Centro de Documentación Pago46

Documentación técnica de la API de Pago46 · v2026.8.2

Contenido

La red de efectivo de Latinoamérica

Pago46 conecta el mundo digital con el efectivo en Latinoamérica. Con una sola integración a la **API de Pago46**, procesas **pagos (Pay-In)** y **retiros (Pay-Out)** en efectivo mediante puntos físicos, como cadenas comerciales, bancos, farmacias y kioscos, y una red de agentes que recolectan y entregan dinero en puntos geolocalizados.

Actores

La API coordina transacciones entre tres roles. Identifica el tuyo para saber qué guía seguir.

Actor	Rol	Descripción
Comercio (Merchant)	Inicia la orden	Negocios online, e-commerce, wallets o fintechs que necesitan **cobrar** a sus usuarios o **dispersar** fondos.
Proveedor de Pago (Payment Provider)	Procesa la orden	Redes físicas (bancos, farmacias, kioscos) que **procesan** en efectivo las órdenes creadas por los comercios.
Consumidor	Entrega o recibe el efectivo	El usuario final que paga o cobra en un punto de la red.

Servicios

La API de Pago46 maneja moneda local y moneda extranjera con la misma semántica.

Pay-In (Pagos)

Tus usuarios pagan en efectivo en cualquier punto de la red.

- **Comercio:** creas una orden y obtienes un código/QR; recibes el dinero conciliado digitalmente.
- **Proveedor:** validas el código, recibes el efectivo y confirmas la operación.

Pay-Out (Retiros)

Tus usuarios retiran efectivo o reciben una dispersión en un punto físico.

- **Comercio:** creas una orden de retiro para tu usuario.
- **Proveedor:** verificas la orden, la bloqueas y entregas el efectivo.

¿Cómo funciona la integración?

La arquitectura es de **servidor a servidor** y cada petición se firma con **HMAC SHA-256**.

1. **Autenticación** — firmas cada petición con HMAC SHA-256.
2. **Creación de la orden** — el Comercio crea la intención de pago o retiro.
3. **Operación física** — el usuario acude a un punto de la red.
4. **Confirmación** — el Proveedor confirma que el dinero cambió de manos.
5. **Webhooks** — Pago46 notifica el resultado al Comercio en tiempo real.

ANTES DE EMPEZAR

La Autenticación es el requisito previo para cualquier llamada a la API. Empieza por ahí.

Elige tu guía

Soy un Comercio

Quiero cobrar pagos o enviar dinero a mis usuarios.

- Cobros (Pay-In)
- Retiros (Pay-Out)
- Retiros en moneda extranjera

Soy un Proveedor de Pago

Tengo puntos físicos y quiero procesar transacciones de Pago46.

- Procesar pagos
- Procesar retiros

Antes de integrar, revisa la cobertura por país para confirmar la disponibilidad operativa.

Países

Cobertura regional

Con **una sola integración** te conectas a la red de pagos en efectivo más extensa de Latinoamérica. Pago46 opera de forma nativa o a través de *Redes de Pago* aliadas, garantizando una liquidación unificada y SLAs homogéneos en todos los mercados.

Países activos y monedas locales fiduciarias

Actualmente procesamos operaciones en moneda local fiduciaria en los siguientes territorios. Asegúrate de usar la moneda correcta en tus solicitudes a la API para cada jurisdicción.

País	Moneda aceptada	Código ISO de moneda
🇦🇷 **Argentina**	Peso argentino	`ARS`
🇨🇱 **Chile**	Peso chileno	`CLP`
🇪🇨 **Ecuador**	Dólar estadounidense	`USD`
🇲🇽 **México**	Peso mexicano	`MXN`
🇵🇪 **Perú**	Sol	`PEN`
🇬🇹 **Guatemala**	Quetzal	`GTQ`

> **Configuración por país**
>
> Se puede realizar una configuración específica de comportamiento y funcionalidad para cada país, permitiendo definir distintos flujos de pago, tarifas y reglas de negocio adaptadas a las normativas locales.

Moneda extranjera

Ofrecemos la funcionalidad de moneda extranjera para procesar transacciones con conversión de divisas. Esto permite a tus usuarios pagar o recibir fondos en una moneda diferente de la local, lo que facilita la expansión internacional sin necesidad de múltiples integraciones.

Divisa de origen → Divisa de destino	Descripción	Países disponibles
USDC → MXN	Permite a los usuarios enviar **USDC** vía **Solana**, **Monad** o **Stellar** y **recibir pesos mexicanos en puntos físicos inmediatamente**.	🇲🇽 México

---

Próximos pasos

Selecciona la guía que mejor se adapte a tu necesidad de integración:

Soy un Comercio

Quiero cobrar pagos o enviar dinero a mis usuarios.

- Cobros (Pay-In)
- Retiros (Pay-Out)
- Retiros en moneda extranjera

Soy un Proveedor de Pago

Tengo puntos físicos y quiero procesar transacciones de Pago46.

- Procesar pagos
- Procesar retiros

¿Dudas sobre la expansión a tu país? Contacta a tu ejecutivo comercial o escríbenos a contacto@pago46.com.

Ambientes

Pago46 opera en dos entornos: **Sandbox** para pruebas y **Producción** para transacciones reales. Cada ambiente tiene sus propias características y requisitos de configuración.

Flujo tradicional de integración

1. Identifica si eres un **Comercio (Merchant)** o un **Proveedor de Pago (Payment Provider)**.
2. Identifica el flujo que deseas integrar: **Cobros (Pay-In)** o **Retiros (Pay-Out)**.
3. Envía un correo a `contacto@pago46.com` indicando tu rol (Comercio o Proveedor de Pago), país de operación y flujo a integrar.
4. Recibirás credenciales para el ambiente **Sandbox** y acceso a la documentación.
5. Realiza las pruebas necesarias en **Sandbox**.
6. Cuando estés listo para producción, solicita las credenciales de producción a tu ejecutivo comercial.

Pruebas (Sandbox)

Servicio Pago46	URL base	Notas
API principal	`https://api.dev.pago46.io/api/v1`	Punto de integración de la API REST sobre HTTPS con firma HMAC.
Checkout (aplicación web de pago)	`https://checkout.dev.pago46.io`	Página alojada para que los consumidores completen flujos de pago.

*Los montos procesados en sandbox **no** generan movimientos reales de dinero.*

Producción

Servicio Pago46	URL base	Notas
API principal	`https://api.prd.pago46.io/api/v1`	Punto de integración de la API REST sobre HTTPS con firma HMAC.
Checkout (aplicación web de pago)	`https://checkout.prd.pago46.io`	Página alojada para que los consumidores completen flujos de pago.
App Pago46 (web)	`https://app.pago46.io`	Funciona en cualquier navegador, sin instalar nada. Se puede añadir a la pantalla de inicio en iPhone y Android.
App Pago46 (Android)	Google Play	La misma app publicada en Google Play.

*Cualquier usuario puede crear una cuenta en la app Pago46; para operar como **Socio46** es necesario completar la verificación de identidad.*

Monitoreo de disponibilidad

Para visibilidad operativa de producción, revisa la página de Estado del Servicio, donde encontrarás incidentes activos, mantenimientos programados y opciones de suscripción por correo y Slack.

> ⚠ **CREDENCIALES DISTINTAS**
>
> Las llaves sandbox y producción **no son intercambiables**. Solicítalas a tu ejecutivo comercial.

Elige tu guía

Soy un Comercio

Quiero cobrar pagos o enviar dinero a mis usuarios.

- Cobros (Pay-In)
- Retiros (Pay-Out)
- Retiros en moneda extranjera

Soy un Proveedor de Pago

Tengo puntos físicos y quiero procesar transacciones de Pago46.

- Procesar pagos
- Procesar retiros

Autenticación

Cada petición a la API de Pago46 se firma con **HMAC SHA-256**. Tú calculas una firma a partir de tu clave secreta y del contenido de la petición; el servidor la recalcula y compara ambas firmas. Si coinciden, confirma dos cosas: que la petición es tuya y que nadie alteró el contenido en tránsito.

El esquema es idéntico para comercios y proveedores de pago. Lo único que cambia es el nombre de la cabecera donde envías tu clave pública.

Cabeceras obligatorias

Cada petición HTTP debe incluir estas tres cabeceras:

Cabecera	Descripción
`Merchant-Key` o `Provider-Key`	Tu clave pública. Usa `Merchant-Key` si eres comercio y `Provider-Key` si eres proveedor de pago.
`Message-Date`	Marca de tiempo Unix en **segundos** (entero o decimal).
`Message-Hash`	La firma HMAC SHA-256 en hexadecimal (ver abajo).

> **VENTANA DE VALIDEZ**
>
> El servidor rechaza la petición si `Message-Date` difiere de la hora del servidor en más de **5 minutos**, para prevenir ataques de repetición. Usa siempre la hora actual y **en segundos**: enviar milisegundos la coloca fuera de la ventana y la petición se rechaza.

Construir la firma

1. Cadena a firmar

Concatena estos cinco valores separados por dos puntos (`:`), en este orden exacto:

```
LLAVE:MESSAGE_DATE:MÉTODO:PATH:BODY
```

1. **Clave pública** — el mismo valor de tu cabecera `Merchant-Key` o `Provider-Key`.
2. **Message-Date** — el mismo valor que envías en la cabecera.
3. **Método HTTP** — `POST`, `GET`, etc.
4. **Ruta** — la ruta del recurso **sin cadena de consulta**, por ejemplo `/api/v1/merchants/orders/pay-in/`.
5. **Body** — el cuerpo crudo de la petición en UTF-8. Si la petición no tiene cuerpo (por ejemplo un `GET`), usa una cadena vacía.

No se ordenan parámetros ni se normaliza el JSON. Firma exactamente los mismos bytes que envías en el cuerpo: cualquier espacio o cambio de orden produce una firma distinta.

2. Calcular el hash

Firma la cadena con HMAC SHA-256 usando tu **clave secreta** y representa el resultado en hexadecimal (`hexdigest`).

Ejemplos de implementación

El mismo firmante sirve para comercios y proveedores; solo cambia el nombre de la cabecera de la clave.

Python

```
import hashlib
import hmac
import json
import time

import requests

HOST = "https://api.dev.pago46.io"


def signed_request(key, secret, key_header, method, path, body=None):
    timestamp = str(time.time())  # Unix en segundos
    body_str = json.dumps(body) if body else ""

    string_to_sign = f"{key}:{timestamp}:{method}:{path}:{body_str}"
    signature = hmac.new(
        secret.encode("utf-8"),
        string_to_sign.encode("utf-8"),
        hashlib.sha256,
    ).hexdigest()

    headers = {
        key_header: key,  # "Merchant-Key" o "Provider-Key"
        "Message-Date": timestamp,
        "Message-Hash": signature,
        "Content-Type": "application/json",
    }
    return requests.request(method, f"{HOST}{path}", headers=headers, data=body_str)


# Ejemplo: un Comercio crea una orden de pay-in
response = signed_request(
    key="TU_MERCHANT_KEY",
    secret="TU_MERCHANT_SECRET",
    key_header="Merchant-Key",
    method="POST",
    path="/api/v1/merchants/orders/pay-in/",
    body={"order_type": "LocalCurrencyOrder", "price": "100.00", "price_currency": "CLP"},
)
print(response.status_code)
```

Node.js

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

const HOST = 'https://api.dev.pago46.io';

async function signedRequest({ key, secret, keyHeader, method, path, body = null }) {
  const timestamp = (Date.now() / 1000).toString(); // Unix en segundos
  const bodyStr = body ? JSON.stringify(body) : '';

  const stringToSign = [key, timestamp, method, path, bodyStr].join(':');
  const signature = crypto
    .createHmac('sha256', secret)
    .update(stringToSign)
    .digest('hex');

  return axios({
    method,
    url: `${HOST}${path}`,
    headers: {
      [keyHeader]: key, // 'Merchant-Key' o 'Provider-Key'
      'Message-Date': timestamp,
      'Message-Hash': signature,
      'Content-Type': 'application/json',
```

```
    },
    data: bodyStr,
  });
}
```

Go

```
package main

import (
        "crypto/hmac"
        "crypto/sha256"
        "encoding/hex"
        "strconv"
        "strings"
        "time"
)

// keyHeader es "Merchant-Key" o "Provider-Key".
func authHeaders(key, secret, keyHeader, method, path, bodyStr string) map[string]string {
        timestamp := strconv.FormatInt(time.Now().Unix(), 10) // Unix en segundos

        stringToSign := strings.Join([]string{key, timestamp, method, path, bodyStr}, ":")

        h := hmac.New(sha256.New, []byte(secret))
        h.Write([]byte(stringToSign))
        signature := hex.EncodeToString(h.Sum(nil))

        return map[string]string{
                keyHeader:      key,
                "Message-Date": timestamp,
                "Message-Hash": signature,
        }
}
```

Errores de autenticación

Cuando la autenticación falla, la API devuelve **`403 Forbidden`** con el formato estándar de errores:

```
{
  "type": "client_error",
  "errors": [
    {
      "code": "authentication_failed",
      "detail": "Incorrect authentication credentials.",
      "attr": null
    }
  ]
}
```

Causa probable	Cómo resolverlo
Falta una cabecera obligatoria	Incluye `Merchant-Key` / `Provider-Key`, `Message-Date` y `Message-Hash`.
`Message-Date` fuera de la ventana de 5 minutos	Usa la hora actual del sistema en segundos.
Llave pública desconocida	Verifica que usas la llave del ambiente correcto (sandbox vs. producción).
La firma no coincide	Revisa el orden de la cadena, el separador `:`, el path sin query string y que el body sea idéntico al enviado.

Elige tu guía

Soy un Comercio

Quiero cobrar pagos o enviar dinero a mis usuarios.

- Cobros (Pay-In)
- Retiros (Pay-Out)
- Retiros en moneda extranjera

Soy un Proveedor de Pago

Tengo puntos físicos y quiero procesar transacciones de Pago46.

- Procesar pagos
- Procesar retiros

OpenAPI

El contrato OpenAPI de Pago46 es la referencia oficial para integrar nuestros servicios REST. Define endpoints, métodos, parámetros, payloads, respuestas y esquemas de error, ayudándote a implementar una integración más rápida y con menos retrabajo.

Recursos disponibles

- **Contrato OpenAPI (YAML)**: Descargar especificación
- **Swagger UI**: Explorar y probar endpoints
- **Redoc UI**: Navegar la documentación

Uso recomendado para integradores

1. **Empieza por el contrato** para entender recursos, operaciones y modelos.
2. **Valida autenticación y cabeceras** antes de consumir endpoints de negocio.
3. **Usa Swagger UI o Redoc** para revisar ejemplos y comportamientos esperados.
4. **Genera artefactos técnicos** (clientes, mocks o pruebas) desde el YAML cuando aplique a tu stack.
5. **Prueba en Sandbox** y promueve a producción solo después de validar flujos críticos y manejo de errores.

Buenas prácticas

- Toma OpenAPI como fuente de verdad técnica durante todo el ciclo de integración.
- Implementa validaciones de esquema para prevenir errores de formato y tipos.
- Estandariza el manejo de errores con base en los códigos y respuestas documentados.
- Revisa periódicamente la documentación para identificar ajustes y nuevas capacidades.

Elige tu guía

Soy un Comercio

Quiero cobrar pagos o enviar dinero a mis usuarios.

- Cobros (Pay-In)
- Retiros (Pay-Out)
- Retiros en moneda extranjera

Soy un Proveedor de Pago

Tengo puntos físicos y quiero procesar transacciones de Pago46.

- Procesar pagos
- Procesar retiros

Integra con IA

¿Vas a integrar Pago46 con ayuda de un asistente de IA? Ponemos a disposición de los asistentes el contexto necesario. Nuestra documentación publica un índice `llms.txt` que enlaza a una copia limpia en **Markdown** de cada página, para que cualquier asistente pueda leer los flujos de Pay-In, Pay-Out, autenticación y Checkout sin tropezar con el HTML del sitio.

Elige tu asistente abajo para abrirlo con el prompt de integración ya cargado, o copia un comando listo para ejecutar en tu agente de programación de terminal.

Nuestra documentación publica un índice `llms.txt` que apunta a una copia en Markdown de cada página. Abre tu asistente con el prompt de integración listo, o copia un comando para tu agente de programación en la terminal.

Copiar prompt | Abrir llms.txt

CHATEA EN TU NAVEGADOR

Abre el asistente con el prompt cargado. Gemini no admite prellenado: usa «Copiar prompt» y pégalo.

- **Claude** — Anthropic →
- **ChatGPT** — OpenAI →
- **Perplexity** — Search + browse →
- **Grok** — xAI →
- **Le Chat** — Mistral AI →
- **Gemini** — Google →

USA TU AGENTE DE PROGRAMACIÓN

Copia un comando listo para ejecutar en tu terminal.

- **Claude Code** — Anthropic CLI
- **Codex CLI** — OpenAI
- **Gemini CLI** — Google
- **opencode** — Open source
- **Cursor** — cursor-agent CLI
- **Aider** — Open source

Los prompts

Estos son los prompts exactos que usan los botones de arriba. Cópialos y adáptalos a tu caso —por ejemplo, indica tu lenguaje, framework o si integras como comercio o como proveedor de pago.

Para asistentes en el navegador (Claude, ChatGPT, Perplexity, Grok, Le Chat, Gemini):

```
Estoy integrando los pagos en efectivo de Pago46 (pay-in y pay-out) para
Latinoamérica. Lee el índice de documentación en https://docs.pago46.io/llms.txt
— enlaza a una copia en Markdown de cada página — y luego ayúdame a planear e
implementar la integración. Empieza por descargar esa URL.
```

Para agentes de programación en la terminal (Claude Code, Codex, Gemini CLI, OpenCode, Cursor, Aider):

```
Lee https://docs.pago46.io/llms.txt y las páginas que enlaza, luego ayúdame a
integrar el pay-in y pay-out en efectivo de Pago46.
```

COPIA CUALQUIER PÁGINA A TU ASISTENTE

Cada página de esta documentación incluye un botón **Copiar página** (arriba a la derecha) que copia su Markdown, o la abre directamente en ChatGPT o Claude. Úsalo cuando tengas una duda puntual sobre una página específica.

¿Cómo funciona?

- `llms.txt` — un índice en la raíz del sitio que lista cada página con su título y un enlace a la versión Markdown. Es el punto de partida ideal para un asistente: ábrelo aquí.
- **Markdown por página** — cada página existe también como `.md` (por ejemplo `authentication.md`), sin la interfaz del sitio, para que el modelo lea solo el contenido.
- **Datos siempre frescos** — estos archivos se generan en cada publicación de la documentación, así que el asistente lee la versión vigente, no una copia desactualizada.

Para la referencia completa de endpoints y payloads, apunta también a tu asistente a la referencia OpenAPI.

Manejo de errores comunes

Formato de respuesta de error

El formato de respuesta de error es el siguiente:

```
{
  "type": "validation_error",
  "errors": [
    {
      "code": "required",
      "detail": "This field is required.",
      "attr": "name"
    }
  ]
}
```

- `type`: puede ser `validation_error`, `client_error` o `server_error`.
- `code`: cadena corta que describe el error. Puedes usarla para adaptar el comportamiento de tu integración.
- `detail`: texto descriptivo y comprensible para el usuario sobre el error. Por lo general está en inglés, pero puede traducirse.
- `attr`: puede ser `null` cuando el error no está asociado a un campo específico; cuando aplica, contiene el nombre del campo, por ejemplo en errores de tipo `validation_error`.

Errores comunes

Algunos errores pueden aparecer en la mayoría de nuestros endpoints. En las siguientes secciones describimos los más comunes. Para más detalles, consulta los recursos enlazados en OpenAPI.

403 No autorizado

Estos errores se devuelven con el código de estado 403 siempre que la autenticación falla o se hace una solicitud a un endpoint sin proporcionar información de autenticación. Estos son los dos errores posibles:

```
{
    "type": "client_error",
    "errors": [
        {
            "code": "authentication_failed",
            "detail": "Incorrect authentication credentials.",
            "attr": null
        }
    ]
}
```

```
{
    "type": "client_error",
    "errors": [
        {
            "code": "not_authenticated",
            "detail": "Authentication credentials were not provided.",
            "attr": null
        }
    ]
}
```

405 Método no permitido

Este error se devuelve cuando se llama a un endpoint con un método HTTP inesperado. Por ejemplo, si para actualizar un usuario se requiere POST y en su lugar se usa PATCH, se devuelve este error. Se ve así:

```
{
    "type": "client_error",
    "errors": [
        {
            "code": "method_not_allowed",
            "detail": "Method \"PATCH\" not allowed.",
            "attr": null
        }
    ]
}
```

406 No aceptable

Este error se devuelve si se envía la cabecera `Accept` con un valor distinto a `application/json`. La respuesta sería:

```
{
    "type": "client_error",
    "errors": [
        {
            "code": "not_acceptable",
            "detail": "Could not satisfy the request Accept header.",
            "attr": null
        }
    ]
}
```

415 Tipo de medio no soportado

Este error se devuelve cuando el tipo de contenido de la solicitud no es JSON. La respuesta sería:

```
{
    "type": "client_error",
    "errors": [
        {
            "code": "unsupported_media_type",
            "detail": "Unsupported media type \"application/xml\" in request.",
            "attr": null
        }
    ]
}
```

500 Error interno del servidor

Este error se devuelve cuando el servidor de la API encuentra un error inesperado. La respuesta sería:

```
{
    "type": "server_error",
    "errors": [
        {
            "code": "error",
            "detail": "A server error occurred.",
            "attr": null
        }
    ]
}
```

Elige tu guía

Soy un Comercio

Quiero cobrar pagos o enviar dinero a mis usuarios.

- Cobros (Pay-In)
- Retiros (Pay-Out)
- Retiros en moneda extranjera

Soy un Proveedor de Pago

Tengo puntos físicos y quiero procesar transacciones de Pago46.

- Procesar pagos
- Procesar retiros

Integración para Comercios

Como comercio en Pago46, accedes a una red híbrida de puntos físicos donde tus usuarios pueden pagar o recibir efectivo. Tú gestionas todo digitalmente mediante nuestra API.

Servicios disponibles

Pay-In (Cobros)

Permite que tus usuarios paguen con efectivo en cualquier punto de la red.

Casos de uso: Recargas de billeteras, suscripciones, e-commerce, servicios.

Ver documentación de Pay-In

Pay-Out (Dispersiones)

Permite que tus usuarios retiren efectivo o reciban pagos en puntos físicos.

Casos de uso: Retiros de saldo, pagos a trabajadores, remesas, reembolsos.

Ver documentación de Pay-Out

Moneda extranjera

Operaciones con conversión de divisas.

Ver documentación de Moneda extranjera

Checkout

Integra la aplicación web de Checkout en un iframe para flujos Pay-In y Pay-Out.

Ver documentación de Checkout

CHECKOUT DE PAGO46

Las órdenes contienen el campo `redirect_url`, que corresponde al enlace de Checkout para que tu usuario continúe el flujo.

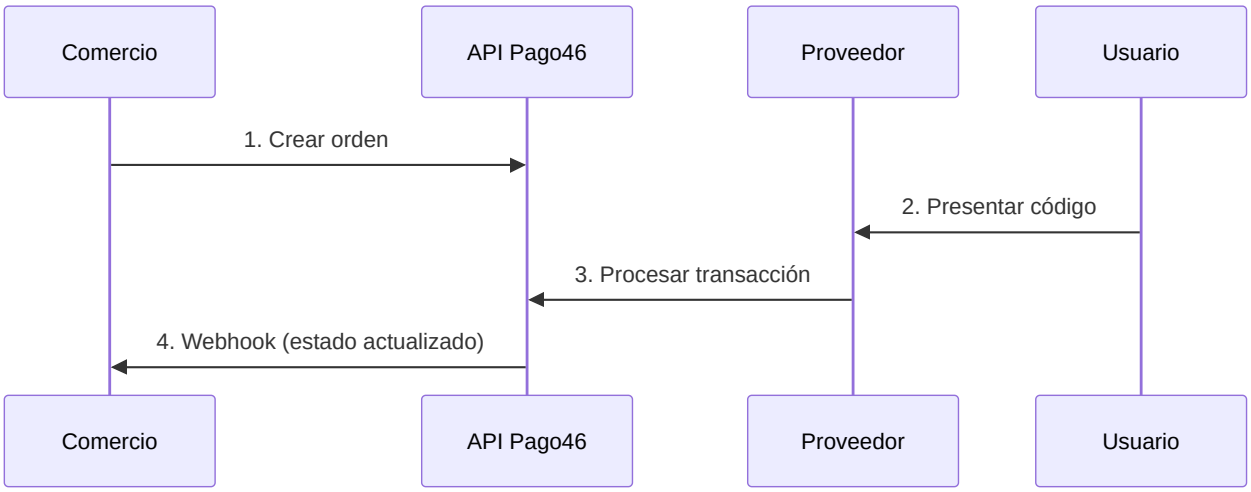
- Sandbox: `https://checkout.dev.pago46.io`
- Producción: `https://checkout.prd.pago46.io`
- Formato por orden: `https://checkout.{dev|prd}.pago46.io/{UUID}`

También puedes entrar a la página principal de Checkout (sin UUID) para ingresar manualmente el identificador de la orden durante las pruebas de integración en un iframe.

> Ver contenido incrustado en línea: https://drive.google.com/file/d/19XG84b29aGuBPA_VWpPrI1J4QknkosVe/preview

Flujo de integración

La arquitectura es de servidor a servidor con autenticación HMAC SHA-256.



Pasos de integración

1. **Obtén tus credenciales** - Recibe tu `Merchant-Key` y `Merchant-Secret`
2. **Implementa autenticación HMAC** - Ver guía
3. **Crea órdenes** - Usa `POST /merchants/orders/pay-in` o `/pay-out`
4. **Recibe webhooks** - Configura tu endpoint HTTPS para notificaciones

> **REQUISITO FUNDAMENTAL**
>
> Todas las peticiones requieren autenticación HMAC. Revisa la documentación de autenticación antes de comenzar.

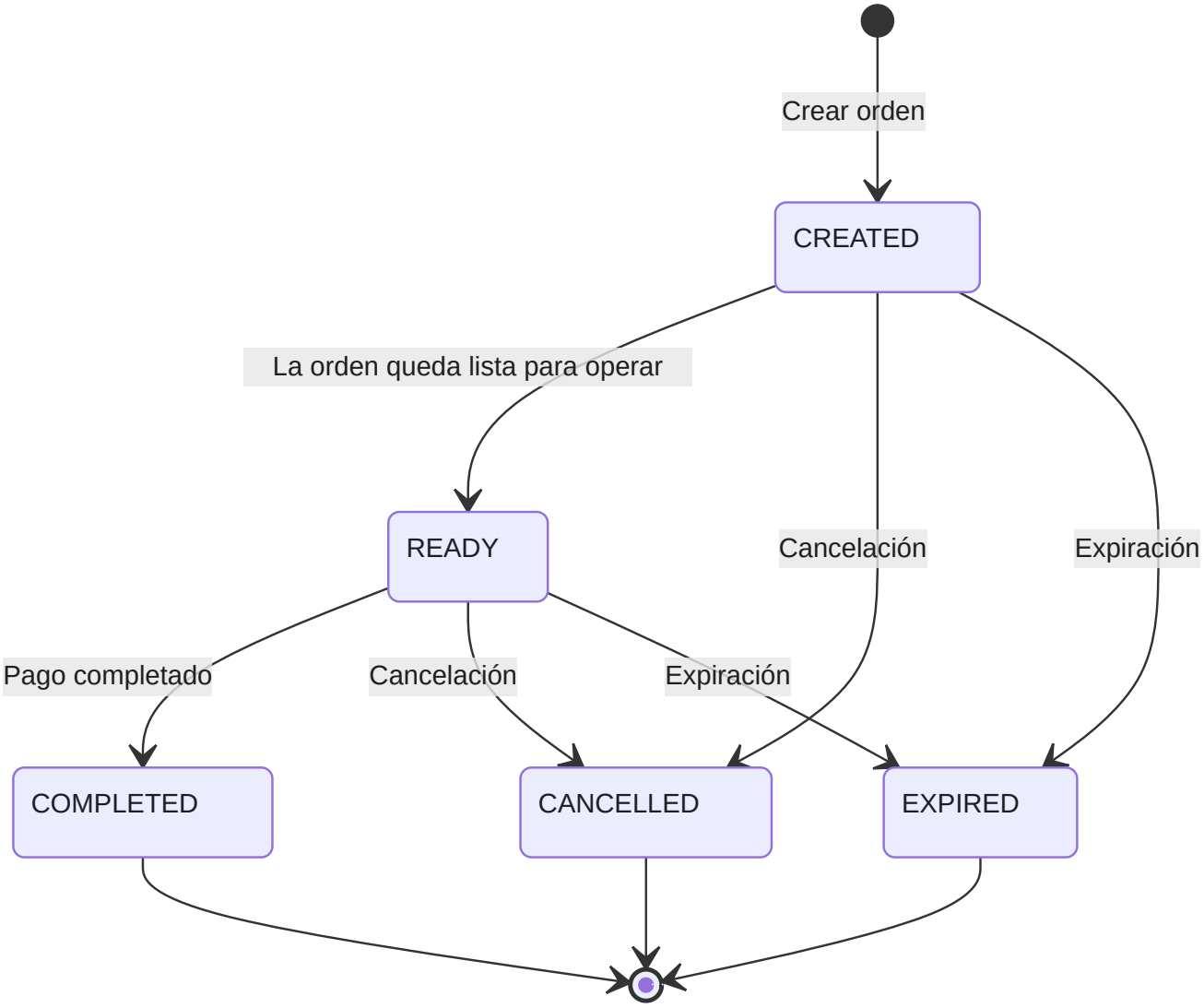
Ambientes

Ambiente	URL	Uso
Sandbox	`api.dev.pago46.io`	Desarrollo y pruebas
Producción	`api.prd.pago46.io`	Operaciones reales

→ Ver detalles de ambientes

Ciclo de vida de las órdenes

Como comercio, tu integración reacciona a los **estados finales** de la orden (`COMPLETED`, `CANCELLED`, `EXPIRED`), que reportamos vía webhooks. El procesamiento intermedio (cuando un proveedor toma la orden y cobra o entrega el efectivo) se resuelve de forma interna y no requiere acción de tu parte.



NOTIFICACIONES DE ÓRDENES EN MONEDA EXTRANJERA

Los comercios reciben webhooks cuando una orden alcanza un estado final. En **Pay-Out en moneda extranjera**, también recibirás dos notificaciones previas: una actualización `CREATED` que agrega la transacción que debes firmar, y un webhook `READY` cuando confirmamos que los fondos on-chain llegaron a Pago46 y la orden está lista para dispersarse en efectivo al usuario final.

¿QUIÉN PUEDE CANCELAR?

Los comercios **no** pueden cancelar órdenes. En **Pay-In** la cancelación la realiza el usuario final o el equipo de Pago46; en **Pay-Out** (moneda local y extranjera) **solo el equipo de Pago46** puede cancelar. Preferimos que las órdenes no atendidas expiren de forma natural.

Recibirás una notificación HTTP POST en tu `notify_url` cuando la orden alcance un estado final. El flujo de Pay-Out en moneda extranjera también envía las notificaciones `CREATED` y `READY` descritas arriba, por lo que en ese flujo puedes recibir más de una notificación con el mismo `status`.

Cobertura

Pago46 opera en múltiples países de Latinoamérica. Ver lista completa.

Pagos (Pay In)

Este módulo permite a los Comercios (Merchants) crear órdenes de pago (pay-in) para recibir fondos de sus usuarios o clientes. Las órdenes creadas quedan disponibles para ser procesadas por los Proveedores de Pago integrados en la red de Pago46.

ENDPOINT BASE

Todas las rutas descritas a continuación son relativas a la URL base de la API: `/api/v1`

CHECKOUT DE PAGO46

Las órdenes contienen el campo `redirect_url`, que corresponde al enlace de Checkout para continuar el flujo del usuario.

- Sandbox: `https://checkout.dev.pago46.io`
- Producción: `https://checkout.prd.pago46.io`
- Formato por orden: `https://checkout.{dev|prd}.pago46.io/{UUID}`

También puedes abrir la URL base sin UUID para ingresar manualmente el identificador de una orden durante las pruebas de integración en un iframe.

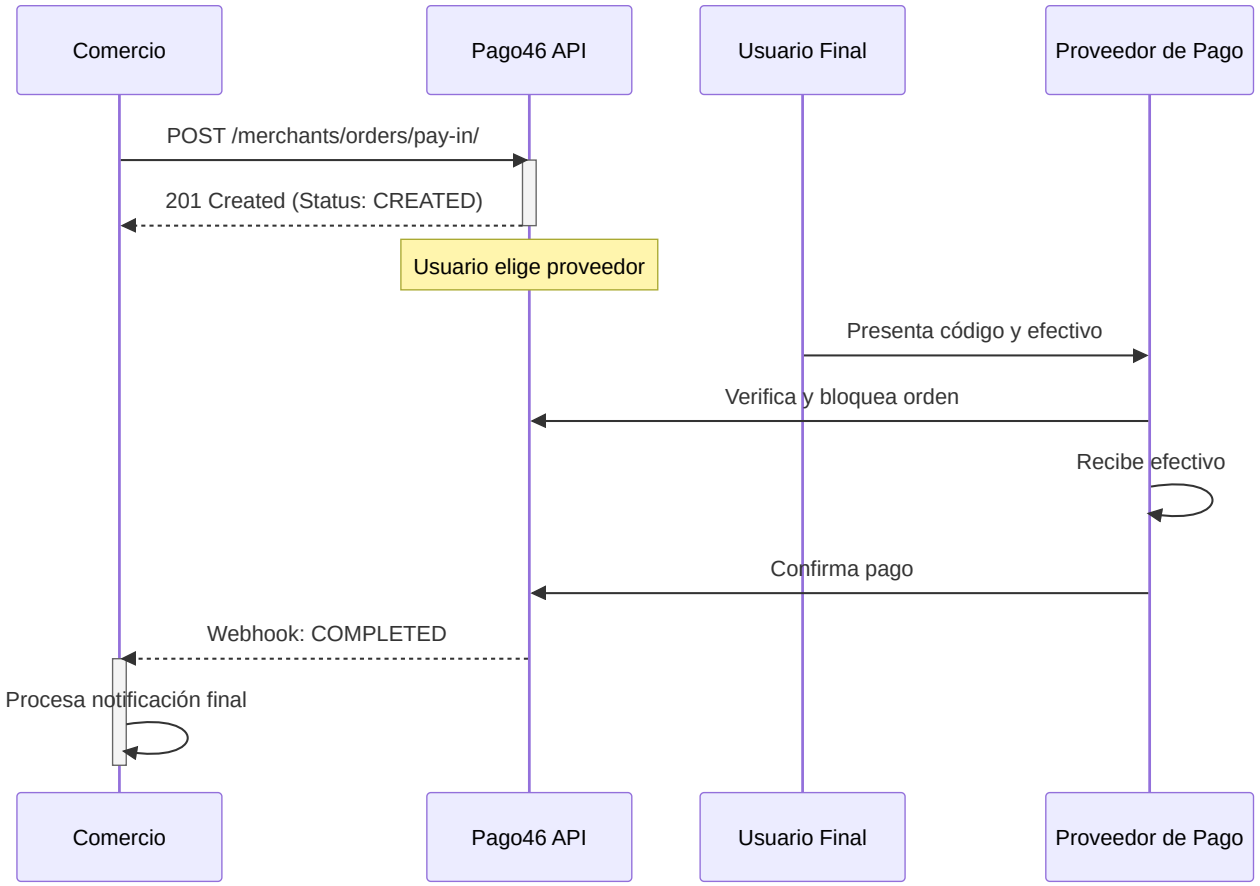
> Ver contenido incrustado en línea: https://drive.google.com/file/d/19XG84b29aGuBPA_VWpPrI1J4QknkosVe/preview

Más detalles en Checkout.

Flujo general

El proceso de pago para comercios se divide en tres etapas principales:

1. **Creación de la orden:** El comercio crea una orden de pago especificando el monto, país, datos del pagador y URLs de notificación.
2. **Procesamiento:** La orden es procesada por un Proveedor de Pago de la red, quien recibe el efectivo del usuario final.
3. **Notificaciones (webhooks):** El comercio recibe el estado final de la orden mediante el webhook configurado.



Requisitos previos

Antes de comenzar, asegúrate de tener:

- **Credenciales de API:** Tu `Merchant-Key` y `Merchant-Secret` proporcionados por Pago46.
- **Autenticación HMAC:** Familiarízate con el esquema de autenticación descrito en la sección de Autenticación.
- **Endpoint de Webhook:** Una URL pública HTTPS donde recibirás las notificaciones de cambios de estado.

> **SEGURIDAD**
>
> Todas las peticiones deben incluir las cabeceras de autenticación HMAC: `Merchant-Key`, `Message-Date` y `Message-Hash`.

---

1. Crear orden de pago

Para iniciar un cobro, debes crear una orden proporcionando la información del monto, país, pagador y configuración de notificaciones.

Endpoint: `POST /merchants/orders/pay-in/`

Parámetros de la petición

Campos obligatorios

Campo	Tipo	Descripción
`order_type`	String	Tipo de orden: `LocalCurrencyOrder`
`country`	String	Código ISO del país (ej: `MX`, `CL`, `AR`)
`price`	Decimal	Monto del pago (formato: `"1500.00"`)
`price_currency`	String	Código ISO de la moneda (ej: `MXN`, `CLP`, `ARS`)
`description`	String	Descripción de la transacción
`merchant_order_id`	String	ID único de tu sistema (max 127 caracteres)
`notify_url`	String (URL)	URL para recibir webhooks de cambios de estado
`return_url`	String (URL)	URL de retorno para el usuario
`expiry`	DateTime	Fecha y hora de expiración (formato ISO 8601)

Campos opcionales

Campo	Tipo	Descripción
`consumer_email`	String	Email del pagador
`consumer_phone_number`	String	Teléfono del pagador (max 128 caracteres)

> **DATOS DE CONTACTO**
>
> En Pay-In, si envías datos de contacto, debes enviar ambos campos, solo `consumer_email` o ninguno de los dos. No se acepta `consumer_phone_number` sin email. El soporte para teléfono sin email estará disponible próximamente.

> **LÍMITES POR PAÍS Y MONEDA**
>
> Tu cuenta debe estar habilitada para la combinación de `country` y `price_currency` que envías; de lo contrario recibes un `400` con el detalle `No matching configuration found for merchant, country, and currency`.
>
> Además, el `price` debe estar dentro de los montos **mínimo y máximo** configurados para tu cuenta en ese país y moneda. Fuera de rango, la API responde `400` con `Value is too low.` o `Value is too high.` en el campo `price`. Revisa los países y monedas disponibles; para tus límites, consulta a tu ejecutivo comercial.

Ejemplo de petición

