CHAPA MERCHANT SECURITY HANDBOOK

2024

Connecting Ethiopia to the Global Economy

Contact Us

info@chapa.co

www.chapa.co

Welcome to the Chapa Merchant Handbook

Your comprehensive guide to securely and efficiently managing transactions with Chapa. This handbook provides clear, step-by-step instructions on using key features such as fraud prevention, transaction approvals, API integrations, and security settings.

Whether you're configuring Radar for fraud detection, managing API keys, enabling OTP approvals, or setting up IP address whitelisting, this guide is here to help you navigate Chapa’s tools with ease.

Our aim is to ensure a seamless, secure payment experience for your business.

Getting Started

URL Approval / Server Approval

- Ensures that a transfer request was initiated by you.
- Uses a two-step verification process:
    1. Create an Approval URL
    2. Add URL to the Dashboard

1. Create an Approval URL

- The Approval URL is a `POST` endpoint that validates transfer details.
- Chapa sends payment data and a hash (HMAC SHA256) generated with your `approval_secret` to this URL.
- Your server must verify the request and respond accordingly:

Response Codes & Meaning:

Response code	Meaning	Status
200	Transfer approved	pending
400	Transfer rejected	reverted

Example Request

Headers:

```
Unset
{ "Chapa-Signature": "9f6a8c3b24d7e1f5a4c2b7a6e8f1c3d4" }
```

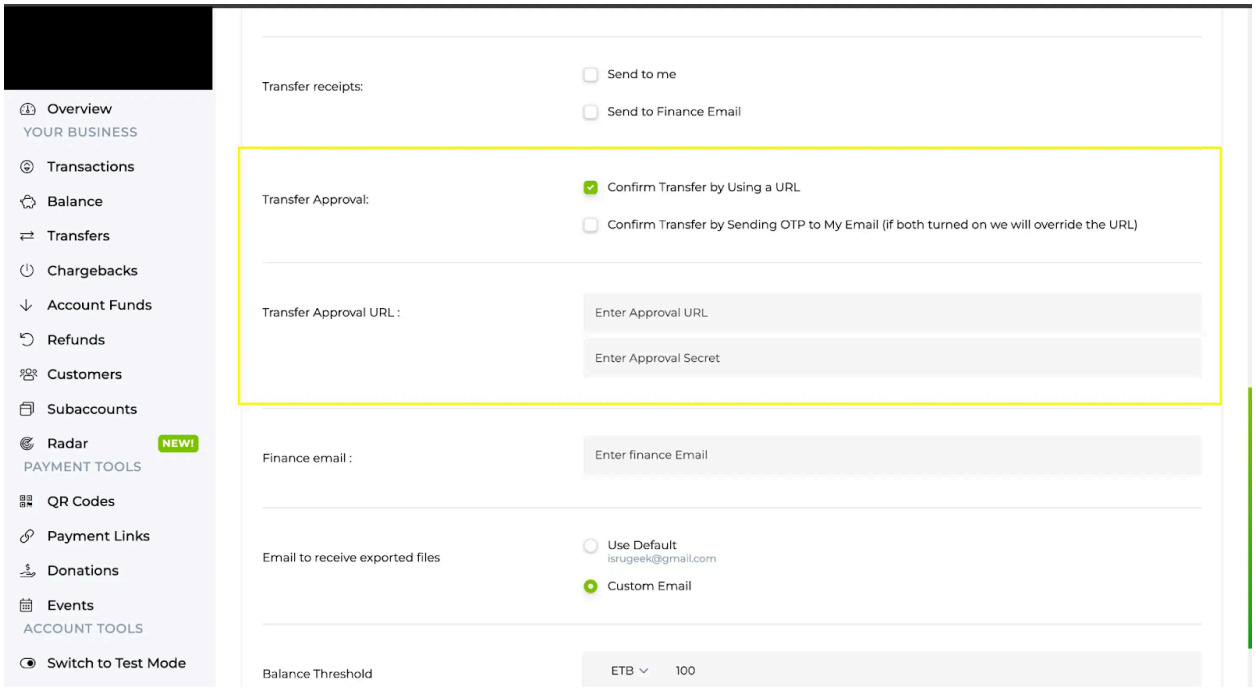
Body:

```
Unset
{
  "amount": "2000.00",
  "reference": "MYMER3434989",
  "bank": "telebirr",
  "account_name": "Customer Name",
  "account_number": "25190000000"
}
```

2. Add URL to dashboard

Once you've implemented your **Approval URL**, follow these steps to add it to Chapa:

1. Log in to your **Chapa Dashboard**.
2. Navigate to **Settings** → **Account Settings**.
3. Locate the **Transfer Approval** section.
4. Enter your **Approval URL** in the provided field.
5. Save your changes.

