

Configure accounts

> **Account configuration requirements**
>
> We recommend that you use a sandbox store that has the same multi-storefront status as the production store. For example, if you're developing for a multi-storefront-enabled merchant store, use a multi-storefront sandbox. For information on configuring multi-storefront, see Multi-Storefront.

To configure your accounts, complete the following steps:

1. To develop and test safely, you need a BigCommerce sandbox store. If you don't have one, Create a Sandbox Store.
2. In the control panel of your sandbox store, create a store-level API account and add the following OAuth scopes:

UI Name	Permission	Parameter
Customers	modify	`store_v2_customers`
Information & settings	read-only	`store_v2_information_read_only`
Orders	modify	`store_v2_orders`
Create payments	create	`store_payments_access_token_create`

UI Name	Permission	Parameter
Get payment methods	read-only	`store_payments_methods_read`
Products	read-only	`store_v2_products_read_only`
Carts	modify	`store_cart`
Checkouts	modify	`store_v2_checkouts`
Channel Settings	modify	`store_channel_settings`

Record the API account credentials

- Record the **access token**, **client ID**, and **client secret** in a safe place. In a later step, you will update the .env file and specify these credentials as the values of the `BC_AUTH_TOKEN`, `BC_APP_CLIENT_ID`, and `BC_APP_SECRET` environment variables.
- In addition, make note of the **store hash**. It is the path parameter that immediately precedes `v3` in the `API PATH` included with your store API account. In a later step, you will specify the store hash as the value of the `BC_STORE_HASH` environment variable.

Fork and install the source repository

To fork the repository, complete the following steps:

1. Fork the point-of-sale-foundation repository (GitHub) to your GitHub account.
2. Clone the fork to your local development environment.
3. Navigate to the root directory of your cloned repository and install the default packages by running the following command:

Install packages

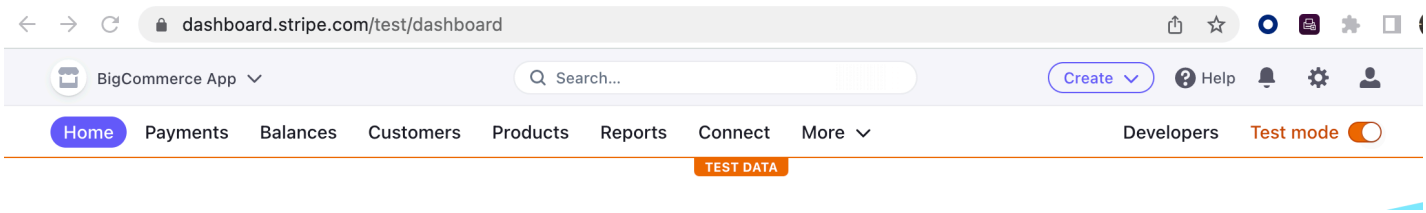
```
1  npm install
```

Configure Stripe

To develop the POS interface, you need to create and configure a Stripe account.

To configure the Stripe account to connect with your implementation, complete the following steps:

1. To create the Stripe account, sign up with Stripe.
2. Sign in to the Stripe Dashboard.
3. In the top right corner of your Stripe Dashboard, click **Test Mode** and verify that **Test mode** is selected.



> **Test mode**
>
> Simulate transactions using test mode to confirm your integration works correctly.

4. Select **API keys** in the left menu, then locate the **Standard keys** section of the page. In the **Secret key** table row, click **Reveal test key** and copy the string that appears.

> **Record the Stripe secret key**
>
> Record the **secret key** and keep it in a safe location. In a later step, you will update the .env file and specify the secret key as the value of the `STRIPE_SECRET_KEY` environment variable.