```
curl -X POST "https://api.dev.pago46.io/api/v1/merchants/orders/pay-in/" \
  -H "Merchant-Key: <TU_MERCHANT_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>" \
  -H "Content-Type: application/json" \
  -d '{
    "order_type": "LocalCurrencyOrder",
    "country": "MX",
    "price": "1500.00",
    "price_currency": "MXN",
    "description": "Pago de suscripción - Usuario ABC123",
    "merchant_order_id": "ORDER-2024-001234",
```

```
    "notify_url": "https://tu-comercio.com/webhooks/pago46",
    "return_url": "https://tu-comercio.com/pago/volver",
    "consumer_email": "usuario@ejemplo.com",
    "consumer_phone_number": "+525512345678",
    "expiry": "2026-06-27T23:59:59Z"
  }'
```

Respuesta exitosa (201 Created)

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "order_type": "LocalCurrencyOrder",
  "country": "MX",
  "price": "1500.00",
  "price_currency": "MXN",
  "description": "Pago de suscripción - Usuario ABC123",
  "merchant_order_id": "ORDER-2024-001234",
  "status": "CREATED",
  "redirect_url": "https://checkout.dev.pago46.io/123e4567-e89b-12d3-a456-426614174000",
  "return_url": "https://tu-comercio.com/pago/volver",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "+525512345678",
  "expiry": "2026-06-27T23:59:59Z",
  "paid": null
}
```

GUARDA EL ID

Guarda el `id` de la orden devuelto en la respuesta. Lo necesitarás para consultar el estado de la orden posteriormente.

---

2. Consultar orden

Puedes consultar el estado actual de una orden en cualquier momento usando su ID.

Endpoint: `GET /merchants/orders/pay-in/{id}/`

Parámetros

Parámetro	Ubicación	Descripción
`id`	Path	UUID de la orden

Ejemplo de petición

```
curl -X GET "https://api.dev.pago46.io/api/v1/merchants/orders/pay-in/123e4567-e89b-12d3-a456-426614174000/" \
  -H "Merchant-Key: <TU_MERCHANT_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>"
```

Respuesta (200 OK)

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "order_type": "LocalCurrencyOrder",
  "country": "MX",
```

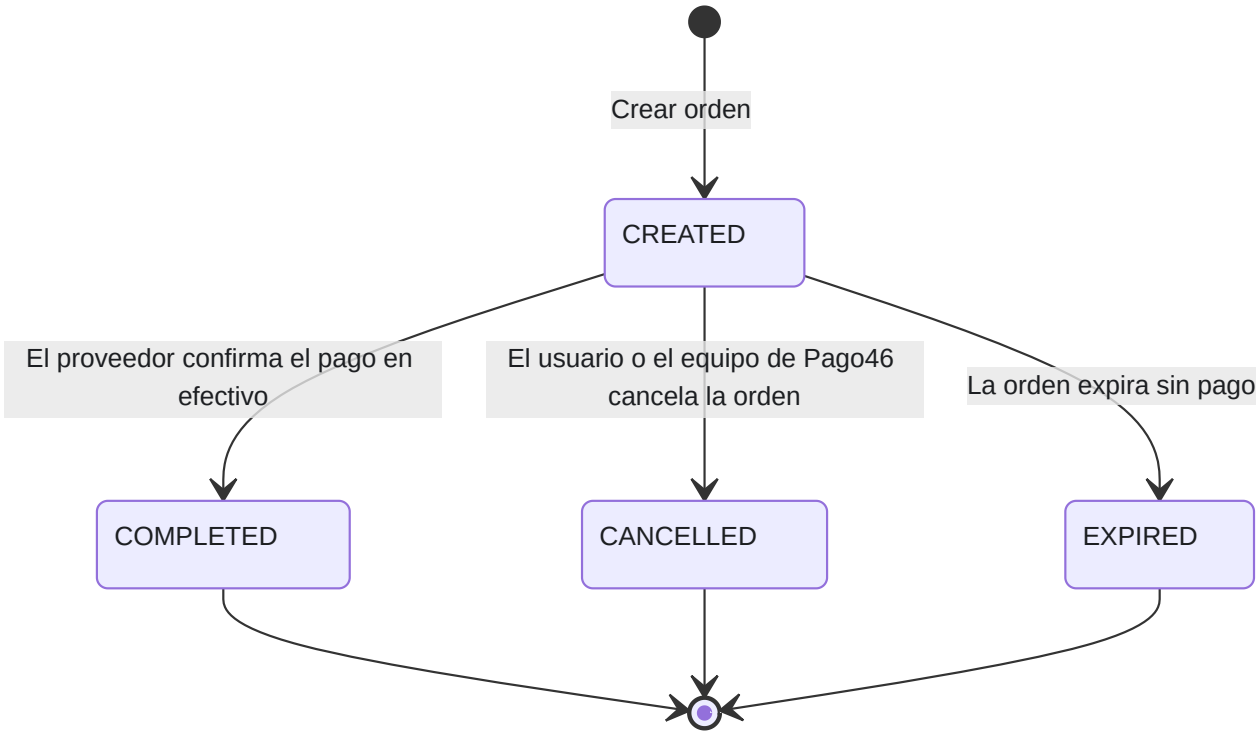
```
    "price": "1500.00",
    "price_currency": "MXN",
    "description": "Pago de suscripción - Usuario ABC123",
    "merchant_order_id": "ORDER-2024-001234",
    "status": "CREATED",
    "redirect_url": "https://checkout.dev.pago46.io/123e4567-e89b-12d3-a456-426614174000",
    "return_url": "https://tu-comercio.com/pago/volver",
    "notify_url": "https://tu-comercio.com/webhooks/pago46",
    "consumer_email": "usuario@ejemplo.com",
    "consumer_phone_number": "+525512345678",
    "expiry": "2026-06-27T23:59:59Z",
    "paid": null
}
```

Estados de la orden

Como comercio solo necesitas reaccionar a los **estados finales** (`COMPLETED` , `CANCELLED` , `EXPIRED`). Los estados intermedios se resuelven de forma interna y, en Pay-In, no se notifican.

Estado	¿Notificado por webhook?	¿Qué significa para el comercio?
`CREATED`	Respuesta al crear la orden	La orden se ha registrado y está pendiente de procesamiento
`READY`	No (interno en Pay-In)	La orden ya puede ser tomada por un proveedor
`PAYMENT_STARTED`	No (interno)	Paso intermedio: el usuario está realizando el pago físico en el punto
`COMPLETED`	Sí — estado final	El pago se completó exitosamente. El proveedor recibió el efectivo del usuario
`CANCELLED`	Sí — estado final	La orden fue cancelada (por el usuario o por el equipo de Pago46) y no se procesará
`EXPIRED`	Sí — estado final	La orden expiró sin pago y no se procesará

Diagrama de transición de estados



Webhooks (notificaciones)

Cuando la orden alcance un estado final (`COMPLETED`, `CANCELLED` o `EXPIRED`), Pago46 enviará una notificación HTTP POST a la URL especificada en el campo `notify_url`.

Estructura del webhook

Pago46 enviará el webhook con autenticación HMAC. Debes verificar la firma para asegurar que la notificación proviene de Pago46.

Cabeceras del webhook

```
POST /webhooks/pago46 HTTP/1.1
Host: tu-comercio.com
Content-Type: application/json
Merchant-Key: <TU_MERCHANT_KEY>
Message-Date: 1704463200.123
Message-Hash: a1b2c3d4e5f6...
```

Payload del webhook

Estado: COMPLETED

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "order_type": "LocalCurrencyOrder",
  "country": "MX",
  "price": "1500.00",
  "price_currency": "MXN",
  "description": "Pago de suscripción - Usuario ABC123",
  "merchant_order_id": "ORDER-2024-001234",
  "status": "COMPLETED",
```

```
  "redirect_url": "https://checkout.dev.pago46.io/123e4567-e89b-12d3-a456-426614174000",
  "return_url": "https://tu-comercio.com/pago/volver",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "+525512345678",
  "expiry": "2026-06-27T23:59:59Z",
  "paid": "2026-06-25T14:30:00Z"
}
```

Estado: CANCELLED

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "order_type": "LocalCurrencyOrder",
  "country": "MX",
  "price": "1500.00",
  "price_currency": "MXN",
  "description": "Pago de suscripción - Usuario ABC123",
  "merchant_order_id": "ORDER-2024-001234",
  "status": "CANCELLED",
  "redirect_url": "https://checkout.dev.pago46.io/123e4567-e89b-12d3-a456-426614174000",
  "return_url": "https://tu-comercio.com/pago/volver",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "+525512345678",
  "expiry": "2026-06-27T23:59:59Z",
  "paid": null
}
```

Verificación de webhooks

Debes verificar la autenticidad de cada webhook recibido para evitar procesar notificaciones fraudulentas.

Ejemplo de verificación en Python

```python
import hmac
import hashlib
import json
from flask import Flask, request, jsonify

app = Flask(__name__)

# Tu Merchant Secret (obtenido de Pago46)
MERCHANT_SECRET = "tu_merchant_secret_aqui"

@app.route('/webhooks/pago46', methods=['POST'])
def webhook_handler():
    # 1. Extraer headers
    merchant_key = request.headers.get('Merchant-Key')
    message_date = request.headers.get('Message-Date')
    received_hash = request.headers.get('Message-Hash')

    # 2. Obtener el body raw
    body_str = request.get_data(as_text=True)

    # 3. Construir string to sign
    # Formato: MERCHANT_KEY:MESSAGE_DATE:METHOD:PATH:BODY
    method = request.method  # "POST"
    path = request.path      # "/webhooks/pago46"
    string_to_sign = f"{merchant_key}:{message_date}:{method}:{path}:{body_str}"

    # 4. Calcular HMAC
    calculated_hash = hmac.new(
        MERCHANT_SECRET.encode('utf-8'),
        string_to_sign.encode('utf-8'),
        hashlib.sha256
    ).hexdigest()

    # 5. Verificar
    if not hmac.compare_digest(calculated_hash, received_hash):
        return jsonify({"error": "Invalid signature"}), 403
```

```
    # 6. Procesar la notificación
    order_data = json.loads(body_str)
    order_id = order_data.get('id')
    order_status = order_data.get('status')
    merchant_order_id = order_data.get('merchant_order_id')

    print(f"Orden {merchant_order_id} ({order_id}) cambió a estado: {order_status}")

    # Actualizar tu base de datos
    if order_status == 'COMPLETED':
        # Marcar como completada
        paid_at = order_data.get('paid')
        print(f"Pago completado el: {paid_at}")
        # Activar servicio, entregar producto, etc.
    elif order_status == 'CANCELLED':
        # Marcar como cancelada
        print("Pago cancelado")

    # 7. Responder con 200 OK
    return jsonify({"status": "received"}), 200

if __name__ == '__main__':
    app.run(port=5000)
```

Ejemplo de verificación en Node.js

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

const MERCHANT_SECRET = 'tu_merchant_secret_aqui';

app.post('/webhooks/pago46', express.text({ type: '*/*' }), (req, res) => {
  // 1. Extraer headers
  const merchantKey = req.headers['merchant-key'];
  const messageDate = req.headers['message-date'];
  const receivedHash = req.headers['message-hash'];

  // 2. Body como string
  const bodyStr = req.body;

  // 3. Construir string to sign
  const method = req.method;
  const path = req.path;
  const stringToSign = `${merchantKey}:${messageDate}:${method}:${path}:${bodyStr}`;

  // 4. Calcular HMAC
  const calculatedHash = crypto
    .createHmac('sha256', MERCHANT_SECRET)
    .update(stringToSign)
    .digest('hex');

  // 5. Verificar
  if (calculatedHash !== receivedHash) {
    return res.status(403).json({ error: 'Invalid signature' });
  }

  // 6. Procesar notificación
  const orderData = JSON.parse(bodyStr);
  const { id, status, merchant_order_id, paid } = orderData;

  console.log(`Orden ${merchant_order_id} (${id}) cambió a estado: ${status}`);

  if (status === 'COMPLETED') {
    console.log(`Pago completado el: ${paid}`);
    // Activar servicio, entregar producto, etc.
  } else if (status === 'CANCELLED') {
    console.log('Pago cancelado');
  }

  // 7. Responder
  res.status(200).json({ status: 'received' });
});

app.listen(5000, () => {
  console.log('Webhook server listening on port 5000');
```

```
});
```

RESPUESTA AL WEBHOOK

Debes responder con un código HTTP `200` o `201` para confirmar la recepción. Si no respondes exitosamente, reintentaremos el envío de la notificación.

Reintentamos ante cualquier respuesta `5XX`, `408` o `429`, y también ante timeouts o errores de conexión (si no recibimos respuesta), con backoff exponencial. En **producción**, hasta 600 intentos a lo largo de aproximadamente 4 días, con un máximo de 10 minutos entre intentos. En **sandbox**, solo 3 intentos en unos pocos segundos, siguiendo el mismo patrón de backoff.

Si la notificación de una orden ya completada falla de forma no reintentable (por ejemplo, se agotan los intentos), nos pondremos en contacto contigo a través de alguno de los canales de comunicación ya establecidos para ayudarte a resolver cualquier inconveniente que puedan estar teniendo tus usuarios con su orden.

Errores comunes

Consulta la sección Manejo de errores comunes para ejemplos y el formato estándar de respuestas de error para validaciones de límites, campos obligatorios y otros errores de negocio.

Validación de campos

Si envías datos inválidos o incompletos, recibirás un error `400 Bad Request` con detalles específicos:


```
{
    "type": "validation_error",
    "errors": [
        {
            "code": "invalid",
            "detail": "No matching configuration found for merchant, country, and currency.",
            "attr": "merchant_country_order_setting"
        },
        {
            "code": "required",
            "detail": "This field is required.",
            "attr": "country"
        },
        {
            "code": "required",
            "detail": "This field is required.",
            "attr": "price"
        },
        {
            "code": "required",
            "detail": "This field is required.",
            "attr": "price_currency"
        }
    ]
}
```


Tabla de errores HTTP

Código HTTP	Descripción	Solución
`400 Bad Request`	Datos inválidos o campos faltantes	Verifica que todos los campos obligatorios estén presentes y con el formato correcto

Código HTTP	Descripción	Solución
`403 Forbidden`	Autenticación fallida	Verifica tus credenciales y la firma HMAC
`404 Not Found`	Orden no encontrada	Verifica que el ID de la orden sea correcto
`422 Unprocessable Entity`	Error de lógica de negocio	Revisa el mensaje de error específico

---

Buenas prácticas

1. Identificador de orden

Asigna un `merchant_order_id` único por operación. La API aplica una restricción de unicidad sobre `(merchant_order_id, merchant)`: si creas otra orden con un `merchant_order_id` ya utilizado, la petición se rechaza en lugar de duplicar la orden. Úsalo para conciliar cada orden con tu sistema. Si no estás seguro de si una creación se completó, **consulta la orden** en lugar de reintentar el `POST`.

2. Manejo de webhooks

- **Procesa webhooks de forma asíncrona:** No bloquees la respuesta HTTP mientras procesas la lógica de negocio.
- **Implementa reintentos:** Si tu servidor webhook está caído, Pago46 reintentará el envío.
- **Valida siempre la firma HMAC:** Nunca confíes en webhooks sin verificar su autenticidad.

3. Expiración de órdenes

Configura `expiry` entre 24 y 72 horas después de crear la orden. Este rango reduce el riesgo de que un usuario conserve una captura del código e intente pagar cuando la operación ya no debería estar vigente. Evita órdenes con varios días o semanas de validez.

4. Monitoreo de órdenes

- Consulta las órdenes que no alcancen un estado final dentro del tiempo esperado.
- Implementa alertas para detectar órdenes pendientes por demasiado tiempo.
- Revisa la página de Estado del Servicio ante errores atípicos para confirmar si hay incidentes o mantenimientos activos.

5. URLs de notificación

- Usa URLs HTTPS para `notify_url`.
- Asegúrate de que el endpoint esté siempre disponible.
- Implementa registros para facilitar la depuración.