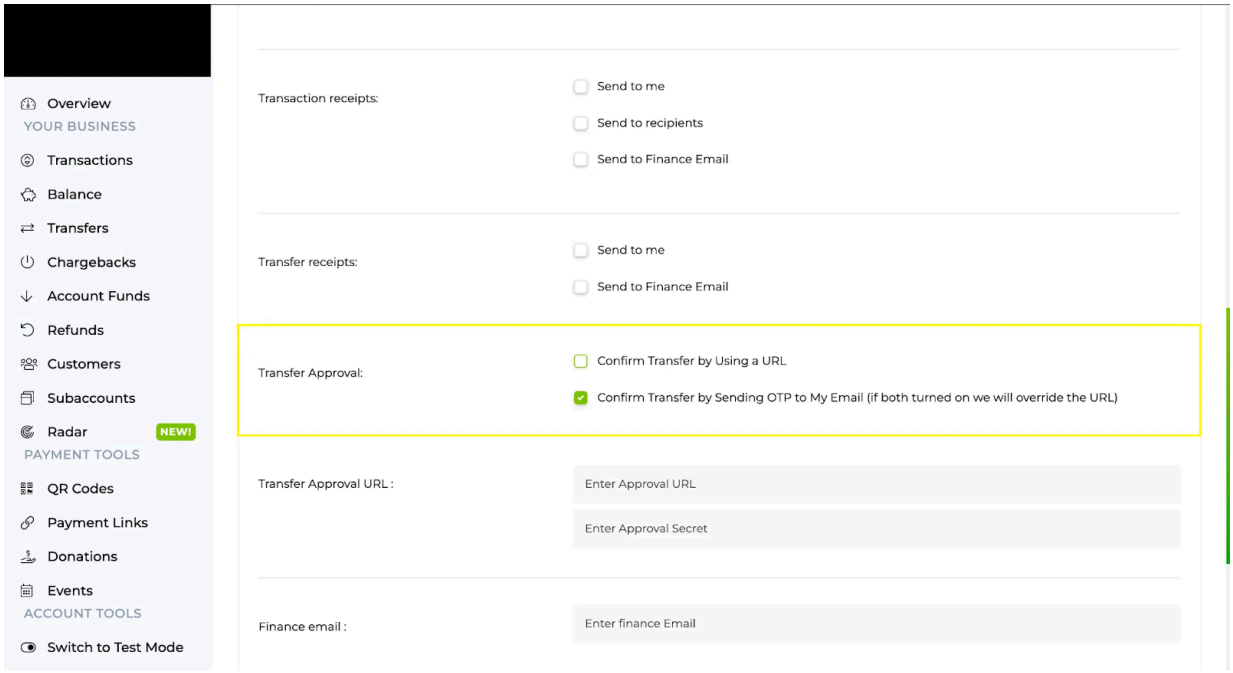

Overview
YOUR BUSINESS
Transactions
Balance
Transfers
Chargebacks
Account Funds
Refunds
Customers
Subaccounts
Radar
NEW!
PAYMENT TOOLS
QR Codes
Payment Links
Donations
Events
ACCOUNT TOOLS
Switch to Test Mode
Transfer receipts:
Send to me
Send to Finance Email
Transfer Approval:
Confirm Transfer by Using a URL
Confirm Transfer by Sending OTP to My Email (if both turned on we will override the URL)
Transfer Approval URL :
Enter Approval URL
Enter Approval Secret
Finance email :
Enter finance Email
Email to receive exported files
Use Default
isrugeek@gmail.com
Custom Email
Balance Threshold
ETB
100

OTP Approval

- Enhance security by requiring a **One-Time Password (OTP)** confirmation before processing transfers.

How to Enable OTP Approval:

1. Log in to your **Chapa Dashboard**.
2. Go to **Settings → Account Settings**.
3. Find the **Transfer Approval** section.
4. Toggle **"Confirm transfer by sending OTP to my email."**
5. Click **Save**.