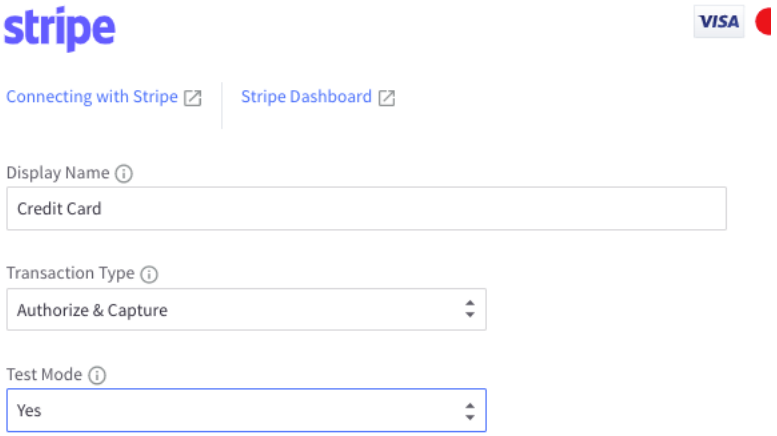
5. On your Stripe Dashboard, click **More > Terminal** to enable Stripe Terminal.
6. When prompted to configure Stripe Terminal, click **Get Started**.
7. Click **Locations** in the left menu, then click the **+ New** button at the top right of the **Locations** section.
8. In the **Create location** dialog, enter a **Name** and an **Address**, then click **Save**.
9. To add a reader to the new location, click on the row that lists the location, then click the **+ New** button at the top right of the **Readers** section.
10. In the **Register reader** dialog, enter the **Registration code** and a **Reader label**, then click **Save**.

Configure the BigCommerce store

After you successfully configure Stripe, configure your BigCommerce sandbox store in the store control panel.

To configure the store to make purchases using the POS interface, complete the following steps in the BigCommerce store control panel:

1. Configure a shipping zone with a **Pickup in Store** shipping method; for directions, see our support article on Setting up an "In-Store Pickup" Shipping Method.
2. Ensure that your store is using **Optimized One-Page Checkout**; for directions, see our support article on Switching to Optimized One-Page Checkout. Learn more about Optimized One-Page Checkout Settings.
3. Connect the sandbox store with the Stripe account you configured in the previous section; for directions, see our support article on Setting up Stripe.
4. In the Stripe settings section, locate the **Test Mode** dropdown and select **Yes**.



Create and configure the database

POS Foundation uses MongoDB as a database engine. If you want to use a different data store, you'll need to modify the Prisma configuration, migration, and seed files. To learn more, see Prisma's list of database connectors.

To configure your POS implementation to use MongoDB, complete the following steps:

1. On your computer, navigate to the fork of the project repository, and locate the `/prisma` directory.
2. In the `/prisma/schema.prisma` settings file, set `provider` to `mongodb`.

schema.prisma

```
generator client {
  provider        = "prisma-client-js"
  previewFeatures = ["mongoDb"]
}

datasource db {
  provider = "mongodb"
  url      =  env("DATABASE_URL")
}

model Role {
  id        String   @id @default(auto()) @map("_id") @db.ObjectId
  name      String
  Employee Employee[]
}
```

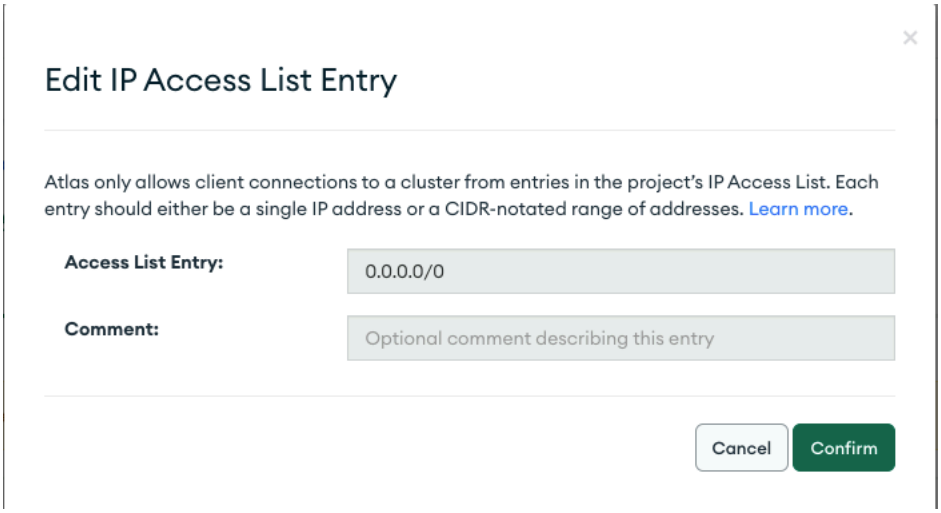
Configure MongoDB Cloud

After you configure your application to use MongoDB, connect to a MongoDB instance. You can use MongoDB Cloud to get your implementation up and running quickly. As an added benefit, MongoDB Cloud makes your data collections accessible from anywhere.