Ejemplo completo de integración

A continuación, un ejemplo completo en Python que muestra cómo crear una orden y manejar webhooks:

```
import hmac
import hashlib
import time
import requests
import json
from flask import Flask, request, jsonify
```

```python
# Configuración
API_BASE_URL = "https://api.dev.pago46.io"
MERCHANT_KEY = "tu_merchant_key"
MERCHANT_SECRET = "tu_merchant_secret"

app = Flask(__name__)

def generate_hmac(method, path, body_dict=None):
    """Genera la firma HMAC para autenticación"""
    timestamp = str(time.time())
    body_str = json.dumps(body_dict) if body_dict else ""

    string_to_sign = f"{MERCHANT_KEY}:{timestamp}:{method}:{path}:{body_str}"

    signature = hmac.new(
        MERCHANT_SECRET.encode('utf-8'),
        string_to_sign.encode('utf-8'),
        hashlib.sha256
    ).hexdigest()

    return {
        "Merchant-Key": MERCHANT_KEY,
        "Message-Date": timestamp,
        "Message-Hash": signature,
        "Content-Type": "application/json"
    }

def create_payin_order(amount, user_email, user_phone, merchant_order_id):
    """Crea una orden de pago"""
    path = "/api/v1/merchants/orders/pay-in/"

    order_data = {
        "order_type": "LocalCurrencyOrder",
        "country": "MX",
        "price": str(amount),
        "price_currency": "MXN",
        "description": f"Pago de usuario {user_email}",
        "merchant_order_id": merchant_order_id,
        "notify_url": "https://tu-comercio.com/webhooks/pago46",
        "return_url": "https://tu-comercio.com/pago/volver",
        "consumer_email": user_email,
        "consumer_phone_number": user_phone,
        "expiry": "2026-06-27T23:59:59Z"
    }

    headers = generate_hmac("POST", path, order_data)

    response = requests.post(
        f"{API_BASE_URL}{path}",
        headers=headers,
        json=order_data
    )

    if response.status_code == 201:
        order = response.json()
        print(f"Orden creada exitosamente: {order['id']}")
        return order
    else:
        print(f"Error al crear orden: {response.status_code}")
        print(response.text)
        return None

@app.route('/webhooks/pago46', methods=['POST'])
def webhook_handler():
    """Maneja webhooks de Pago46"""
    # Verificar firma HMAC
    merchant_key = request.headers.get('Merchant-Key')
    message_date = request.headers.get('Message-Date')
    received_hash = request.headers.get('Message-Hash')
    body_str = request.get_data(as_text=True)

    string_to_sign = f"{merchant_key}:{message_date}:{request.method}:{request.path}:{body_str}"
    calculated_hash = hmac.new(
        MERCHANT_SECRET.encode('utf-8'),
        string_to_sign.encode('utf-8'),
        hashlib.sha256
    ).hexdigest()

    if not hmac.compare_digest(calculated_hash, received_hash):
        return jsonify({"error": "Invalid signature"}), 403
```

```
    # Procesar webhook
    order = json.loads(body_str)

    print(f"Webhook recibido para orden: {order['merchant_order_id']}")
    print(f"   Estado: {order['status']}")

    # Los webhooks para comercios solo contienen estados finales
    if order['status'] == 'COMPLETED':
        print(f"   Pago completado el {order['paid']}")
        # Activar servicio, entregar producto, etc.
    elif order['status'] == 'CANCELLED':
        print("   Pago cancelado")
    elif order['status'] == 'EXPIRED':
        print("   Orden expirada")

    return jsonify({"status": "received"}), 200

if __name__ == '__main__':
    # Ejemplo: Crear una orden de pago
    order = create_payin_order(
        amount=1500.00,
        user_email="usuario@ejemplo.com",
        user_phone="+525512345678",
        merchant_order_id="ORDER-2024-" + str(int(time.time()))
    )

    # Iniciar servidor de webhooks
    print("\nIniciando servidor de webhooks...")
    app.run(port=5000)
```

Próximos pasos

- **Autenticación:** Revisa la guía de autenticación HMAC para entender el mecanismo de seguridad en detalle.
- **Ambientes:** Familiarízate con los ambientes de desarrollo y producción.

SOPORTE

Si tienes preguntas o necesitas ayuda con tu integración, contacta a tu ejecutivo comercial o escríbenos a contacto@pago46.com.

Retiros (Pay Out)

Este módulo permite a los Comercios (Merchants) crear órdenes de retiro (pay-out) para dispersar fondos a sus usuarios o clientes. Las órdenes creadas quedan disponibles para ser procesadas por los Proveedores de Pago integrados en la red de Pago46.

ENDPOINT BASE

Todas las rutas descritas a continuación son relativas a la URL base de la API: `/api/v1`

CHECKOUT DE PAGO46

Las órdenes contienen el campo `redirect_url`, que corresponde al enlace de Checkout para continuar el flujo del usuario.

- Sandbox: `https://checkout.dev.pago46.io`
- Producción: `https://checkout.prd.pago46.io`
- Formato por orden: `https://checkout.{dev|prd}.pago46.io/{UUID}`

También puedes abrir la URL base sin UUID para ingresar manualmente el identificador de una orden durante las pruebas de integración en un iframe.

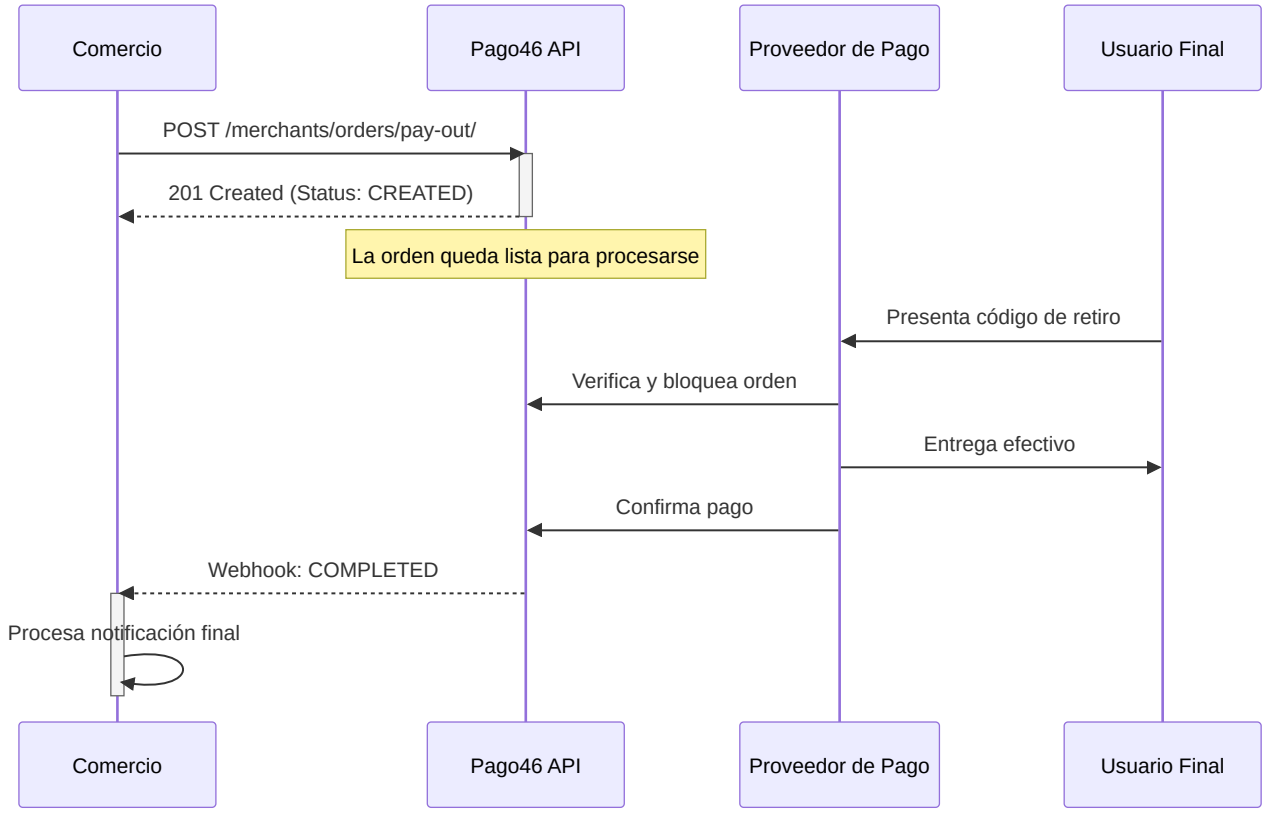
> Ver contenido incrustado en línea: https://drive.google.com/file/d/1iSXarQQocBATZjLsEGcDEajYLfvm6Aon/preview

Más detalles en Checkout.

Flujo general

El proceso de retiro para comercios se divide en tres etapas principales:

1. **Creación de la orden:** El comercio crea una orden de retiro especificando el monto, país, datos del beneficiario y URLs de notificación.
2. **Procesamiento:** La orden es procesada por un Proveedor de Pago de la red, quien entrega el efectivo al usuario final.
3. **Notificaciones (webhooks):** El comercio recibe el estado final de la orden mediante el webhook configurado.



Requisitos previos

Antes de comenzar, asegúrate de tener:

- **Credenciales de API:** Tu `Merchant-Key` y `Merchant-Secret` proporcionados por Pago46.
- **Autenticación HMAC:** Familiarízate con el esquema de autenticación descrito en la sección de Autenticación.
- **Endpoint de Webhook:** Una URL pública HTTPS donde recibirás las notificaciones de cambios de estado.

> **SEGURIDAD**
>
> Todas las peticiones deben incluir las cabeceras de autenticación HMAC: `Merchant-Key`, `Message-Date` y `Message-Hash`.

---

1. Crear orden de retiro

Para iniciar un retiro, debes crear una orden proporcionando la información del monto, país, beneficiario y configuración de notificaciones.

Endpoint: `POST /merchants/orders/pay-out/`

Parámetros de la petición

Campos obligatorios

Campo	Tipo	Descripción
`order_type`	String	Tipo de orden: `LocalCurrencyOrder`
`country`	String	Código ISO del país (ej: `MX`, `CL`, `AR`)
`price`	Decimal	Monto del retiro (formato: `"1500.00"`)
`price_currency`	String	Código ISO de la moneda (ej: `MXN`, `CLP`, `ARS`)
`description`	String	Descripción de la transacción
`merchant_order_id`	String	ID único de tu sistema (max 127 caracteres)
`notify_url`	String (URL)	URL para recibir webhooks de cambios de estado
`return_url`	String (URL)	URL de retorno para el usuario
`expiry`	DateTime	Fecha y hora de expiración (formato ISO 8601)

Campos opcionales

Campo	Tipo	Descripción
`consumer_email`	String	Email del beneficiario
`consumer_phone_number`	String	Teléfono del beneficiario (max 128 caracteres)

> **⚠ CONTACTO DEL BENEFICIARIO**
>
> En Pay-Out debes enviar **al menos uno** de `consumer_email` o `consumer_phone_number`.

> **RETIROS EN MONEDA EXTRANJERA**
>
> Para retiros internacionales o con conversión de divisas, consulta la documentación de Retiros en moneda extranjera.

> **LÍMITES POR PAÍS Y MONEDA**
>
> Tu cuenta debe estar habilitada para la combinación de `country` y `price_currency` que envías; de lo contrario recibes un `400` con el detalle `No matching configuration found for merchant, country, and currency`.
>
> Además, el `price` debe estar dentro de los montos **mínimo y máximo** configurados para tu cuenta en ese país y moneda. Fuera de rango, la API responde `400` con `Value is too low.` o `Value is too high.` en el campo `price`. Revisa los países y monedas disponibles; para tus límites, consulta a tu ejecutivo comercial.

Ejemplo de petición

```
curl -X POST "https://api.dev.pago46.io/api/v1/merchants/orders/pay-out/" \
  -H "Merchant-Key: <TU_MERCHANT_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>" \
  -H "Content-Type: application/json" \
  -d '{
    "order_type": "LocalCurrencyOrder",
```

```
    "country": "MX",
    "price": "1500.00",
    "price_currency": "MXN",
    "description": "Retiro de saldo - Usuario ABC123",
    "merchant_order_id": "ORDER-2024-001234",
    "notify_url": "https://tu-comercio.com/webhooks/pago46",
    "return_url": "https://tu-comercio.com/retiro/volver",
    "consumer_email": "usuario@ejemplo.com",
    "consumer_phone_number": "+525512345678",
    "expiry": "2026-06-27T23:59:59Z"
  }'
```

Respuesta exitosa (201 Created)

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "order_type": "LocalCurrencyOrder",
  "country": "MX",
  "price": "1500.00",
  "price_currency": "MXN",
  "description": "Retiro de saldo - Usuario ABC123",
  "merchant_order_id": "ORDER-2024-001234",
  "status": "CREATED",
  "redirect_url": "https://checkout.dev.pago46.io/123e4567-e89b-12d3-a456-426614174000",
  "return_url": "https://tu-comercio.com/retiro/volver",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "+525512345678",
  "expiry": "2026-06-27T23:59:59Z",
  "paid": null
}
```

GUARDA EL ID

Guarda el `id` de la orden devuelto en la respuesta. Lo necesitarás para consultar el estado de la orden posteriormente.

---

2. Consultar orden

Puedes consultar el estado actual de una orden en cualquier momento usando su ID.

Endpoint: `GET /merchants/orders/pay-out/{id}/`

Parámetros

Parámetro	Ubicación	Descripción
`id`	Path	UUID de la orden

Ejemplo de petición

```
curl -X GET "https://api.dev.pago46.io/api/v1/merchants/orders/pay-out/123e4567-e89b-12d3-a456-426614174000/" \
  -H "Merchant-Key: <TU_MERCHANT_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>"
```

Respuesta (200 OK)

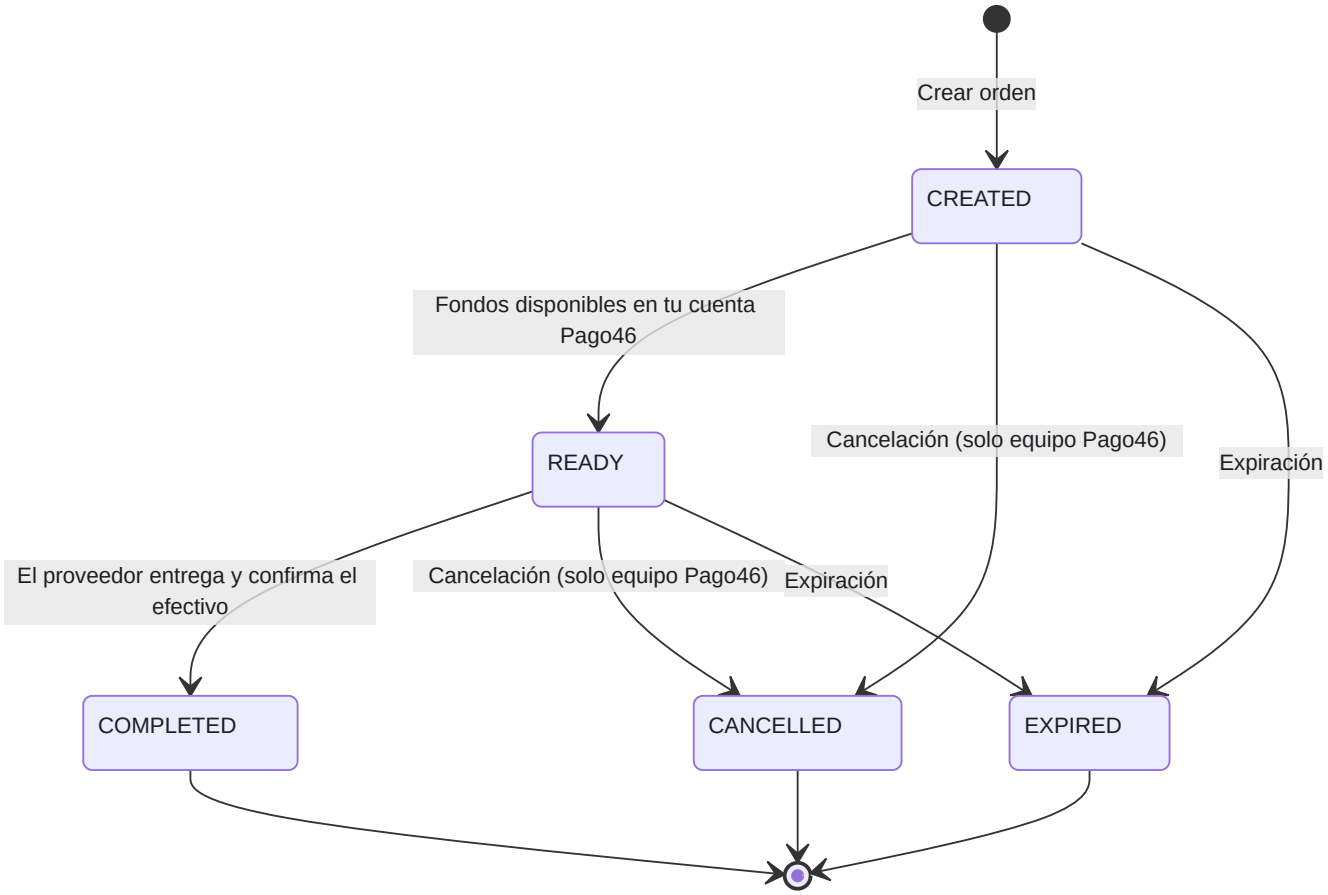
```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "order_type": "LocalCurrencyOrder",
  "country": "MX",
  "price": "1500.00",
  "price_currency": "MXN",
  "description": "Retiro de saldo - Usuario ABC123",
  "merchant_order_id": "ORDER-2024-001234",
  "status": "READY",
  "redirect_url": "https://checkout.dev.pago46.io/123e4567-e89b-12d3-a456-426614174000",
  "return_url": "https://tu-comercio.com/retiro/volver",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "+525512345678",
  "expiry": "2026-06-27T23:59:59Z",
  "paid": null
}
```

Estados de la orden

Como comercio solo recibes webhooks para los estados finales (`COMPLETED` , `CANCELLED` y `EXPIRED`). Los estados intermedios se resuelven de forma interna.

Estado	¿Notificado por webhook?	¿Qué significa para el comercio?
`CREATED`	Respuesta al crear la orden	La orden se ha registrado; pendiente de fondos disponibles
`READY`	No (interno)	Hay fondos disponibles en tu cuenta Pago46; la orden puede avanzar
`PAYMENT_STARTED`	No (interno)	Paso intermedio: un proveedor está entregando el efectivo al beneficiario
`COMPLETED`	Sí — estado final	El retiro se completó exitosamente. El beneficiario recibió el efectivo
`CANCELLED`	Sí — estado final	La orden fue cancelada por el equipo de Pago46 (ni el comercio ni el usuario pueden cancelar) y no se procesará
`EXPIRED`	Sí — estado final	La orden expiró y no se procesará

Diagrama de transición de estados



Webhooks (notificaciones)

Cuando la orden alcance un estado final (`COMPLETED`, `CANCELLED` o `EXPIRED`), Pago46 enviará una notificación HTTP POST a la URL especificada en el campo `notify_url`.

Estructura del webhook

Pago46 enviará el webhook con autenticación HMAC. Debes verificar la firma para asegurar que la notificación proviene de Pago46.

Cabeceras del webhook