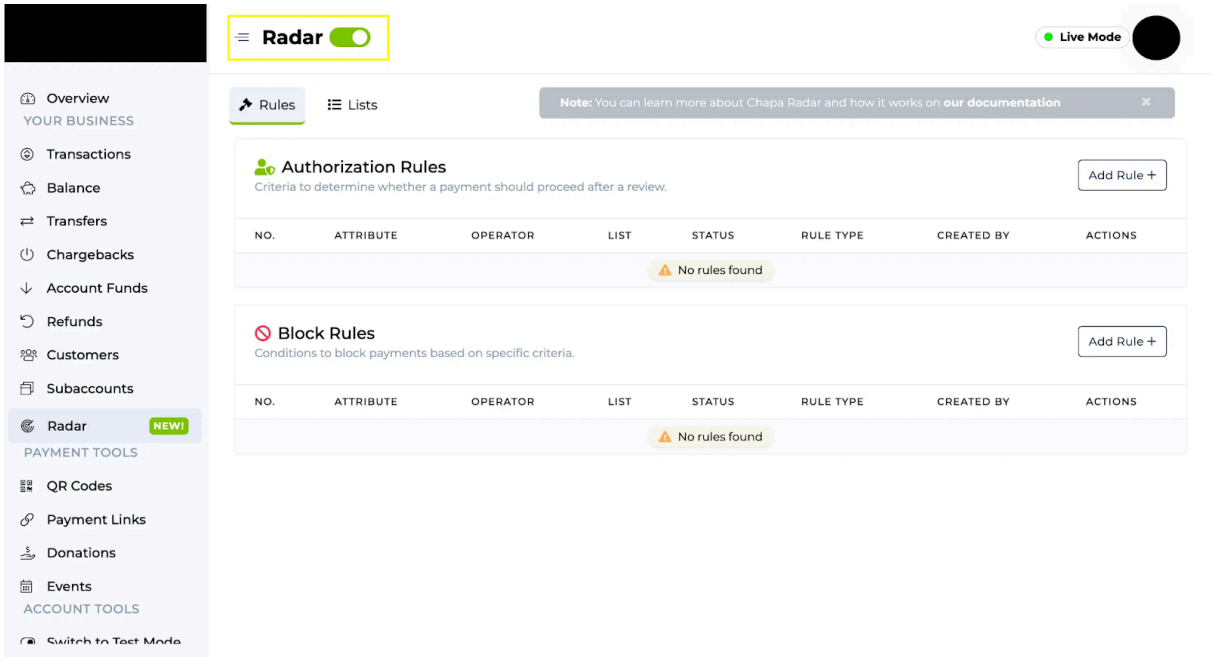


Now, transfers will require an OTP sent to your email for confirmation before processing.

Radar - Real-Time Fraud Protection Guide

Getting Started with Radar

1. Go to dashboard.chapa.co.
2. Sign Up or Sign In.
3. Click **Radar** from the side tab.
4. If not enabled, turn the toggle **on** to activate Radar.

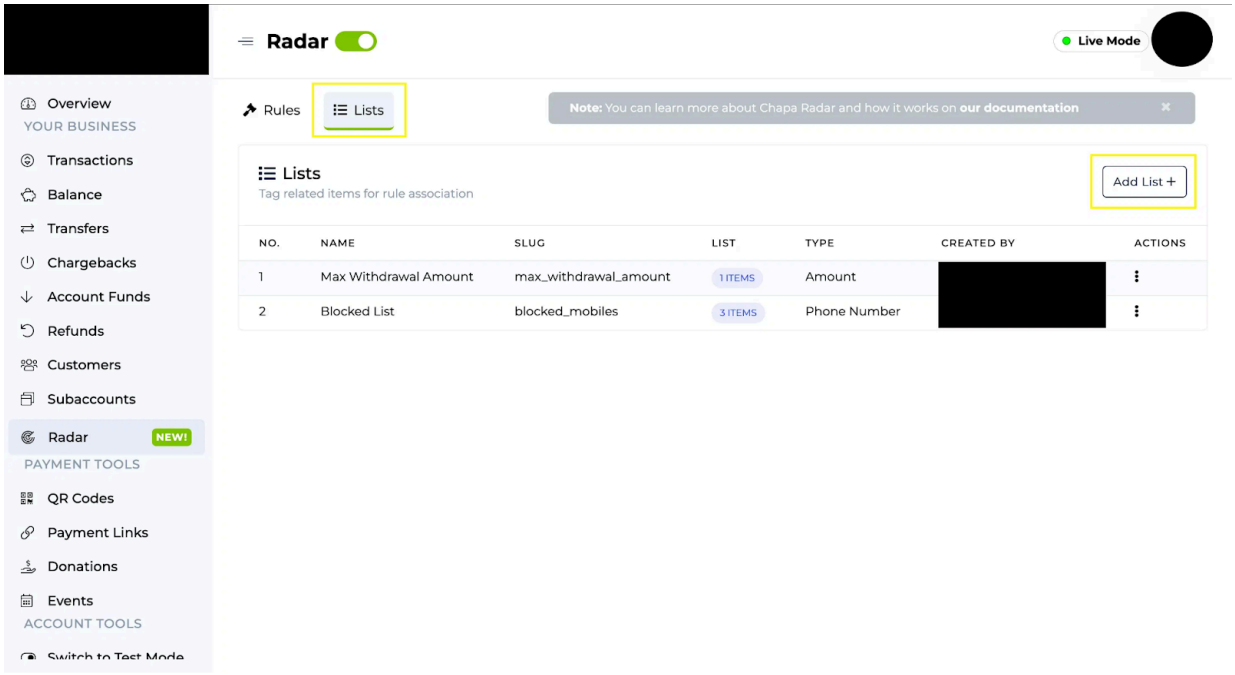


Lists

Create lists to **block, allow, or review** specific payments.