To configure MongoDB Cloud and generate a connection URL, complete the following steps:

1. Navigate to MongoDB Cloud and sign in or sign up for an account.
2. From the left menu, select **Database Access**, then click **Add New Database User**.
   a. For **Authentication Method**, select **Password**.
   b. Enter a username and password. Leave all other information unchanged.
   c. To save, click **Add User**.
3. From the left menu, select **Network Access**, then click **Add IP Address**.

a. For **Access List Entry**, enter an IP address. Using `0.0.0.0/0` allows you to connect from anywhere, but is not secure in production. To learn more, consult MongoDB's IP access documentation.



b. To save, click **Confirm**.

4. From the left menu, select **Database**.

a. Locate your running cluster, then click **Connect**.

b. In the dialog that opens, click **Connect your application**.

c. Copy the connection string that appears, and replace `<username>` and `<password>` with the values you specified in step 2b. Add **myFirstDatabase** to the connection string so that it resembles the following example:

Example MongoDB Cloud connection string

```
mongodb+srv://<username>:<password>@cluster0.sdfdfg65.mongodb.net/myFirstDatabase?retryWrites=true&w=majority
```

```
# Prisma DB Client
DATABASE_URL=mongodb+srv://username:password@cluster0.jfohhb8.mongodb.net/
myFirstDatabase?retryWrites=true&w=majority
```

Record the connection string

Record the MongoDB Cloud connection string and keep it in a safe location. In a later step, you will update the .env file and specify the connection string as the value of the `DATABASE_URL` environment variable.

Declare environment variables

After you create and configure the database, you have all the information you need to add environment variables to your project.

To declare environment variables, complete the following steps:

1. Create a `.env` file in the root directory of your project.
2. Copy the contents of `.env.sample` to `.env` with the following command:

Copy .env.sample contents

```
cp .env.sample .env
```

3. Open the `.env` file you just created and update the following environment variables:

Environment variable	Description	Reference location
`BC_STORE_HASH`	The unique string that identifies your store	Configure accounts
`BC_AUTH_TOKEN`	The store API account's access token	Configure accounts
`BC_APP_CLIENT_ID`	The store API account's client ID	Configure accounts
`BC_APP_SECRET`	The store API account's client secret	Configure accounts
`STRIPE_SECRET_KEY`	The Stripe account's secret key	Configure Stripe
`DATABASE_URL`	The database connection string	Create and configure the database
`BC_CHANNEL_ID`	Modify this value if you create a dedicated channel for the POS system	Create a channel (API Reference)
`BC_GQL_URL`	The URL of the store's GraphQL Storefront API endpoint	See notes in the .env.sample file (GitHub)

Run migration and start the server

After you add environment variables to your project, run the migration script that creates the initial collections and documents your MongoDB instance needs to run the POS system.

To run the migration script and seed the database, complete the following steps:

1. Run the Prisma migration script that is pre-defined in `/prisma/migrations/*` with the following command:

Create the database

```
npx prisma db push
```

2. Seed your database with pre-configured collections and documents using the following command:

Seed the database

```
npm run seed
```

⚠️

Important

If you miss the preceding steps, you will not have any data to work with.

3. Generate a new Prisma client with the following command:

Generate the Prisma client

```
npx prisma generate
```

> Successfully generating the Prisma client relies on the provider settings you configured in the preceding section on [creating and configuring the database](#create- and-configure-the-database).

4. Access the database to verify you have created the collections and documents correctly. You can use the following command, or connect with a GUI, such as Compass.