```
POST /webhooks/pago46 HTTP/1.1
Host: tu-comercio.com
Content-Type: application/json
Merchant-Key: <TU_MERCHANT_KEY>
Message-Date: 1704463200.123
Message-Hash: a1b2c3d4e5f6...
```

Payload del webhook

Estado: COMPLETED

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "order_type": "LocalCurrencyOrder",
```

```
  "country": "MX",
  "price": "1500.00",
  "price_currency": "MXN",
  "description": "Retiro de saldo - Usuario ABC123",
  "merchant_order_id": "ORDER-2024-001234",
  "status": "COMPLETED",
  "redirect_url": "https://checkout.dev.pago46.io/123e4567-e89b-12d3-a456-426614174000",
  "return_url": "https://tu-comercio.com/retiro/volver",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "+525512345678",
  "expiry": "2026-06-27T23:59:59Z",
  "paid": "2026-06-25T14:30:00Z"
}
```

Estado: CANCELLED

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "order_type": "LocalCurrencyOrder",
  "country": "MX",
  "price": "1500.00",
  "price_currency": "MXN",
  "description": "Retiro de saldo - Usuario ABC123",
  "merchant_order_id": "ORDER-2024-001234",
  "status": "CANCELLED",
  "redirect_url": "https://checkout.dev.pago46.io/123e4567-e89b-12d3-a456-426614174000",
  "return_url": "https://tu-comercio.com/retiro/volver",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "+525512345678",
  "expiry": "2026-06-27T23:59:59Z",
  "paid": null
}
```

Verificación de webhooks

Debes verificar la autenticidad de cada webhook recibido para evitar procesar notificaciones fraudulentas.

Ejemplo de verificación en Python

```
import hmac
import hashlib
import json
from flask import Flask, request, jsonify

app = Flask(__name__)

# Tu Merchant Secret (obtenido de Pago46)
MERCHANT_SECRET = "tu_merchant_secret_aqui"

@app.route('/webhooks/pago46', methods=['POST'])
def webhook_handler():
    # 1. Extraer headers
    merchant_key = request.headers.get('Merchant-Key')
    message_date = request.headers.get('Message-Date')
    received_hash = request.headers.get('Message-Hash')

    # 2. Obtener el body raw
    body_str = request.get_data(as_text=True)

    # 3. Construir string to sign
    # Formato: MERCHANT_KEY:MESSAGE_DATE:METHOD:PATH:BODY
    method = request.method  # "POST"
    path = request.path      # "/webhooks/pago46"
    string_to_sign = f"{merchant_key}:{message_date}:{method}:{path}:{body_str}"

    # 4. Calcular HMAC
    calculated_hash = hmac.new(
        MERCHANT_SECRET.encode('utf-8'),
        string_to_sign.encode('utf-8'),
        hashlib.sha256
```

```
    ).hexdigest()

    # 5. Verificar
    if not hmac.compare_digest(calculated_hash, received_hash):
        return jsonify({"error": "Invalid signature"}), 403

    # 6. Procesar la notificación
    order_data = json.loads(body_str)
    order_id = order_data.get('id')
    order_status = order_data.get('status')
    merchant_order_id = order_data.get('merchant_order_id')

    print(f"Orden {merchant_order_id} ({order_id}) cambió a estado: {order_status}")

    # Actualizar tu base de datos
    if order_status == 'COMPLETED':
        # Marcar como completada
        paid_at = order_data.get('paid')
        print(f"Retiro completado el: {paid_at}")
    elif order_status == 'CANCELLED':
        # Marcar como cancelada
        print("Retiro cancelado")

    # 7. Responder con 200 OK
    return jsonify({"status": "received"}), 200

if __name__ == '__main__':
    app.run(port=5000)
```

Ejemplo de verificación en Node.js

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

const MERCHANT_SECRET = 'tu_merchant_secret_aqui';

app.post('/webhooks/pago46', express.text({ type: '*/*' }), (req, res) => {
  // 1. Extraer headers
  const merchantKey = req.headers['merchant-key'];
  const messageDate = req.headers['message-date'];
  const receivedHash = req.headers['message-hash'];

  // 2. Body como string
  const bodyStr = req.body;

  // 3. Construir string to sign
  const method = req.method;
  const path = req.path;
  const stringToSign = `${merchantKey}:${messageDate}:${method}:${path}:${bodyStr}`;

  // 4. Calcular HMAC
  const calculatedHash = crypto
    .createHmac('sha256', MERCHANT_SECRET)
    .update(stringToSign)
    .digest('hex');

  // 5. Verificar
  if (calculatedHash !== receivedHash) {
    return res.status(403).json({ error: 'Invalid signature' });
  }

  // 6. Procesar notificación
  const orderData = JSON.parse(bodyStr);
  const { id, status, merchant_order_id, paid } = orderData;

  console.log(`Orden ${merchant_order_id} (${id}) cambió a estado: ${status}`);

  if (status === 'COMPLETED') {
    console.log(`Retiro completado el: ${paid}`);
    // Actualizar base de datos
  } else if (status === 'CANCELLED') {
    console.log('Retiro cancelado');
    // Actualizar base de datos
  }

  // 7. Responder
  res.status(200).json({ status: 'received' });
```

```
});

app.listen(5000, () => {
  console.log('Webhook server listening on port 5000');
});
```

⚠ RESPUESTA AL WEBHOOK

Debes responder con un código HTTP `200` o `201` para confirmar la recepción. Si no respondes exitosamente, reintentaremos el envío de la notificación.

Reintentamos ante cualquier respuesta `5XX`, `408` o `429`, y también ante timeouts o errores de conexión (si no recibimos respuesta), con backoff exponencial. En **producción**, hasta 600 intentos a lo largo de aproximadamente 4 días, con un máximo de 10 minutos entre intentos. En **sandbox**, solo 3 intentos en unos pocos segundos, siguiendo el mismo patrón de backoff.

Si la notificación de una orden ya completada falla de forma no reintentable (por ejemplo, se agotan los intentos), nos pondremos en contacto contigo a través de alguno de los canales de comunicación ya establecidos para ayudarte a resolver cualquier inconveniente que puedan estar teniendo tus usuarios con su orden.

Errores comunes

Consulta la sección Manejo de errores comunes para ejemplos y el formato estándar de respuestas de error para validaciones de límites, campos obligatorios y otros errores de negocio.

Validación de campos

Si envías datos inválidos o incompletos, recibirás un error `400 Bad Request` estructurado como se muestra en la sección "Manejo de errores comunes". Allí encontrarás ejemplos detallados de respuestas típicas y explicaciones de cada campo del error.

Ejemplo de un posible error por campos requeridos:

```
{
  "type": "validation_error",
  "errors": [
    {
      "attr": "country",
      "code": "required",
      "detail": "This field is required."
    },
    {
      "attr": "price",
      "code": "invalid",
      "detail": "A valid number is required."
    },
    {
      "attr": "expiry",
      "code": "invalid",
      "detail": "Datetime has wrong format. Use one of these formats instead: YYYY-MM-DDThh:mm[:ss[.uuuuuu]][+HH:MM|-
HH:MM|Z]."
    }
  ]
}
```

Tabla de errores HTTP

Código HTTP	Descripción	Solución
`400 Bad Request`	Datos inválidos o campos faltantes	Verifica que todos los campos obligatorios estén presentes y con el formato correcto
`403 Forbidden`	Autenticación fallida	Verifica tus credenciales y la firma HMAC
`404 Not Found`	Orden no encontrada	Verifica que el ID de la orden sea correcto
`422 Unprocessable Entity`	Error de lógica de negocio	Revisa el mensaje de error específico

---

Buenas prácticas

1. Identificador de orden

Asigna un `merchant_order_id` único por operación. La API aplica una restricción de unicidad sobre `(merchant_order_id, merchant)`: si creas otra orden con un `merchant_order_id` ya utilizado, la petición se rechaza en lugar de duplicar la orden. Úsalo para conciliar cada orden con tu sistema. Si no estás seguro de si una creación se completó, **consulta la orden** en lugar de reintentar el `POST`.

2. Manejo de webhooks

- **Procesa webhooks de forma asíncrona:** No bloquees la respuesta HTTP mientras procesas la lógica de negocio.
- **Implementa reintentos:** Si tu servidor webhook está caído, Pago46 reintentará el envío.
- **Valida siempre la firma HMAC:** Nunca confíes en webhooks sin verificar su autenticidad.

3. Expiración de órdenes

Configura `expiry` entre 24 y 72 horas después de crear la orden. Este rango reduce el riesgo de que un usuario conserve una captura del código e intente retirar efectivo cuando la operación ya no debería estar vigente. Evita órdenes con varios días o semanas de validez.

4. Monitoreo de órdenes

- Consulta las órdenes que no alcancen un estado final dentro del tiempo esperado.
- Implementa alertas para detectar órdenes pendientes por demasiado tiempo.
- Revisa la página de Estado del Servicio ante errores atípicos para confirmar si hay incidentes o mantenimientos activos.

5. URLs de notificación

- Usa URLs HTTPS para `notify_url`.
- Asegúrate de que el endpoint esté siempre disponible.
- Implementa registros para facilitar la depuración.

Ejemplo completo de integración

A continuación, un ejemplo completo en Python que muestra cómo crear una orden y manejar webhooks:

```python
import hmac
import hashlib
import time
import requests
import json
from flask import Flask, request, jsonify

# Configuración
API_BASE_URL = "https://api.dev.pago46.io"
MERCHANT_KEY = "tu_merchant_key"
MERCHANT_SECRET = "tu_merchant_secret"

app = Flask(__name__)

def generate_hmac(method, path, body_dict=None):
    """Genera la firma HMAC para autenticación"""
    timestamp = str(time.time())
    body_str = json.dumps(body_dict) if body_dict else ""

    string_to_sign = f"{MERCHANT_KEY}:{timestamp}:{method}:{path}:{body_str}"

    signature = hmac.new(
        MERCHANT_SECRET.encode('utf-8'),
        string_to_sign.encode('utf-8'),
        hashlib.sha256
    ).hexdigest()

    return {
        "Merchant-Key": MERCHANT_KEY,
        "Message-Date": timestamp,
        "Message-Hash": signature,
        "Content-Type": "application/json"
    }

def create_payout_order(amount, user_email, user_phone, merchant_order_id):
    """Crea una orden de retiro"""
    path = "/api/v1/merchants/orders/pay-out/"

    order_data = {
        "order_type": "LocalCurrencyOrder",
        "country": "MX",
        "price": str(amount),
        "price_currency": "MXN",
        "description": f"Retiro de usuario {user_email}",
        "merchant_order_id": merchant_order_id,
        "notify_url": "https://tu-comercio.com/webhooks/pago46",
        "return_url": "https://tu-comercio.com/retiro/volver",
        "consumer_email": user_email,
        "consumer_phone_number": user_phone,
        "expiry": "2026-06-27T23:59:59Z"
    }

    headers = generate_hmac("POST", path, order_data)

    response = requests.post(
        f"{API_BASE_URL}{path}",
        headers=headers,
        json=order_data
    )

    if response.status_code == 201:
        order = response.json()
        print(f"Orden creada exitosamente: {order['id']}")
        return order
    else:
        print(f"Error al crear orden: {response.status_code}")
        print(response.text)
        return None

@app.route('/webhooks/pago46', methods=['POST'])
def webhook_handler():
    """Maneja webhooks de Pago46"""
    # Verificar firma HMAC
    merchant_key = request.headers.get('Merchant-Key')
    message_date = request.headers.get('Message-Date')
    received_hash = request.headers.get('Message-Hash')
    body_str = request.get_data(as_text=True)

    string_to_sign = f"{merchant_key}:{message_date}:{request.method}:{request.path}:{body_str}"
    calculated_hash = hmac.new(
```

```
        MERCHANT_SECRET.encode('utf-8'),
        string_to_sign.encode('utf-8'),
        hashlib.sha256
    ).hexdigest()

    if not hmac.compare_digest(calculated_hash, received_hash):
        return jsonify({"error": "Invalid signature"}), 403

    # Procesar webhook
    order = json.loads(body_str)

    print(f"Webhook recibido para orden: {order['merchant_order_id']}")
    print(f"   Estado: {order['status']}")

    # Los webhooks para comercios solo contienen estados finales
    if order['status'] == 'COMPLETED':
        print(f"   Retiro completado el {order['paid']}")
        # Actualizar saldo del usuario, enviar notificación, etc.
    elif order['status'] == 'CANCELLED':
        print("   Retiro cancelado")
        # Reembolsar saldo, notificar usuario, etc.
    elif order['status'] == 'EXPIRED':
        print("   Orden expirada")

    return jsonify({"status": "received"}), 200

if __name__ == '__main__':
    # Ejemplo: Crear una orden de retiro
    order = create_payout_order(
        amount=1500.00,
        user_email="usuario@ejemplo.com",
        user_phone="+525512345678",
        merchant_order_id="ORDER-2024-" + str(int(time.time()))
    )

    # Iniciar servidor de webhooks
    print("\nIniciando servidor de webhooks...")
    app.run(port=5000)
```

Próximos pasos

- **Retiros en moneda extranjera:** Consulta la documentación de Retiros en moneda extranjera para aprender a crear órdenes internacionales.
- **Autenticación:** Revisa la guía de autenticación HMAC para entender el mecanismo de seguridad en detalle.
- **Ambientes:** Familiarízate con los ambientes de desarrollo y producción.

SOPORTE

Si tienes preguntas o necesitas ayuda con tu integración, contacta a tu ejecutivo comercial o escríbenos a contacto@pago46.com.

Retiros en moneda extranjera (Pay Out)

Este módulo permite crear retiros (pay-out) en moneda local usando fondos de origen en moneda extranjera (por ejemplo, USDC → MXN).

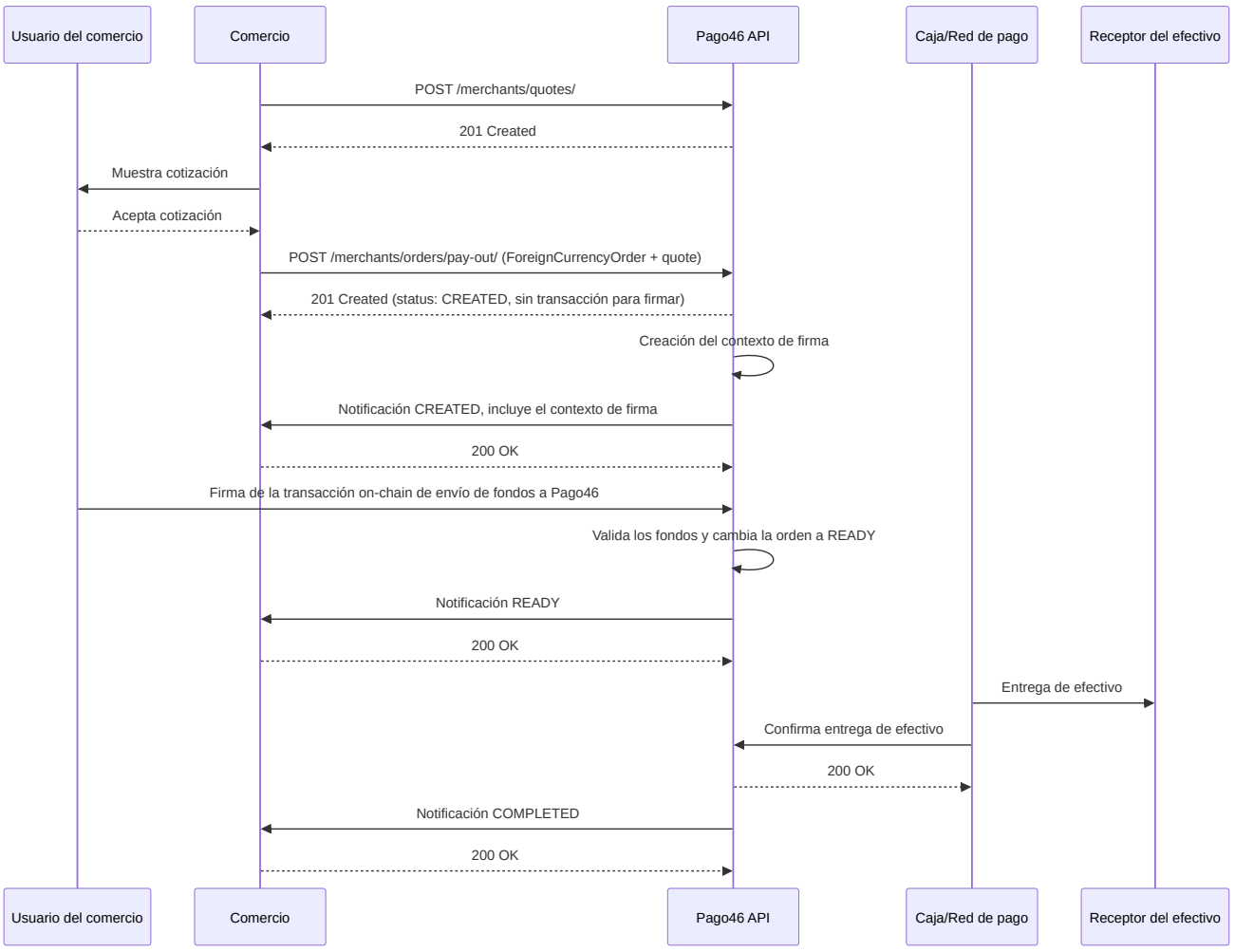
Como Comercio, tu integración habla exclusivamente con Pago46. Pago46 gestiona internamente la cotización, el seguimiento de la recepción de fondos y la habilitación del retiro en efectivo.

Consulta los pares y países disponibles y firma cada petición según la guía de Autenticación.

> **ENDPOINT BASE**
>
> Todas las rutas descritas a continuación son relativas a la URL base de la API: `/api/v1`

Flujo general

1. Creas una cotización para un par de activos (`POST /merchants/quotes/`).
2. Tu usuario acepta la cotización en tu aplicación.
3. Creas una orden de retiro en moneda extranjera asociada a esa cotización.
4. Tu usuario envía los fondos a la dirección (address) indicada por Pago46.
5. Si el fondeo llega dentro de la ventana válida, la orden avanza a `READY` y queda lista para continuar el flujo normal de retiro.



Nota: la generación y entrega de la transacción blockchain ocurre en segundo plano, nunca en la respuesta inmediata del POST. Siempre espera el webhook antes de continuar con la firma.