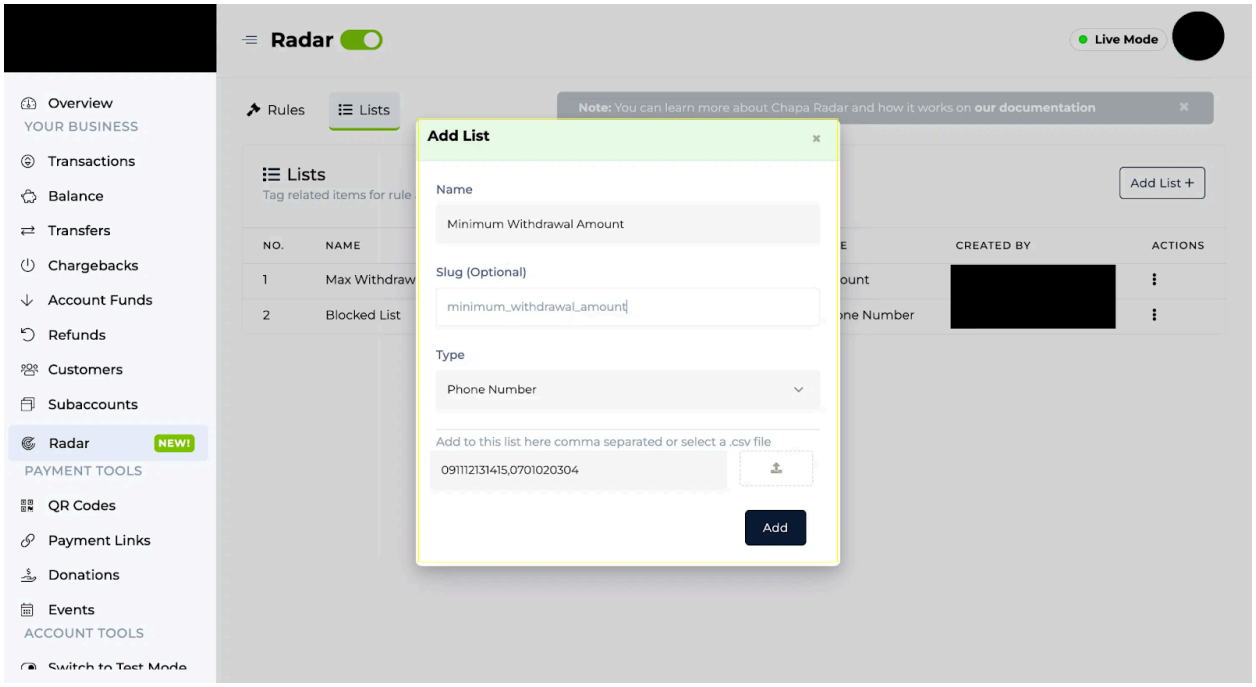
Common List Examples:

- **Blocked List** – Automatically deny payments from specific customers.
- **Maximum Withdrawal Amount** – Set a limit on withdrawals.



How to Add a List:

1. Click the **Lists** tab → **Add List**.
2. Fill in:
    - **Name** (required) – A unique, descriptive identifier.
    - **Slug** (optional) – A URL-friendly version of the name.
    - **Type** – Choose from:
        - Phone Number, Email, Account Number, IP Address, Amount, Card Number, Customer ID, Device ID, Location, or Name.
    - **List Items** – Enter data as a comma-separated list or upload a CSV file. Ensure formatting matches the selected data type.
3. Click **Add** – A confirmation message appears.



Rules

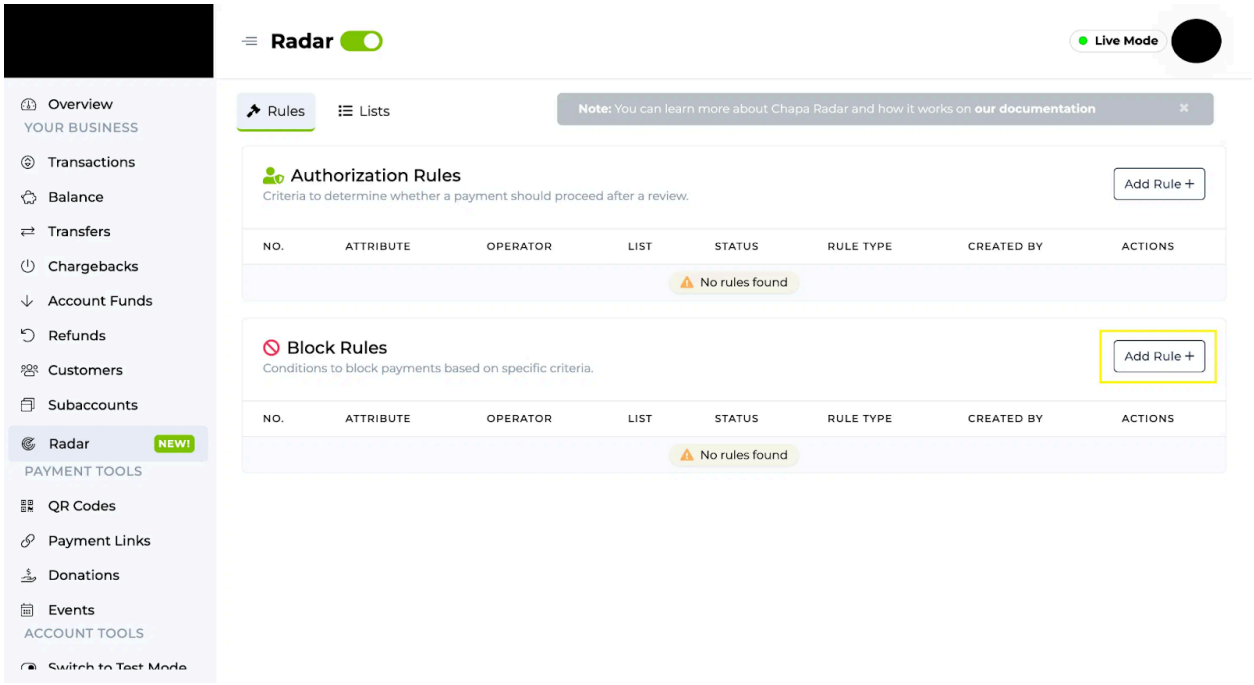
- Radar allows you to **create rules to block or authorize transactions** based on specific conditions

