mercoa api key: 4956c7d4b2944b8aa656a2ae8b604732
Organization Id: org_test_3dd0f7c5-4d90-4d1f-8059-9a0eb7eeba2b

Mercoa Overview

About Mercoa
Mercoa is an embedded financial automation platform. With Mercoa, any B2B platform can easily embed, offer, and monetize AP and AR automation for their users. Our full-service automation solutions allow you to launch invoice processing, approval workflows & automations, payment scheduling, billpay, working capital, email notifications, AI agents, and much more.

Account Structure
Mercoa's account structure is designed to support complex B2B platforms with multiple customers and vendors. We use the following terminology to describe our account structure:

Account Description
C1 The platform/product team that embeds Mercoa (whether via our embed, React components, or API) within their product. If you are reading this, you are probably a C1.
C2 The customer of a C1. A C2 is an entity that uses the C1's product and has access to Mercoa via the C1's product.
C3 The counterparty of a C2. A C3 is an entity that either sends invoices to or receives invoices from a C2.
Accounts Payable vs. Accounts Receivable
In Mercoa's accounts payable experience, the C2 is the payer and the C3 is the vendor.

In Mercoa's accounts receivable experience, the C2 is the vendor and the C3 is the payer.

AI Agents
Mercoa can also augment your platform's payments experience with our integrated AI agent functionality. Our AI agents plugs directly into the rest of Mercoa's infrastructure, allowing you to build custom AI workflows that can:

Read and understand complex B2B contracts
Contact customers to collect overdue payments
Contact vendors to send upcoming invoices on time
Edit invoices and billing schedules with natural language
More coming soon!
Check out our demo and documentation for MAi, our Invoicing Agent for accounts receivable!

API Object Model
Mercoa's API is based around CRUD operations organized around a set of objects that represent the data in Mercoa.

Step 1: API Keys and Concepts

Creating your Mercoa Instance
To create your Mercoa instance, please book a call with our team here. We'll help you get set up and answer any questions you may have.

Getting your Mercoa Organization Details
You can find your organization's API key and Organization ID in the Mercoa Dashboard.

Do not expose this key on the front-end, it is for back-end use only.
Mercoa Architecture
Mercoa architecture overview

Creating a Mercoa Entity
An entity in Mercoa is an individual or business (C2) that pays or sends invoices through your platform.

When creating your first entity, we recommend using the Mercoa Dashboard.

You can also create an entity using the Mercoa API.

POST
/entity

cURL

```
curl -X POST https://api.mercoa.com/entity \
     -H "Authorization: Bearer <token>" \
     -H "Content-Type: application/json" \
     -d '{
  "isCustomer": true,
  "isPayor": true,
  "isPayee": false,
  "accountType": "business",
  "foreignId": "MY-DB-ID-12345",
  "profile": {
    "business": {
      "email": "customer@acme.com",
      "legalBusinessName": "Acme Inc.",
      "website": "http://www.acme.com",
      "businessType": "llc",
      "phone": {
        "countryCode": "1",
        "number": "4155551234"
```

```
      },
      "address": {
        "addressLine1": "123 Main St",
        "addressLine2": "Unit 1",
        "city": "San Francisco",
        "stateOrProvince": "CA",
        "postalCode": "94105",
        "country": "US"
      },
      "taxId": {
        "ein": {
          "number": "12-3456789"
        }
      }
    }
  }
}'
```

You can create an entity with just a name and email address. Mercoa can automatically collect the rest of the entity's information with our onboarding flows for payers in AP and vendors in AR.

Once you've created your payer, you will have an entityId that you can use to create a token for the payer.

Entity IDs always start with ent_ followed by a UUID.

Enable Mercoa Payments

If you are using Mercoa's payment rails, you will need to collect data required to run KYB. Follow our payments guide for more information.

Step 2: Backend Integration

If you are using Mercoa for just payments check out our Creating Payouts via API guide.

Mercoa has a fully documented REST API that can be used in any language. We also have Node, Python, Java, .NET, and Go SDKs for easy backend integration.

At Mercoa, we aim to provide a seamless and whitelabeled experience for your users. As part of this experience, Mercoa does not force users to create a new account or log in to a different system.

Instead, Mercoa uses JWT tokens that you can generate to transparently authenticate the user session on the frontend.

Steps to generate a token:
User logs into your platform
Find the Mercoa Entity that corresponds to that user's business.
Optional: Sync individual users and their roles. This is required for Approvals.
Generate a JWT and pass it to the frontend.
Use the JWT with our frontend SDK, React Components, or embedded iFrame.
Creating a Token
Let's create an endpoint that authenticates the user, generates a JWT with the entityId, and return the generated token. We will use Mercoa's Generate JWT Token endpoint to make this easy.