Verify migration and seed

```
npx prisma studio
```

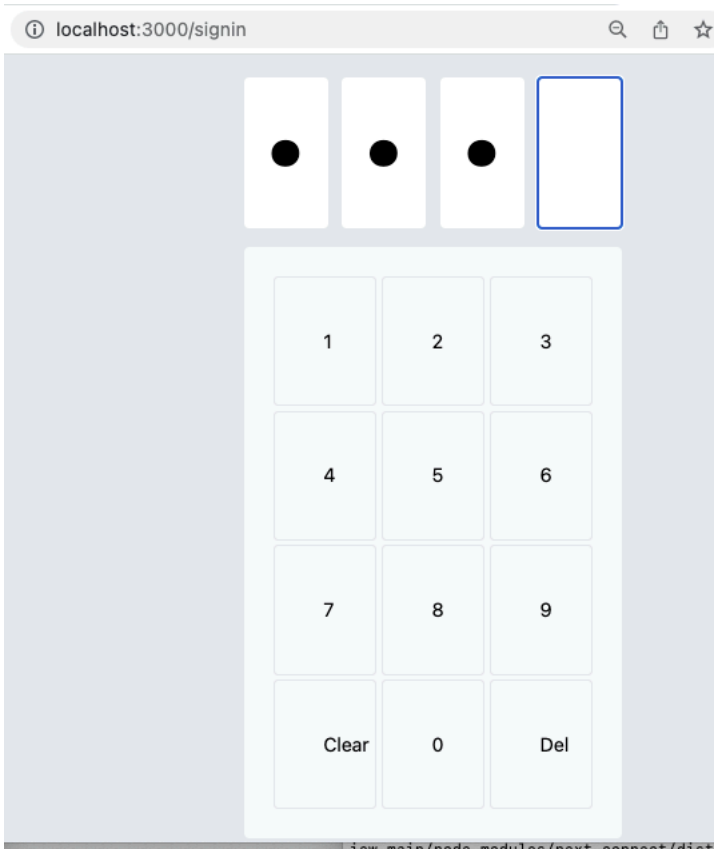
5. To start the server that runs the POS system, run the following command:

Start the server

```
npm run dev
```

When the preceding script runs successfully, the app is running locally.

6. Sign in to the app as an admin at `http://localhost:3000/signin`. The default admin PIN in the seed data is `1234`.



7. After you access Prisma Studio, navigate to the **Settings** view to save your store address, which the app uses for checkout tax calculations. If you fail to supply the store address, you will receive an unhandled runtime error.

Connect the POS to an EMV card reader

After you start the server and sign in to the app, connect the device running the POS with one of your Stripe account's pre-certified EMV card readers.

To connect the POS with a card reader, complete the following steps:

1. To set up one or more pre-certified Stripe Terminal card readers, follow Stripe's directions to Set up your reader.
2. Locate the tablet, phone, computer, or other device you want to use to take payment at checkout. Connect it to the local network that's running your application and sign in at `http://localhost:3000/signin`.
3. To scan the local network for Stripe Terminal card readers connected with your account, navigate to the main register view. Click or press the **Wifi** icon.
4. In the list of card readers that appears, locate the reader you want to connect. Click or press **Connect**.

Troubleshooting

Issue: Environment variable not found when running the database migration

This error can occur when the `/prisma/schema.prisma` file contains the incorrect provider, or the `.env` file contains an incorrect `DATABASE_URL`. Verify that the information in both files is correct.

Issue: A "no available servers" server selection timeout occurs when running the database migration

Change your MongoDB Cloud IP address to `0.0.0.0/0`, or consult MongoDB's IP access documentation for alternatives.

Contributing

Want to help expand this foundation? We'd love to hear from you. Submit a pull request, and consider joining the BigCommerce Developer Community Slack workspace to connect with us.

Resources

- Point of Sale Foundation README (GitHub)
- Introducing the Point of Sale Foundation (Dev Blog)
- Create a Sandbox Store
- Create a Store-level API Account
- Connecting with Stripe
- Setting up an "In-Store Pickup" Shipping Method
- Switching to Optimized One-Page Checkout

External resources

- Stripe Terminal
- Set up Stripe Terminal reader
- MongoDB Cloud

Join our community of thousands of developers

Enter your email

Join #BigCommerceDevs Slack

Join thousands of other #BigCommerceDevs that are building on BigCommerce. We will also keep you updated with our monthly BigCommerce Developers newsletter. You may opt out at any time.

Need help? Join our BigCommerceDevs Slack

Find endpoints quickly in the API reference

Stay up to date with Release notes

Visit BigCommerce home

Everything you need for growth, all from one modern platform.