Types of Rules

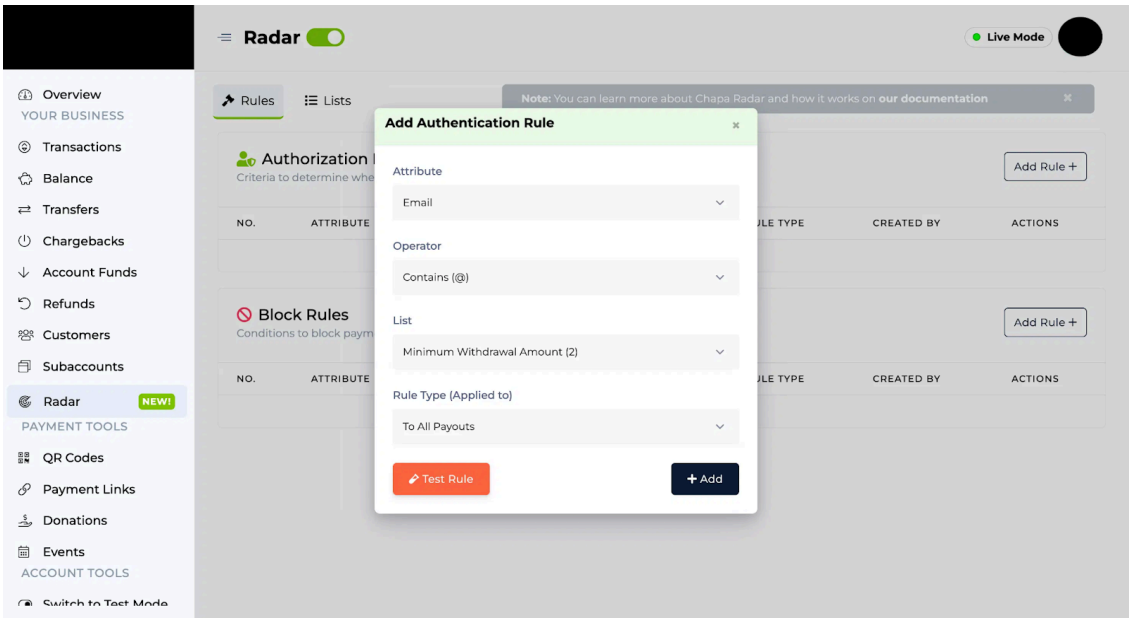
1. **Authorization Rules** – Determine whether a payment should proceed after review
2. **Block Rules** – Automatically deny payments that match specific criteria

How to Add an Authorization Rule:

1. Click **Rules** tab → **Add Rule**



2. Fill in:
    - **Attribute** – The type of data used in the rule (e.g., Mobile, Email, Amount, Daily Amount)
    - **Operator** – Defines how the data is compared:
        - **=** (Equal To)
        - **!=** (Not Equal To)
        - **>** (Greater Than)
        - **<** (Less Than)
        - **>=** (Greater Than or Equal To)
        - **<=** (Less Than or Equal To)
        - **LIKE** (Matches a pattern)
        - **@** (Contains)
        - **!** (Does Not Contain)
    - **List** – Select a previously created list to validate against
    - **Rule Type** – Choose whether the rule applies to:
        - **Payouts** – Outgoing transactions
        - **Payins** – Incoming transactions
3. Click **Test Rule** – Verify if the rule works as expected
4. Click **Add** – a confirmation message appears.



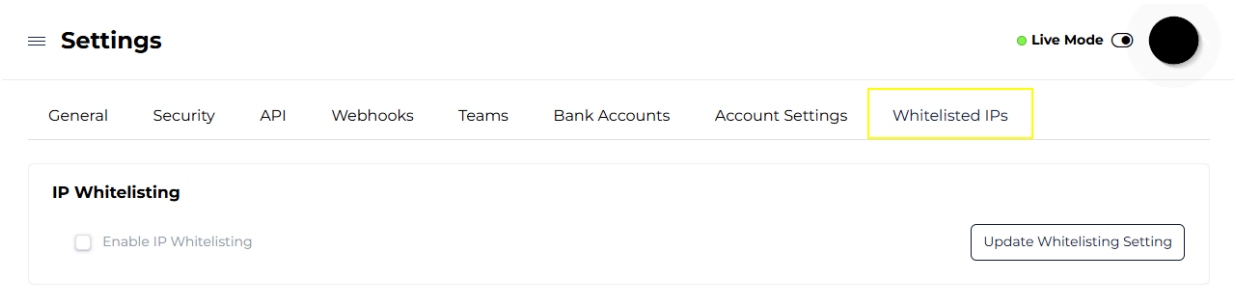
NB: Radars Default status: **Disabled** (Manually enable after review).