Python Django

Node Express

Next.js

```
from django.http import HttpResponse
from mercoa.client import Mercoa
client = Mercoa(token="YOUR_API_KEY")
def generate_mercoa_token(request):
  client.entity.get_token(entity_id="ENTITY_ID_FROM_STEP_ONE",{})
  return HttpResponse(token)
```

Using the Token
Now that we have a token, we can use it to authenticate the user in our frontend application. Tokens have a default expiration of 1 hour, but you can change this by passing in the expiresIn option when generating the token.

Step 3: Frontend Integration

Now that you have the JWT token, you can build your frontend with Mercoa's React components (recommended) or the embedded iFrame.

The React components give you full control over the design and layout of your frontend as well as an all-in-one experience for a quick start, while the embedded iFrame is a simpler drop-in integration for the Mercoa AP inbox.

React (Next.js)

Embedded iFrame

```
// Remember to install the react library: npm install --save @mercoa/react
// Create a new page called bills.jsx
import { useEffect, useState } from 'react'
import { MercoaSession } from '@mercoa/react'
const MercoaComponent = () => {
  const [token, setToken] = useState(null)
  useEffect(() => {
    // Call Your Token Generator Endpoint
    fetch('/generateMercoaToken').then(async (resp) => {
      if (resp.status === 200) {
        setToken(await resp.text())
      }
    })
  }, [])
  // NOTE: This is the all-in-one experience.
  // See react.mercoa.com to learn how to use
  // our individual React components.
  return <MercoaSession token={token} />
}
export default function Bills() {
  return <MercoaComponent />
}
```

Building with React

Mercoa's React components (documented on react.mercoa.com) enable you to quickly ship and customize your payment platform's frontend to fit your needs.

To render correctly, all Mercoa-powered frontend elements must be inside a <MercoaSession> component with a valid JWT token. The <MercoaSession> component handles authentication and provides a React Context called MercoaContext to its children, which encodes all the information you need to build your frontend.

To access this information in your frontend, you can drop in and customize Mercoa's React components or use the useMercoaSession hook to render a completely custom frontend. For most use cases, the React components are the easiest way to get started.

For more details, see the React SDK documentation.

Customizing the Mercoa iFrame with a JWT Token
Mercoa also supports some customization of the embedded iFrame via a JWT token. When you generate a JWT token, you can pass in additional options in the request body:

```
{
  ...
  "options": {
    "entity": {
      "enableMercoaPayments": true // If this is true, the user will run though Mercoa's KYB onboarding and will be able to use our built-in payment rails. If you are using your own payment rails, set this to false.
    },
    "invoice": {
      "lineItems": "REQUIRED",
      "status": [
        "DRAFT",
        "NEW",
        "APPROVED",
        "SCHEDULED",
        "PENDING",
        "PAID",
        "CANCELED",
        "FAILED"
      ] // This is an array of invoice statuses that will be displayed in the Mercoa AP inbox.
    },
    "pages": {
      "paymentMethods": true, // If this is true, the user will be able to add payment methods in the Mercoa AP inbox.
      "representatives": true, // If this is true, the user will be able to add business representatives in the Mercoa AP inbox.
      "notifications": true // If this is true, the user will be able to view notifications in the Mercoa AP inbox.
    },
    "vendors": {
      "disableCreation": false, // If this is true, the user will not be able to create new vendors in the Mercoa AP inbox.
```

```
      "network": "all" // This limits the vendors that will be displayed in the Mercoa AP inbox. Options are "entity" which limits vendors to those created by your customer, "platform" which will show all vendors created by any of your customers, or "all" which shows additional verified vendors created by Mercoa.
    }
  }
}
```

SDKs
Python

If you're working with Python, the official mercoa package is the easiest way to interact with the Mercoa API.

Installation
Add this dependency to your project's build file:

```
pip install mercoa
# or
poetry add mercoa
```

Usage

```
from mercoa.client import Mercoa
mercoa_client = Mercoa(token="YOUR_API_KEY")
entity = mercoa_client.entity.get(entity_id='YOUR_ENTITY_ID')
print(entity)
```

Async client
This SDK also includes an async client, which supports the await syntax:

```
import asyncio
from mercoa.client import AsyncMercoa
mercoa_client = AsyncMercoa(token="YOUR_API_KEY")
async def get_entity() -> None:
    entity = await mercoa_client.entity.get(entity_id='YOUR_ENTITY_ID')
    print(entity)
asyncio.run(get_entity())
```