Requisitos previos

- Credenciales de Comercio (`Merchant-Key` y `Merchant-Secret`).
- Firma HMAC en cada petición (`Message-Date`, `Message-Hash`).
- `notify_url` pública por HTTPS para cambios de estado.
- Al menos uno entre `consumer_email` o `consumer_phone_number` al crear la orden.

Revisa la sección de Autenticación para la firma.

---

1) Crear cotización

Solicita una cotización para el par de activos deseado.

Endpoint: `POST /merchants/quotes/`

Campos de la petición

Campo	Tipo	Requerido	Descripción
`blockchain`	String	Sí	Red de fondeo (`SOLANA`, `STELLAR`, `MONAD`).
`send`	Decimal	Sí	Monto que enviará el usuario.
`send_currency`	String	Sí	Moneda de origen (ej. `USDC`).
`target_country`	String	Sí	País de destino (`MX`, `CL`, etc.).
`target_currency`	String	Sí	Moneda local objetivo (ej. `MXN`).

Ejemplo de petición

```
curl -X POST "https://api.dev.pago46.io/api/v1/merchants/quotes/" \
  -H "Merchant-Key: <TU_MERCHANT_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>" \
  -H "Content-Type: application/json" \
  -d '{
    "blockchain": "SOLANA",
    "send": "150.00",
    "send_currency": "USDC",
    "target_country": "MX",
    "target_currency": "MXN"
  }'
```

Respuesta (201 Created)

```
{
  "id": "01952d8f-8da1-7f4b-bb36-cfba8a3be829",
  "blockchain": "SOLANA",
  "send": "150.00",
  "send_currency": "USDC",
```

```
  "target_country": "MX",
  "target_currency": "MXN",
  "fee": "2.00",
  "fee_currency": "USDC",
  "order_price": "2895.00",
  "order_price_currency": "MXN",
  "rate": "19.56",
  "expires": "2030-01-01T11:05:00Z"
}
```

CONSEJO

Guarda el `id` de la cotización. Lo usarás al crear la orden en moneda extranjera.

VALIDACIÓN DE LA COTIZACIÓN

La creación de la cotización puede fallar con `400` si:

- tu cuenta no está configurada para el país o la moneda de destino (`No matching configuration found for merchant, country, and currency`),
- el par no es válido para la blockchain (`<send_currency>:<target_currency> is not a valid pair for <blockchain> blockchain`),
- el monto a enviar es demasiado bajo (`The amount to send is too low.`).

PROTECCIÓN DE TIPO DE CAMBIO (`424`)

Si en ese momento no podemos asegurar un tipo de cambio justo, la cotización responde `424` con `A dependent operation failed.`. Es una protección, no un error de tu integración: solo emitimos la cotización cuando la tasa cumple nuestras garantías, de modo que tu usuario nunca queda con una tasa desfavorable.

Es una condición **temporal** y puede afectar a una blockchain en particular mientras la liquidez se normaliza. Si la recibes, **reintenta en unos minutos**; si una red sigue respondiendo `424`, ofrece a tu usuario otra de las blockchains soportadas (`SOLANA`, `STELLAR`, `MONAD`) como ruta alternativa.

---

2) Crear orden de retiro en moneda extranjera

Cuando tu usuario acepte la cotización, crea la orden de pay-out en moneda extranjera usando el `quote`.

Endpoint: `POST /merchants/orders/pay-out/`

Campos de la petición

Campo	Tipo	Requerido	Descripción
`order_type`	String	Sí	Debe ser `ForeignCurrencyOrder`.
`quote`	UUID	Sí	ID de la cotización creada previamente.
`description`	String	Sí	Descripción de la transacción.
`merchant_order_id`	String	Sí	ID único de tu sistema (único por comercio).

Campo	Tipo	Requerido	Descripción
`notify_url`	URL	Sí	URL para notificaciones de estado.
`return_url`	URL	Sí	URL de retorno.
`expiry`	DateTime	Sí	Expiración de la orden (ISO 8601).
`consumer_email`	String	Condicional	Requerido si no envías teléfono.
`consumer_phone_number`	String	Condicional	Requerido si no envías email.
`wallet`	String	Sí	Wallet que firmará la transacción on-chain.

Ejemplo de petición

```
curl -X POST "https://api.dev.pago46.io/api/v1/merchants/orders/pay-out/" \
  -H "Merchant-Key: <TU_MERCHANT_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>" \
  -H "Content-Type: application/json" \
  -d '{
    "quote": "01952d8f-8da1-7f4b-bb36-cfba8a3be829",
    "description": "Retiro de efectivo desde saldo en USDC",
    "merchant_order_id": "FX-PO-2026-0001",
    "notify_url": "https://tu-comercio.com/webhooks/pago46",
    "return_url": "https://tu-comercio.com/retiro/volver",
    "consumer_email": "usuario@ejemplo.com",
    "expiry": "2030-01-03T12:05:00Z",
    "wallet": "7EcDhSYGxXyscszYEp35KHN8vvw3svAuLKTzXwCFLtV",
    "order_type": "ForeignCurrencyOrder"
  }'
```

⚠ IMPORTANTE

Este POST devuelve **201 Created**:

- La orden ha sido creada, pero **NO INCLUYE** la transacción blockchain por firmar.
- Pago46 genera la transacción de forma asíncrona.
- Cuando la transacción on-chain esté lista, recibirás un webhook de actualización de la orden: el `status` seguirá siendo `CREATED`, pero ahora el payload incluirá el campo `transaction` con los datos para firmar.
- Es decir, puedes recibir múltiples webhooks con el mismo `status` (`CREATED`) pero con distinto contenido; considera este webhook como el evento de **"transacción lista"**.
- **¡No intentes firmar ni mostrar datos de blockchain al usuario hasta recibir este webhook!**

Después de crear la orden y recibir el 201 Created, tu integración debe esperar la notificación asincrónica por webhook. Solo cuando llegue el webhook con la transacción, el usuario podrá firmar on-chain.

Respuesta inmediata del POST (sin transacción):

```
{
  "id": "01952d91-a0ff-7f57-8f4e-68d95be01122",
  "order_type": "ForeignCurrencyOrder",
  "country": "MX",
  "price": "2895.00",
  "price_currency": "MXN",
  "description": "Retiro de efectivo desde saldo en USDC",
  "merchant_order_id": "FX-PO-2026-0001",
  "status": "CREATED",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
```

```
  "redirect_url": "https://checkout.dev.pago46.io/01952d91-a0ff-7f57-8f4e-68d95be01122",
  "return_url": "https://tu-comercio.com/retiro/volver",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "",
  "expiry": "2030-01-03T12:05:00Z",
  "paid": null,
  "transaction": "",
  "quote": "01952d8f-8da1-7f4b-bb36-cfba8a3be829",
  "wallet": "7EcDhSYGxXyscszYEp35KHN8vvw3svAuLKTzXwCFLtV"
}
```

Ejemplo de webhook: aquí la orden sí incluye el campo `transaction`:


```
{
  "id": "01952d91-a0ff-7f57-8f4e-68d95be01122",
  "order_type": "ForeignCurrencyOrder",
  "country": "MX",
  "price": "2895.00",
  "price_currency": "MXN",
  "description": "Retiro de efectivo desde saldo en USDC",
  "merchant_order_id": "FX-PO-2026-0001",
  "status": "CREATED",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
  "redirect_url": "https://checkout.dev.pago46.io/01952d91-a0ff-7f57-8f4e-68d95be01122",
  "return_url": "https://tu-comercio.com/retiro/volver",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "",
  "expiry": "2030-01-03T12:05:00Z",
  "paid": null,
  "quote": "01952d8f-8da1-7f4b-bb36-cfba8a3be829",
  "transaction":
"AgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADyTBSt8EsV++3pX0uVJVEfDjDp8mnxoqBQocHm
  "wallet": "7EcDhSYGxXyscszYEp35KHN8vvw3svAuLKTzXwCFLtV"
}
```


---

3) Firmar y enviar la transacción on-chain

Cuando llega el webhook con el campo `transaction`, ese valor es la transacción **sin firmar** que Pago46 ya construyó. El formato depende de la red de la cotización (base64, XDR, JSON, etc.) — revisa el detalle de cada red más abajo. Tu usuario debe firmarla con la `wallet` que indicaste al crear la orden y enviarla a la red para depositar los fondos en la dirección de Pago46.

El procedimiento es el estándar de cada red; no necesitas lógica propia de Pago46:

1. **Decodifica y analiza** el campo `transaction` según el formato de la red (ver detalle abajo).
2. **Deserialízalo** con el SDK de la blockchain correspondiente.
3. **Fírmalo** con la `wallet` del usuario.
4. **Transmite la transacción** a la red y espera la confirmación.

Cuando la transacción se confirma on-chain y Pago46 valida la recepción de los fondos, la orden avanza a `READY` y recibirás un webhook. Este estado confirma que los fondos on-chain llegaron a Pago46 dentro de la ventana esperada y que la orden está lista para dispersarse en efectivo al usuario final.

Según la red de la cotización:

- **Solana** — la transacción es una `VersionedTransaction`. Deserialízala desde base64, fírmala con el keypair de la wallet y envíala con un RPC de Solana. Ver @solana/web3.js y la documentación de Solana.
- **Stellar** — la transacción es un sobre XDR (`TransactionEnvelope`). Decodifícalo, fírmalo con la clave de la cuenta y envíalo a Horizon. Ver el SDK de Stellar para JS y la documentación de Stellar.

- **Monad** — es compatible con EVM. El campo `transaction` llega como un objeto JSON con los parámetros de la transacción (no en base64). Pásalo a tu librería habitual (ethers.js o viem) para firmarla y enviarla como cualquier transacción de Ethereum. Ver la documentación de Monad.

> **SANDBOX USA TESTNET; PRODUCCIÓN USA MAINNET**
>
> **Sandbox** opera sobre la **testnet** de las tres blockchains (Solana, Stellar y Monad); **Producción** sobre **mainnet**. Apunta la wallet y el RPC a la red correcta según el ambiente. Revisa los ambientes para las URLs base de cada uno.

> **TOKEN DE PRUEBA EN SANDBOX**
>
> En Sandbox usamos un token de prueba **no minteable** para los activos de origen (por ejemplo, `USDC`). Para probar en testnet necesitas que te entreguemos saldo de ese token: solicítalo a tu ejecutivo comercial o escríbenos a contacto@pago46.com.

---

4) Consultar orden

Puedes consultar la orden de retiro cuando necesites verificar su estado.

Endpoint: `GET /merchants/orders/pay-out/{id}/`

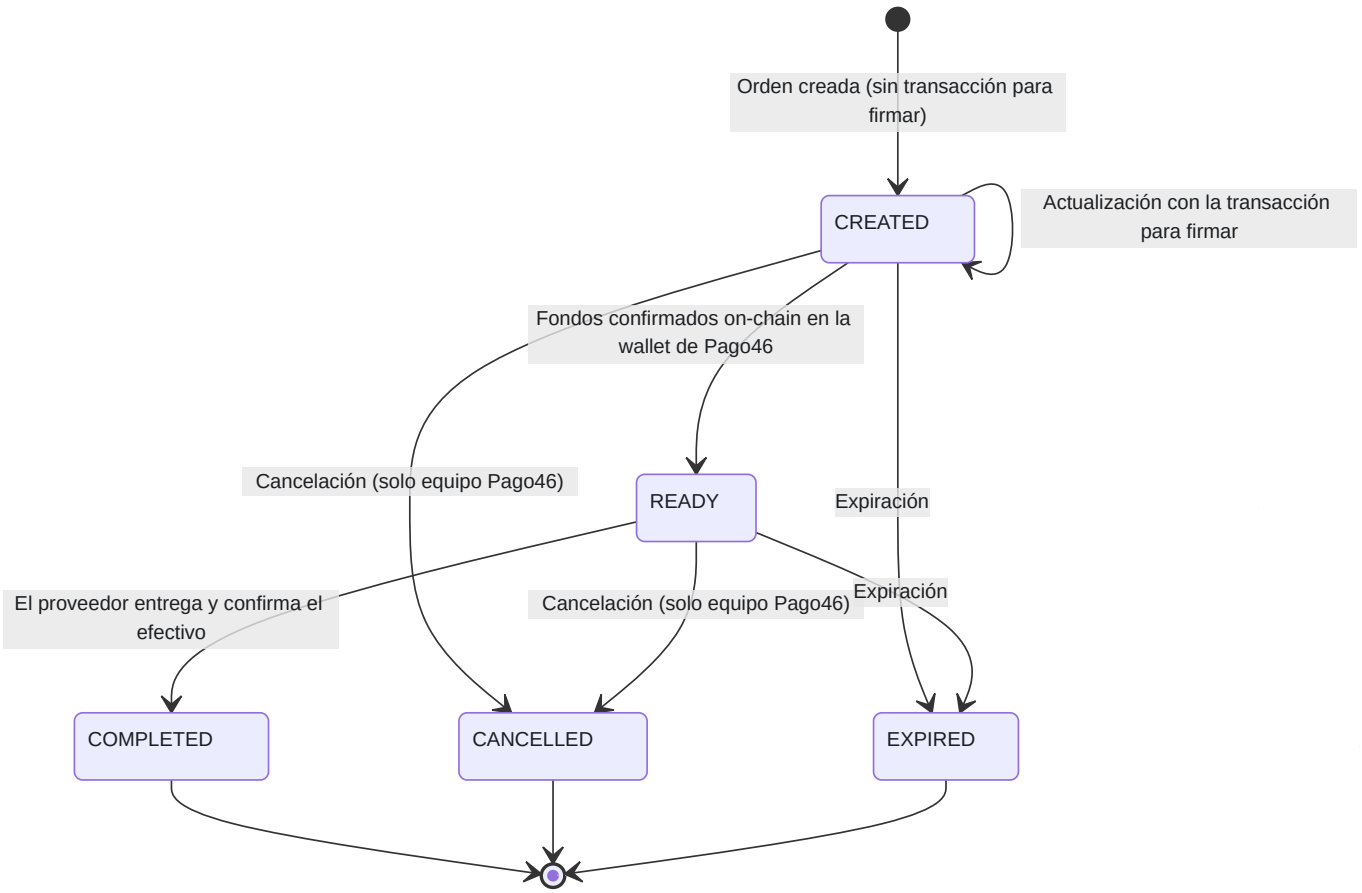
```
curl -X GET "https://api.dev.pago46.io/api/v1/merchants/orders/pay-out/01952d91-a0ff-7f57-8f4e-68d95be01122/" \
  -H "Merchant-Key: <TU_MERCHANT_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>"
```

Estados de la orden (Pay-Out en moneda extranjera)

Como comercio recibes webhooks cuando la orden alcanza un estado final. Además, recibes una actualización `CREATED` con la transacción que debes firmar y un webhook `READY` cuando confirmamos los fondos on-chain. El paso intermedio de entrega del efectivo se resuelve de forma interna.

Estado	¿Notificado por webhook?	¿Qué significa para el comercio?
`CREATED`	Respuesta y actualización	La respuesta inicial registra la orden sin `transaction`; una actualización posterior, aún con estado `CREATED`, incluye la información que debes firmar
`READY`	Sí	Confirmamos la recepción de los fondos on-chain en la wallet de Pago46; la orden está lista para dispersarse en efectivo
`PAYMENT_STARTED`	No (interno)	Paso intermedio: un proveedor está entregando el efectivo al beneficiario
`COMPLETED`	Sí — estado final	El retiro se completó exitosamente. El beneficiario recibió el efectivo
`CANCELLED`	Sí — estado final	La orden fue cancelada por el equipo de Pago46 (ni el comercio ni el usuario pueden cancelar) y no se procesará
`EXPIRED`	Sí — estado final	La orden expiró y no se procesará

Diagrama de transición de estados



Webhooks (notificaciones)

A lo largo de este flujo recibirás varias notificaciones por webhook a la `notify_url` indicada al crear la orden: la actualización `CREATED` con la transacción para firmar, `READY` al confirmar los fondos on-chain, y el estado final (`COMPLETED`, `CANCELLED` o `EXPIRED`). Consulta la tabla de la sección anterior para el detalle de cada una.

Estructura del webhook

Pago46 enviará el webhook con autenticación HMAC. Debes verificar la firma para asegurar que la notificación proviene de Pago46.

Cabeceras del webhook

```
POST /webhooks/pago46 HTTP/1.1
Host: tu-comercio.com
Content-Type: application/json
Merchant-Key: <TU_MERCHANT_KEY>
Message-Date: 1704463200.123
Message-Hash: a1b2c3d4e5f6...
```

Payload del webhook

Estado: COMPLETED

```
{
  "id": "01952d91-a0ff-7f57-8f4e-68d95be01122",
```

```
  "order_type": "ForeignCurrencyOrder",
  "country": "MX",
  "price": "2895.00",
  "price_currency": "MXN",
  "description": "Retiro de efectivo desde saldo en USDC",
  "merchant_order_id": "FX-PO-2026-0001",
  "status": "COMPLETED",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
  "redirect_url": "https://checkout.dev.pago46.io/01952d91-a0ff-7f57-8f4e-68d95be01122",
  "return_url": "https://tu-comercio.com/retiro/volver",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "",
  "expiry": "2030-01-03T12:05:00Z",
  "paid": "2030-01-01T12:10:00Z",
  "quote": "01952d8f-8da1-7f4b-bb36-cfba8a3be829",
  "transaction":
"AgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADyTBSt8EsV++3pX0uVJVEfDjDp8mnxoqBQocHm
  "wallet": "7EcDhSYGxXyscszYEp35KHN8vvw3svAuLKTzXwCFLtV"
}
```


Estado: CANCELLED


```
{
  "id": "01952d91-a0ff-7f57-8f4e-68d95be01122",
  "order_type": "ForeignCurrencyOrder",
  "country": "MX",
  "price": "2895.00",
  "price_currency": "MXN",
  "description": "Retiro de efectivo desde saldo en USDC",
  "merchant_order_id": "FX-PO-2026-0001",
  "status": "CANCELLED",
  "notify_url": "https://tu-comercio.com/webhooks/pago46",
  "redirect_url": "https://checkout.dev.pago46.io/01952d91-a0ff-7f57-8f4e-68d95be01122",
  "return_url": "https://tu-comercio.com/retiro/volver",
  "consumer_email": "usuario@ejemplo.com",
  "consumer_phone_number": "",
  "expiry": "2030-01-03T12:05:00Z",
  "paid": null,
  "quote": "01952d8f-8da1-7f4b-bb36-cfba8a3be829",
  "transaction":
"AgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADyTBSt8EsV++3pX0uVJVEfDjDp8mnxoqBQocHm
  "wallet": "7EcDhSYGxXyscszYEp35KHN8vvw3svAuLKTzXwCFLtV"
}
```


Verificación de webhooks

Debes verificar la autenticidad de cada webhook recibido para evitar procesar notificaciones fraudulentas.