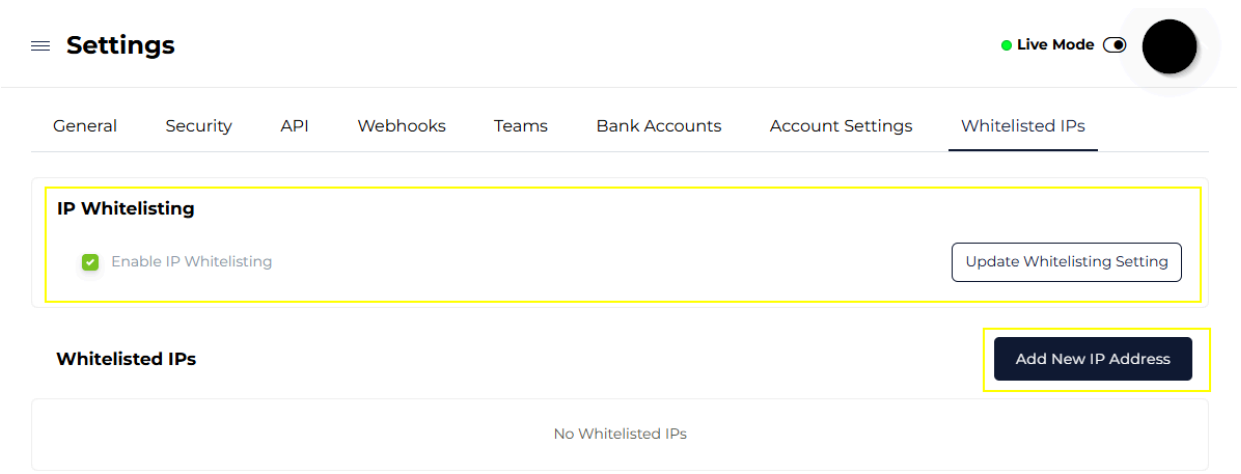
IP Whitelisting

- Enhances security by allowing only trusted IP addresses to access your account
- Any login attempt from an unlisted IP will be **blocked**

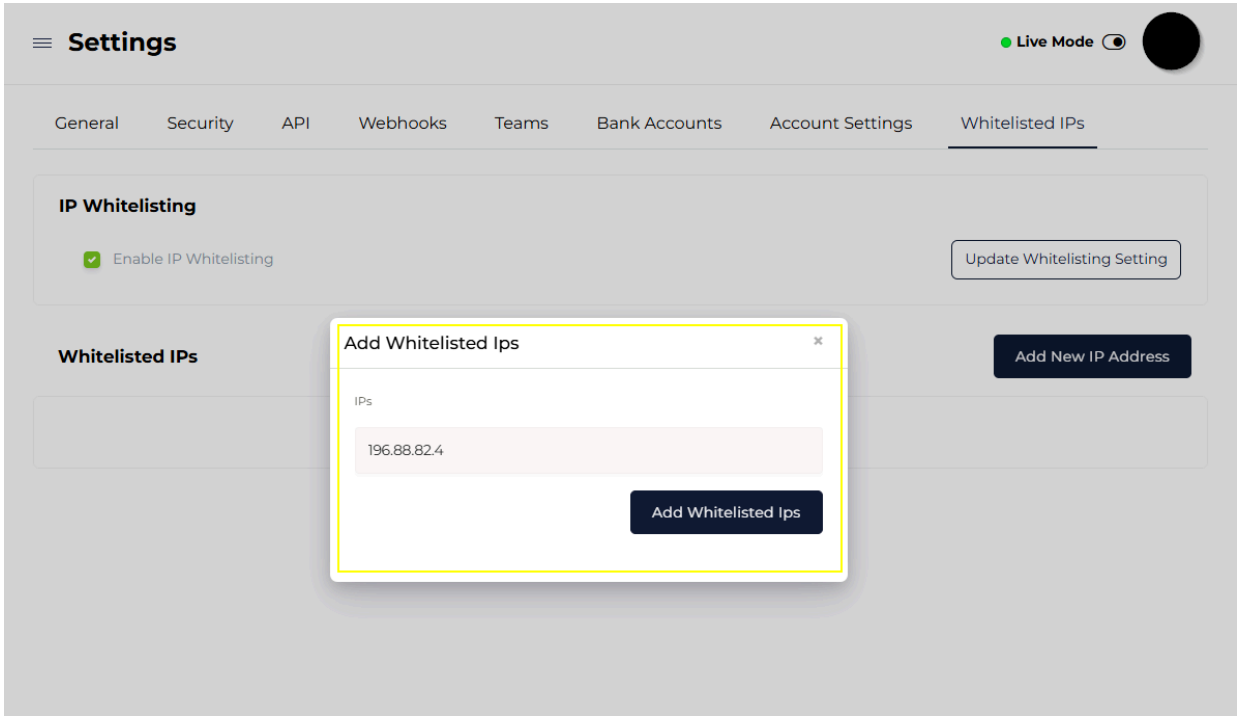
How to Enable IP Whitelisting:

1. Log in to your **Chapa Dashboard**.
2. Go to **Settings → Whitelisted IPs**.
3. Toggle **"Enable IP Whitelisting"** → Click **Update Whitelisting Setting**





4. Click **Add New IP Address** → Enter your **trusted IP**
5. Click **Add Whitelisted IPs** to save



Now, only approved IP addresses can access your account, preventing unauthorized logins.

Encryption for Direct Charge API

- When using **Direct Charge API** to charge a card or send OTP, you must encrypt the payload before making the request.

How to Encrypt the Payload?

1. Get your **encryption key** from **Settings** → **API** in your Chapa Dashboard.
2. Use the **3DES (Triple DES) algorithm** to encrypt the payload.

What to Encrypt?

For **sensitive data** like card details or OTP, encrypt these parameters:

```
Unset
{
                                                    "requestID":
"13434jjfhd8ududfy82e324234234jkhjsfhdfhskdjfh89fjhduohdfjh
sgkdfksjdfskldhfkjs",
    "otp": 1234
}
```

- If OTP is required, **include the request token** returned during transaction initiation.

Example Encryption Function (Python)

```
Python
import json
```

```
import base64

from Crypto.Cipher import DES3


def encryptData(key, plainText):

    blockSize = 8

    padDiff = blockSize - (len(plainText) % blockSize)

                plainText    =    "{}{}".format(plainText,
"".join(chr(padDiff) * padDiff))

    cipher = DES3.new(key, DES3.MODE_ECB)

    encrypted = base64.b64encode(cipher.encrypt(plainText))

    return encrypted
```

- The function:
    1. Converts the **payload to JSON**.
    2. **Pads** the text to match block size.
    3. Uses **3DES encryption** in **ECB mode**.
    4. **Encodes** the result in **Base64**.

API Keys

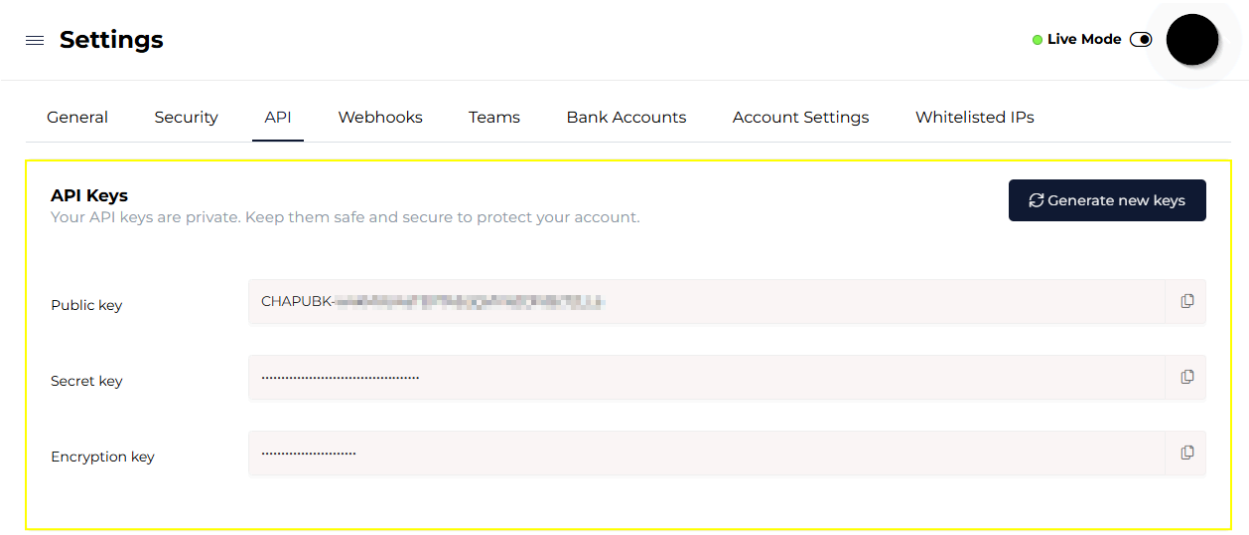
When you sign up on the Chapa dashboard, you receive **two types of API keys**:

1. **Secret Key**
    - The most powerful key, used for **authorizing** actions on your account.
    - **Never share** or expose this key publicly.

2. **Public Key**
    - Used for **client-side operations** (e.g., front-end JavaScript).
    - Safe to include in public code but should still be handled securely.

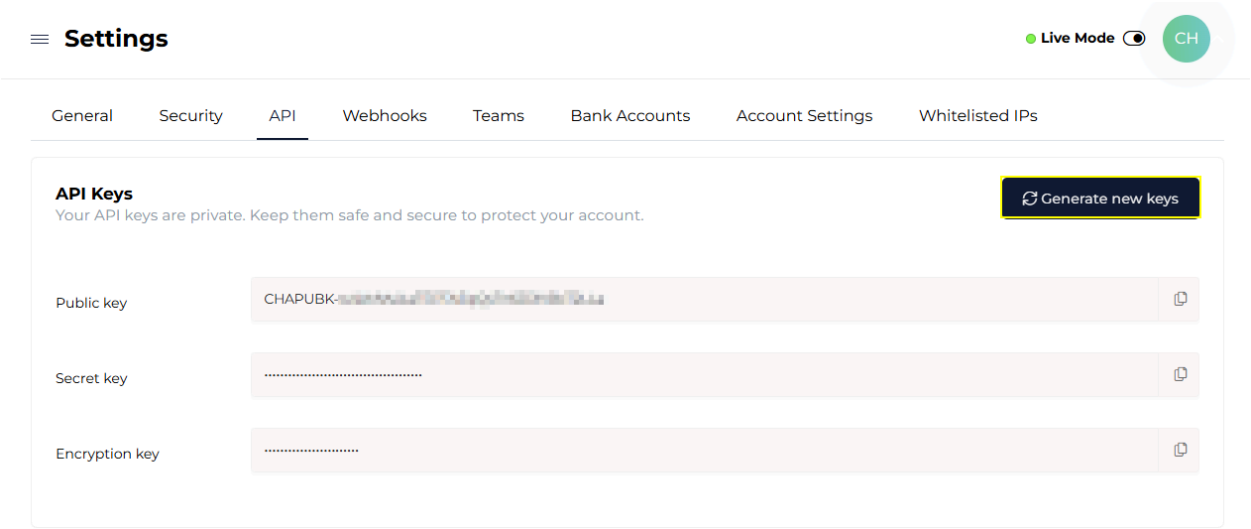
How to Find Your API Keys

1. **Log in** to your Chapa Dashboard.
2. Click the **profile icon** (top-right corner, showing your initials).
3. Go to **Settings** → **API Keys** to view both **public and secret keys**.



Keeping Your Keys Secure

- **Do not expose your secret key** in public repositories or code.
- If a key is **compromised**:
    1. Open **Settings** → **API** in your dashboard.
    2. Click **"Generate new keys"** to revoke the old ones.



3. Update your app with the **new keys**

NB

- ➔ Key with the prefix “**PUBK**” is the Public key
- ➔ Key with the prefix “**SECK**” is the Private/Secret key
- ➔ Public Keys with the prefix "**TEST**" are for **Test mode** account

Verifying Webhook Origin

- To ensure webhook requests come from Chapa, you need to set up a **secret hash**. This helps prevent unauthorized access to your webhook URL.

Setting Up Webhook Verification

1. **Define a Secret Hash**
    - Set a **random and secure** secret hash when configuring your webhook.
    - Store this secret hash securely as an **environment variable** on your server.
2. **How Chapa Uses the Secret Hash**
    - Every webhook request from Chapa includes a **Chapa-Signature** header.
    - This signature is an **HMAC SHA256 hash of your secret key**, signed using the secret key itself.
    - Additionally, we send another header called **x-chapa-signature**, which is an **HMAC SHA256 hash of the event payload**, also signed with your secret key.
3. **Verifying the Webhook Request**
    - In your webhook endpoint, **check for either the `Chapa-Signature` or `x-chapa-signature` header**.
    - Compute the expected hash using your stored secret key and compare it with the received signature.
    - If **either of the headers is missing or invalid, discard the request**.
    - If **both headers are present, at least one must be valid** for the request to be processed.

4. **Ensure Verification Before Processing Events**
    - Always verify the request **before executing any business logic** (e.g., updating payment statuses).

Example Code

```javascript
var crypto = require('crypto');
var secret = process.env.SECRET_KEY;

// Using Express
app.post("/my/webhook/url", function(req, res) {
    // Validate event
    const hash = crypto.createHmac('sha256', secret)
                       .update(JSON.stringify(req.body))
                       .digest('hex');

    if (hash == req.headers['Chapa-Signature']) {
        // Retrieve the request's body
        const event = req.body;
        // Do something with the event
    }

    res.send(200);
});
```

The provided code verifies webhook requests from Chapa using **HMAC SHA256**. Here's how it works:

1. **Retrieve the Secret Key** – The `SECRET_KEY` is stored in environment variables.
2. **Generate a Hash** – It creates an HMAC SHA256 hash of the request body using the secret key.
3. **Compare the Hash** – It checks if the computed hash matches the `Chapa-Signature` header.
4. **Process the Event** – If the signature is valid, the event data is processed. Otherwise, the request is rejected.
5. **Respond to Chapa** – A `200 OK` response is sent to acknowledge receipt.

This ensures that only **authentic** webhook requests from Chapa are accepted.