Ejemplo de verificación en Python

```python
import hmac
import hashlib
import json
from flask import Flask, request, jsonify

app = Flask(__name__)

# Tu Merchant Secret (obtenido de Pago46)
MERCHANT_SECRET = "tu_merchant_secret_aqui"

@app.route('/webhooks/pago46', methods=['POST'])
def webhook_handler():
    # 1. Extraer headers
    merchant_key = request.headers.get('Merchant-Key')
    message_date = request.headers.get('Message-Date')
    received_hash = request.headers.get('Message-Hash')

    # 2. Obtener el body raw
    body_str = request.get_data(as_text=True)

    # 3. Construir string to sign
    # Formato: MERCHANT_KEY:MESSAGE_DATE:METHOD:PATH:BODY
```

```
    method = request.method  # "POST"
    path = request.path      # "/webhooks/pago46"
    string_to_sign = f"{merchant_key}:{message_date}:{method}:{path}:{body_str}"

    # 4. Calcular HMAC
    calculated_hash = hmac.new(
        MERCHANT_SECRET.encode('utf-8'),
        string_to_sign.encode('utf-8'),
        hashlib.sha256
    ).hexdigest()

    # 5. Verificar
    if not hmac.compare_digest(calculated_hash, received_hash):
        return jsonify({"error": "Invalid signature"}), 403

    # 6. Procesar la notificación
    order_data = json.loads(body_str)
    order_id = order_data.get('id')
    order_status = order_data.get('status')
    merchant_order_id = order_data.get('merchant_order_id')

    print(f"Orden {merchant_order_id} ({order_id}) cambió a estado: {order_status}")

    # Actualizar tu base de datos
    if order_status == 'COMPLETED':
        # Marcar como completada
        paid_at = order_data.get('paid')
        print(f"Retiro completado el: {paid_at}")
    elif order_status == 'CANCELLED':
        # Marcar como cancelada
        print("Retiro cancelado")

    # 7. Responder con 200 OK
    return jsonify({"status": "received"}), 200

if __name__ == '__main__':
    app.run(port=5000)
```

Ejemplo de verificación en Node.js

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

const MERCHANT_SECRET = 'tu_merchant_secret_aqui';

app.post('/webhooks/pago46', express.text({ type: '*/*' }), (req, res) => {
  // 1. Extraer headers
  const merchantKey = req.headers['merchant-key'];
  const messageDate = req.headers['message-date'];
  const receivedHash = req.headers['message-hash'];

  // 2. Body como string
  const bodyStr = req.body;

  // 3. Construir string to sign
  const method = req.method;
  const path = req.path;
  const stringToSign = `${merchantKey}:${messageDate}:${method}:${path}:${bodyStr}`;

  // 4. Calcular HMAC
  const calculatedHash = crypto
    .createHmac('sha256', MERCHANT_SECRET)
    .update(stringToSign)
    .digest('hex');

  // 5. Verificar
  if (calculatedHash !== receivedHash) {
    return res.status(403).json({ error: 'Invalid signature' });
  }

  // 6. Procesar notificación
  const orderData = JSON.parse(bodyStr);
  const { id, status, merchant_order_id, paid } = orderData;

  console.log(`Orden ${merchant_order_id} (${id}) cambió a estado: ${status}`);

  if (status === 'COMPLETED') {
```

```
    console.log(`Retiro completado el: ${paid}`);
    // Actualizar base de datos
  } else if (status === 'CANCELLED') {
    console.log('Retiro cancelado');
    // Actualizar base de datos
  }

  // 7. Responder
  res.status(200).json({ status: 'received' });
});

app.listen(5000, () => {
  console.log('Webhook server listening on port 5000');
});
```

⚠ RESPUESTA AL WEBHOOK

Debes responder con un código HTTP `200` o `201` para confirmar la recepción. Si no respondes exitosamente, reintentaremos el envío de la notificación.

Reintentamos ante cualquier respuesta `5XX`, `408` o `429`, y también ante timeouts o errores de conexión (si no recibimos respuesta), con backoff exponencial. En **producción**, hasta 600 intentos a lo largo de aproximadamente 4 días, con un máximo de 10 minutos entre intentos. En **sandbox**, solo 3 intentos en unos pocos segundos, siguiendo el mismo patrón de backoff.

Si la notificación de una orden ya completada falla de forma no reintentable (por ejemplo, se agotan los intentos), nos pondremos en contacto contigo a través de alguno de los canales de comunicación ya establecidos para ayudarte a resolver cualquier inconveniente que puedan estar teniendo tus usuarios con su orden.

Buenas prácticas de integración

- Usa un `merchant_order_id` idempotente por operación de negocio.
- Maneja cambios de estado de forma tolerante a reintentos.
- No asumas `COMPLETED` inmediatamente después de crear la orden.
- Presenta al usuario una cuenta regresiva según la ventana de cotización.
- **Después del POST, mantén un estado "Esperando transacción" hasta recibir el webhook con el campo `transaction`.**
- Almacena el `id` de la orden devuelta para correlacionar el POST inicial y el webhook subsiguiente.
- Ante errores de validación, solicita una nueva cotización antes de reintentar. Consulta la sección Manejo de errores comunes para más detalles sobre el formato de los mensajes de error devueltos por la API.
- **Implementa reintentos:** Si tu servidor webhook está caído, Pago46 reintentará el envío.

Si necesitas soporte para nuevos pares (por ejemplo, activos de origen o países adicionales), contacta a tu ejecutivo comercial o escríbenos a contacto@pago46.com.

Checkout

Checkout es la aplicación web de Pago46 diseñada para integrarse en un iframe en tu experiencia de usuario. Desde ahí, tus usuarios pueden completar flujos de Pay-In y Pay-Out con una interfaz optimizada para dispositivos móviles y computadoras de escritorio.

¿Qué es Checkout?

Checkout estandariza la experiencia final del usuario en un solo punto, independientemente del tipo de operación o canal de acceso. Esto te ayuda a:

- reducir tiempos de implementación para nuevos flujos,
- mantener una UX consistente en web y móvil,
- acelerar pruebas de punta a punta,
- simplificar soporte operativo con una ruta de ejecución común.

Internamente, Checkout incluye monitoreo técnico y seguimiento de rendimiento que respaldan el soporte operativo del día a día.

Ambientes

Ambiente	URL base
Sandbox	`https://checkout.dev.pago46.io`
Producción	`https://checkout.prd.pago46.io`

Formato por orden:

- Sandbox: `https://checkout.dev.pago46.io/{UUID}`
- Producción: `https://checkout.prd.pago46.io/{UUID}`

También puedes abrir la URL base (sin UUID) e ingresar el UUID de forma manual.

`redirect_url` llega en la respuesta de creación/consulta de la orden y apunta directamente a la URL de Checkout para esa orden.

Integración en iframe

Flujo recomendado:

1. Crea la orden en Pago46.
2. Toma `redirect_url` de la respuesta de la orden.
3. Carga ese `redirect_url` en un iframe dentro de tu aplicación.
4. Para las pruebas de aseguramiento de calidad (QA), valida también el ingreso manual del UUID desde la página principal de Checkout.

Ejemplo base de iframe:

```
<iframe
  src="https://checkout.dev.pago46.io/01952d91-a0ff-7f57-8f4e-68d95be01122"
  width="100%"
  height="720"
  style="border: 0"
```

```
  allow="clipboard-write"
></iframe>
```

Recomendaciones puntuales para un iframe web:

- Usa ancho completo y altura suficiente para evitar un desplazamiento interno excesivo.
- No bloquees cookies, almacenamiento ni navegación dentro del dominio de Checkout.
- Permite `clipboard-write` y capacidades estándar del navegador cuando aplique.
- Maneja expiración de sesión y recargas sin romper el estado de tu pantalla.
- Si haces pruebas manuales, abre también la URL base e ingresa el UUID sin usar una URL prearmada.

Integración en apps móviles

Checkout se integra como WebView, sin SDK adicional.

App móvil con WebView

- Carga `redirect_url` directamente en el WebView.
- Evita interceptar navegación de Checkout salvo casos estrictamente necesarios.
- Mantén un User-Agent estable para evitar diferencias de render inesperadas.

App nativa (iOS/Android)

- Integra Checkout como WebView embebido en el flujo de retiro o cobro.
- Controla el botón Atrás para no cerrar el flujo accidentalmente.
- Define una política clara para abrir enlaces externos fuera del WebView.

App React Native

- Integra Checkout en un WebView (sin SDK propietario).
- Reutiliza el mismo contrato: recibir `redirect_url` y cargarlo en la vista.
- Prueba ciclo completo en foreground/background para validar continuidad.

Permisos del navegador

Checkout usa el portapapeles para que el usuario copie códigos de pago. Si lo embebes en un iframe, habilita `clipboard-write` (ver el ejemplo de arriba). También requiere una conexión a internet estable: el estado de la orden se actualiza por sondeo periódico mientras la pantalla está abierta.

Flujo de pruebas recomendado

1. Crea una orden en sandbox.
2. Toma `redirect_url` y cárgalo en un iframe o WebView.
3. Ejecuta el flujo completo con la orden pre-cargada.
4. Repite entrando a la página principal de Checkout e ingresando el UUID manualmente.
5. Valida comportamiento en desktop, Android e iOS.

Demos

Android - Pay-In

Ver contenido incrustado en línea: https://drive.google.com/file/d/19XG84b29aGuBPA_VWpPrI1J4QknkosVe/preview

Android - Pay-Out

Ver contenido incrustado en línea: https://drive.google.com/file/d/1iSXarQQocBATZjLsEGcDEajYLfvm6Aon/preview

iOS

Ver contenido incrustado en línea: https://drive.google.com/file/d/1Dy4FT7jsURd4IItC0MKwaOFUdeZG4dcU/preview

Desktop - Pay-In

Ver contenido incrustado en línea: https://drive.google.com/file/d/1mhqbKd0hJ0yt9YpZrglx5ORVV8M8umiY/preview

Desktop - Pay-Out

Ver contenido incrustado en línea: https://drive.google.com/file/d/1dD6Ma2HMlS9_X3YZVJf9D84wG2XIRZVU/preview

Integración para Proveedores de Pago

Como Proveedor de Pago, integras tu red de puntos físicos (tiendas, kioscos, agentes) a Pago46 para procesar cobros y dispersiones de efectivo generados por comercios.

Servicios disponibles

Pay-In (Cobros)

Procesa pagos en efectivo de usuarios finales en tus puntos físicos.

Flujo: El usuario presenta el código → validas y bloqueas la orden → recibes el efectivo → confirmas la transacción.

Ver documentación de Pay-In

Pay-Out (Dispersiones)

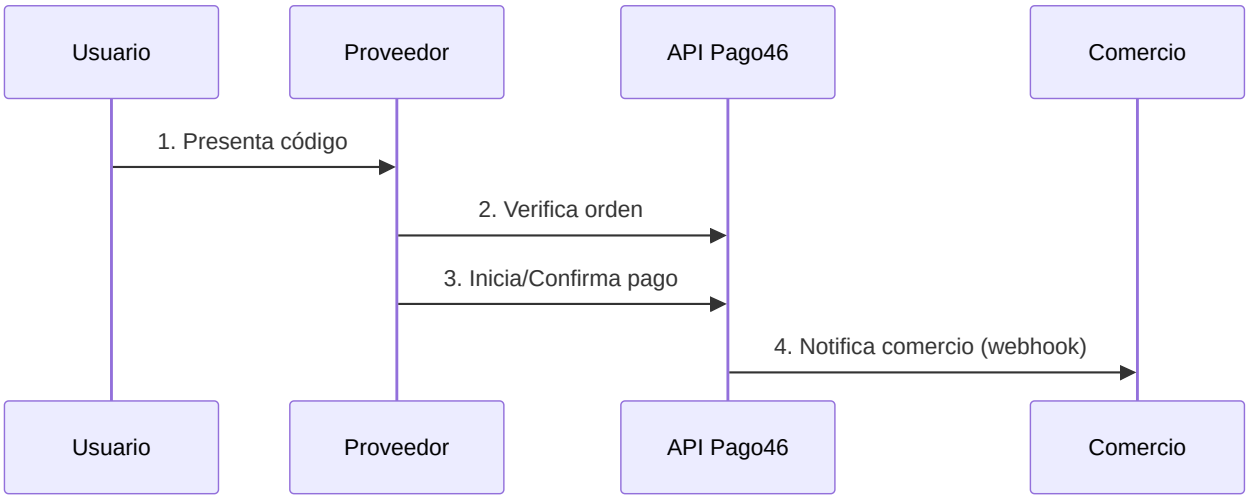
Dispersa efectivo a usuarios finales en tus puntos físicos.

Flujo: El usuario presenta el código → validas y bloqueas la orden → entregas el efectivo → confirmas la transacción.

Ver documentación de Pay-Out

Flujo de integración

La arquitectura es de servidor a servidor con autenticación HMAC SHA-256.



Pasos de integración

1. **Obtén tus credenciales** - Recibe tu `Provider-Key` y `Provider-Secret`
2. **Registra tus redes** - Envíanos tu lista de redes/subredes para asignar `network_id`
3. **Implementa autenticación HMAC** - Ver guía
4. **Consulta órdenes** - Usa `GET /providers/orders/{type}/{code}/`
5. **Procesa transacciones** - Usa `start-payment` y `confirm-payment`

MECANISMO DE BLOQUEO

El bloqueo aplica tanto en Pay-In como en Pay-Out. Cuando llamas a `start-payment`, la orden queda reservada para tu red hasta confirmar o cancelar.

REDES Y SUBREDES

Para usar los endpoints, necesitas un `network_id` válido. Envíanos tu lista de redes/subredes para que podamos registrarlas y asignar los IDs.

Ambientes

Ambiente	URL	Uso
Sandbox	`api.dev.pago46.io`	Desarrollo y pruebas
Producción	`api.prd.pago46.io`	Operaciones reales

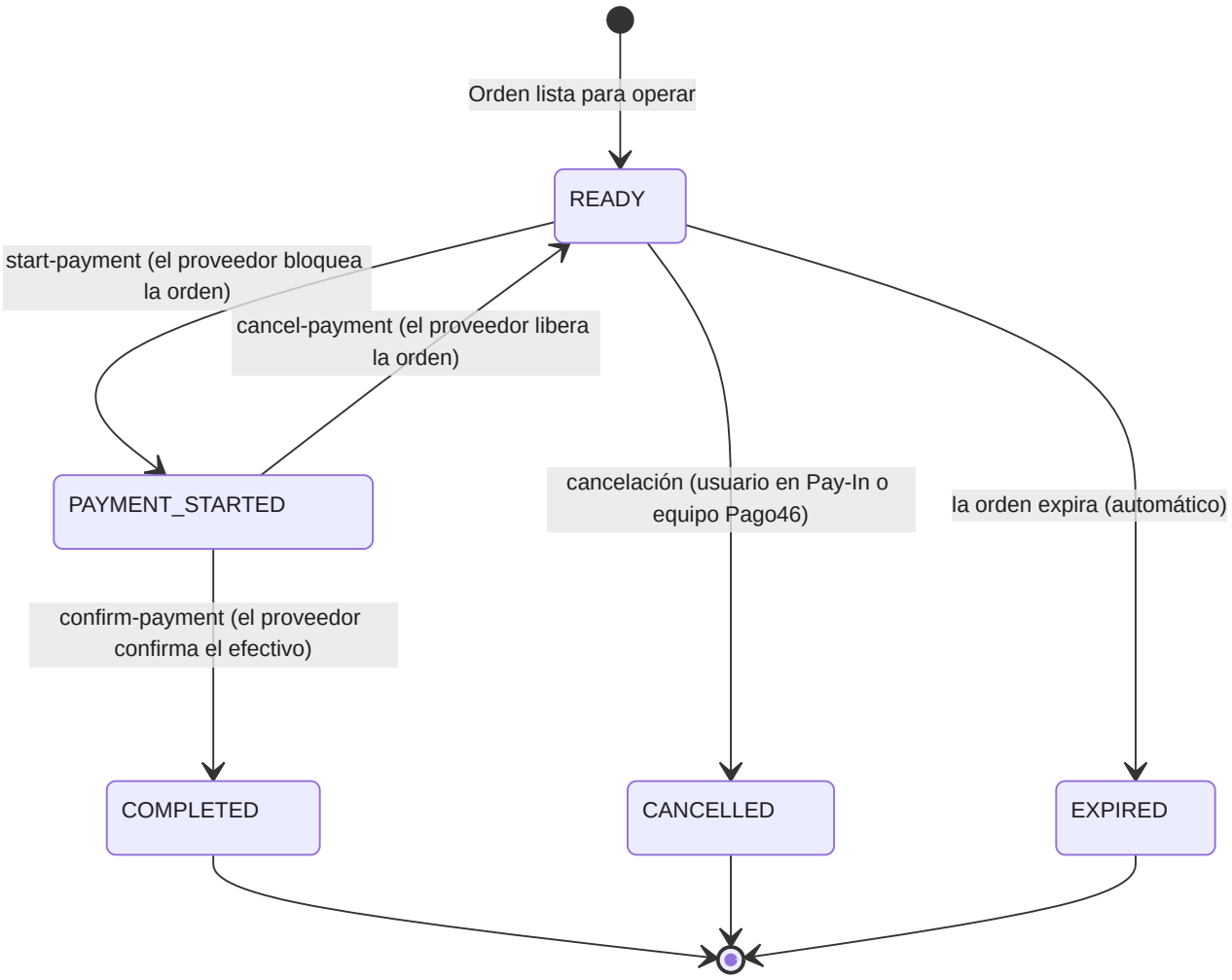
→ Ver detalles de ambientes

MONITOREO OPERATIVO

Para operación diaria en caja/soporte, consulta la página de Estado del Servicio y suscríbete a alertas por correo o Slack para enterarte de incidentes y mantenimientos.

Ciclo de vida de las órdenes

El diagrama de estados es el mismo para Pay-In y Pay-Out. Cada flecha indica la **acción que dispara la transición**: las que llevan nombre de endpoint (`start-payment`, `confirm-payment`, `cancel-payment`) son las que tú, como proveedor, invocas vía API; el resto ocurre de forma automática o lo dispara el comercio.



¿DESDE QUÉ ESTADO PUEDO ACTUAR?

La acción permitida depende del estado actual de la orden:

- **Tomar una orden** → `start-payment`, disponible **solo desde** **`READY`**.
- **Liberar / "devolver" una orden** → `cancel-payment`, disponible **solo desde** **`PAYMENT_STARTED`**, **antes de confirmar** y únicamente para el proveedor que bloqueó la orden. La orden vuelve a `READY` y queda disponible de nuevo. Si otro proveedor la bloqueó, no puedes actuar sobre ella.
- **Confirmar la transacción** → `confirm-payment`, disponible **solo desde** **`PAYMENT_STARTED`**. La orden pasa a `COMPLETED`. Confirma **únicamente cuando el efectivo ya cambió de manos**: en Pay-In cuando recibiste el dinero, en Pay-Out cuando lo entregaste.
- Los estados `COMPLETED`, `CANCELLED` y `EXPIRED` son **finales**: no admiten más transiciones.

NO HAY REVERSAS VÍA API

Es una operación en efectivo y la confirmación es inmediata: **una orden en** **`COMPLETED`** **no se puede devolver ni revertir** desde la API. Cualquier ajuste posterior se resuelve en la operación de liquidación (*settlement*) acordada con el equipo de Pago46. Por eso `confirm-payment` debe reflejar siempre un movimiento real de efectivo.

¿QUIÉN PUEDE CANCELAR UNA ORDEN?

Como proveedor **nunca** llevas una orden a `CANCELLED`: `cancel-payment` solo la libera de vuelta a `READY`. La cancelación definitiva la dispara:

- En **Pay-In**: el usuario final o el equipo de Pago46.
- En **Pay-Out**: **solo el equipo de Pago46**.

Los comercios no pueden cancelar órdenes.

Cobertura

Pago46 opera en múltiples países de Latinoamérica. Ver lista completa.

Pagos (Pay In)

Este módulo permite a los Proveedores de Pago (Payment Providers) procesar órdenes de **cobro en efectivo** creadas por comercios para sus usuarios. Al igual que en los retiros, este flujo utiliza un mecanismo de bloqueo (*locking*) para evitar que una misma orden sea cobrada dos veces por distintos proveedores simultáneamente.

Firma cada petición con HMAC SHA-256, revisa el formato de errores para manejar respuestas no exitosas y consulta los países y monedas soportados. Si eres nuevo, empieza por la introducción para Proveedores.

ENDPOINT BASE

Todas las rutas descritas a continuación son relativas a la URL base de la API: `/api/v1`

Registro de redes de pago

Antes de procesar transacciones, los proveedores deben registrar las redes de pago que utilizan. Cada red debe tener un identificador único (`network_id`) que se utilizará en todas las transiciones de estado.

Ejemplos de IDs de red:

- `01`
- `oxxo_network`
- `seven_eleven_mx`
- `farmacia_abc`
- `network_123`

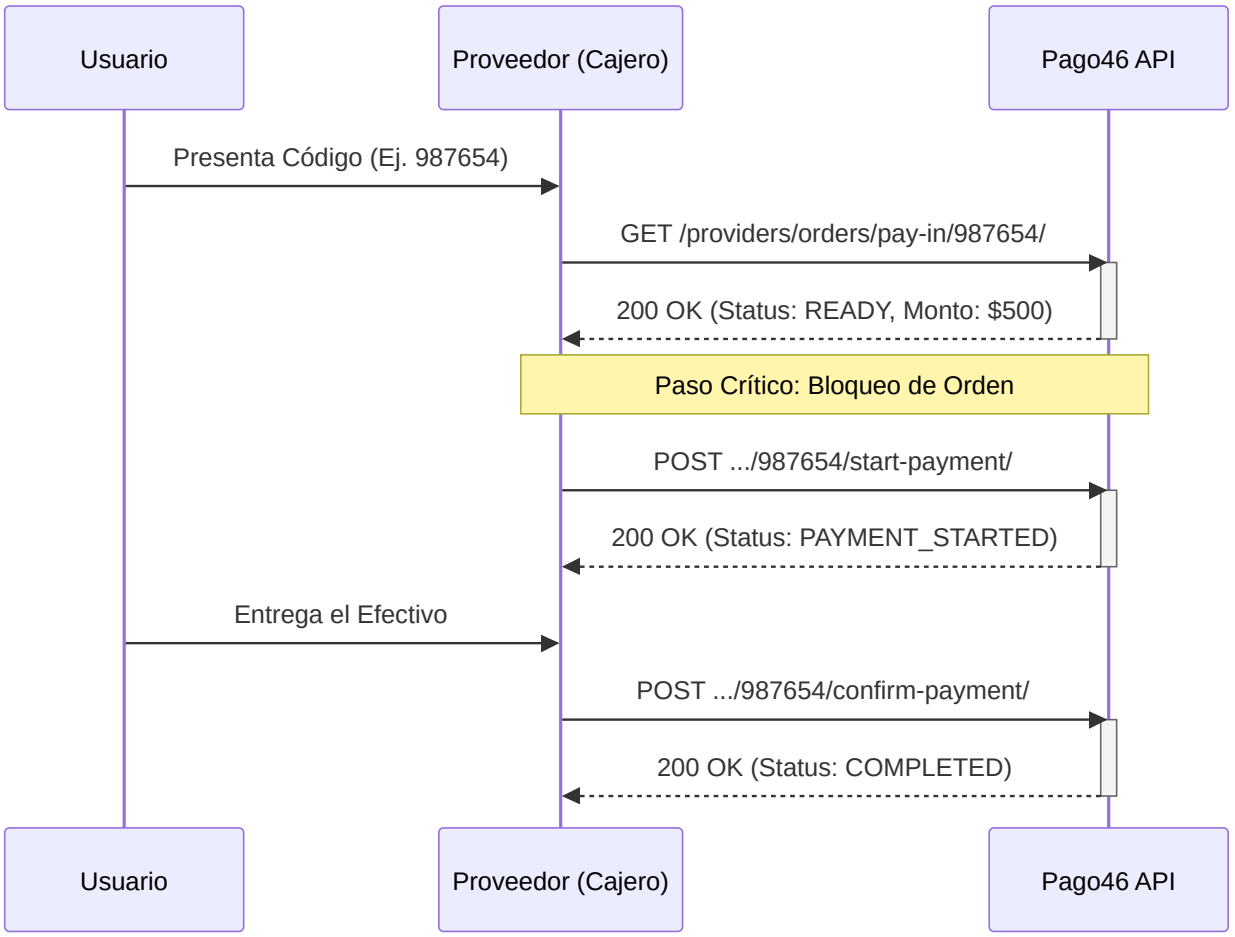
CONTACTO

Para registrar tus redes de pago, contacta a tu ejecutivo comercial o escríbenos a contacto@pago46.com. Los IDs que proporciones serán los que deberás usar en las solicitudes de cobro.

Ciclo de vida de la orden

El proceso asegura que el dinero sea recolectado y conciliado correctamente antes de liberar el servicio al usuario final.

1. **Consulta:** El proveedor escanea o ingresa el código presentado por el usuario para ver el monto a cobrar y validar que el estado sea `READY`.
2. **Bloqueo:** El proveedor "toma" la orden. El estado cambia a `PAYMENT_STARTED`. **Esto asegura la intención de cobro.**
3. **Confirmación:** Una vez que el usuario entrega el efectivo y este ingresa a la caja, el proveedor confirma la transacción (estado `COMPLETED`).



---

1. Verificar orden

Cuando el cliente se acerca a la caja para pagar, debes consultar el código para informar el monto exacto a cobrar.

Endpoint: `GET /providers/orders/pay-in/{code}/`

Parámetro	Ubicación	Descripción
`code`	Path	El código de pago presentado por el usuario (numérico o QR).

Petición

```
curl -X GET "https://api.dev.pago46.io/api/v1/providers/orders/pay-in/9876543210/" \
  -H "Provider-Key: <TU_PROVIDER_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>"
```

Respuesta (200 OK) - Orden disponible


```
{
  "network_id": null,
  "price": "500.00",
  "price_currency": "MXN",
  "created": "2026-06-25T14:00:00Z",
  "modified": "2026-06-25T14:00:00Z",
  "status": "READY",
  "expiry": "2026-06-26T14:00:00Z"
}
```


> ⚠ **VALIDACIÓN DE ESTADO**
>
> Verifica siempre que el `status` sea **`READY`**. Si la orden está `EXPIRED` o `COMPLETED`, no debes recibir dinero del usuario.

---

2. Iniciar cobro (bloqueo)

Antes de recibir el dinero, debes informar a Pago46 que estás atendiendo esta orden. Esto evita que el usuario intente pagar el mismo código en otro proveedor al mismo tiempo.

Endpoint: `POST /providers/orders/pay-in/{code}/start-payment/`

Petición

```
curl -X POST "https://api.dev.pago46.io/api/v1/providers/orders/pay-in/9876543210/start-payment/" \
  -H "Provider-Key: <TU_PROVIDER_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>" \
  -H "Content-Type: application/json" \
  -d '{
    "network_id": "network_123",
    "price": "500.00",
    "price_currency": "MXN"
  }'
```

Respuesta (200 OK)


```
{
  "network_id": "network_123",
  "price": "500.00",
  "price_currency": "MXN",
  "created": "2026-06-25T14:00:00Z",
  "modified": "2026-06-25T14:05:00Z",
  "status": "PAYMENT_STARTED",
  "expiry": "2026-06-26T14:00:00Z"
}
```


Error (400) - Orden no disponible


```
{
  "type": "validation_error",
  "errors": [
    {
      "code": "invalid",
      "detail": "Transition is not allowed",
      "attr": null
```

```
    }
  ]
}
```

Campo	Tipo	Descripción
`network_id`	string	ID de la red de pago utilizada por el proveedor. **Requerido**.
`price`	string	Opcional. Si lo envías, debe coincidir con el `price` de la orden.
`price_currency`	string	Opcional. Si lo envías, debe coincidir con el `price_currency` de la orden.

SUBREDES PARA PROVEEDORES

Como proveedor, debes suministrar el `network_id` que identifica la red específica utilizada para procesar esta transacción. Esto permite a Pago46 mantener un registro detallado de qué red procesó cada pago y proporcionar esta información al usuario final.

MOMENTO DEL COBRO

Una vez recibas el estado `PAYMENT_STARTED`, puedes proceder a solicitar y recibir el dinero del cliente con seguridad.

---

3. Confirmar cobro (finalización)

Una vez que el dinero esté seguro en tu caja, confirma la transacción. En ese momento, Pago46 notificará al comercio que creó la orden para que entregue el producto o servicio al usuario.

Endpoint: `POST /providers/orders/pay-in/{code}/confirm-payment/`

Petición

```
curl -X POST "https://api.dev.pago46.io/api/v1/providers/orders/pay-in/9876543210/confirm-payment/" \
  -H "Provider-Key: <TU_PROVIDER_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>" \
  -H "Content-Type: application/json" \
  -d '{
    "network_id": "network_123",
    "price": "500.00",
    "price_currency": "MXN"
  }'
```

Respuesta (200 OK)

```
{
  "network_id": "network_123",
  "price": "500.00",
  "price_currency": "MXN",
  "created": "2026-06-25T14:00:00Z",
  "modified": "2026-06-25T14:06:00Z",
  "status": "COMPLETED",
  "expiry": "2026-06-26T14:00:00Z"
}
```

Cancelación

Si el usuario decide no pagar en el último momento o hay un problema con el efectivo, debes liberar la orden. Solo puedes hacerlo si la orden está en `PAYMENT_STARTED` y tu proveedor fue quien la bloqueó. Si otro proveedor la bloqueó, no puedes actuar sobre ella. Al liberarla, vuelve a `READY` para que pueda procesarse nuevamente.

Endpoint: `POST /providers/orders/pay-in/{code}/cancel-payment/`

```
curl -X POST "https://api.dev.pago46.io/api/v1/providers/orders/pay-in/9876543210/cancel-payment/" \
  -H "Provider-Key: <TU_PROVIDER_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>" \
  -H "Content-Type: application/json" \
  -d '{
    "network_id": "network_123",
    "price": "500.00",
    "price_currency": "MXN"
  }'
```

Resumen de estados

Estado	Descripción	Acción requerida del proveedor
`READY`	La orden está pendiente de pago.	Verificar monto y llamar a `start-payment`.
`PAYMENT_STARTED`	La orden está en proceso de cobro.	Recibir dinero y llamar a `confirm-payment`.
`COMPLETED`	El dinero fue recaudado exitosamente.	Entregar comprobante al usuario.
`CANCELLED`	La orden fue cancelada.	No recibir dinero.

Retiros (Pay Out)

Este módulo permite a los Proveedores de Pago (Payment Providers) procesar órdenes de retiro de efectivo creadas por comercios para sus usuarios. El flujo está diseñado para garantizar la atomicidad de la transacción y evitar entregas duplicadas de efectivo mediante un mecanismo de bloqueo (*locking*).

Firma cada petición con HMAC SHA-256, revisa el formato de errores para manejar respuestas no exitosas y consulta los países y monedas soportados. Si eres nuevo, empieza por la introducción para Proveedores.

> **ENDPOINT BASE**
>
> Todas las rutas descritas a continuación son relativas a la URL base de la API: `/api/v1`

Registro de redes de pago

Antes de procesar transacciones, los proveedores deben registrar las redes de pago que utilizan. Cada red debe tener un identificador único (`network_id`) que se utilizará en todas las transiciones de estado.

Ejemplos de IDs de red:

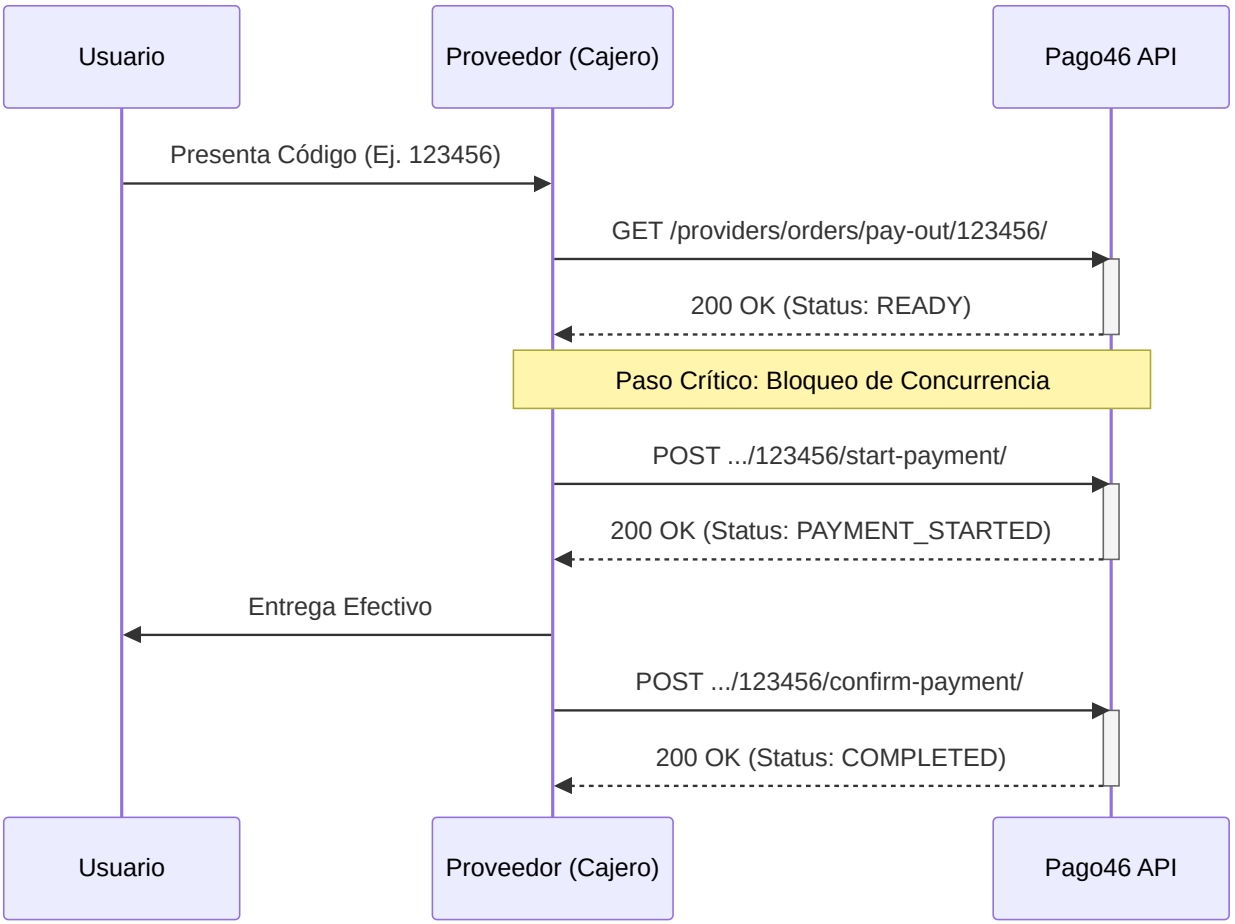
- `01`
- `oxxo_network`
- `seven_eleven_mx`
- `farmacia_abc`
- `network_123`

> **CONTACTO**
>
> Para registrar tus redes de pago, contacta a tu ejecutivo comercial o escríbenos a contacto@pago46.com. Los IDs que proporciones serán los que deberás usar en las solicitudes de retiro.

Ciclo de vida de la orden

El proceso se rige por un cambio estricto de estados para asegurar que el dinero solo se entregue una vez.

1. **Consulta:** El proveedor verifica si el código presentado por el usuario es válido y la orden está en estado `READY`.
2. **Bloqueo:** El proveedor "toma" la orden. Esto cambia el estado a `PAYMENT_STARTED`. **En este punto, ningún otro proveedor puede procesar esta orden.**
3. **Confirmación:** Una vez entregado el dinero, el proveedor confirma la transacción, pasando la orden a `COMPLETED`.



---

1. Verificar orden

El primer paso ocurre cuando el usuario presenta su código de retiro en el punto físico. Debes consultar los detalles de la orden para validar el monto y su disponibilidad.

Endpoint: `GET /providers/orders/pay-out/{code}/`

Parámetro	Ubicación	Descripción
`code`	Path	El código numérico o alfanumérico presentado por el usuario final.

Petición

```
curl -X GET "https://api.dev.pago46.io/api/v1/providers/orders/pay-out/1234567890/" \
  -H "Provider-Key: <TU_PROVIDER_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>"
```

Respuesta (200 OK) - Orden disponible

```
{
  "network_id": null,
  "price": "1500.00",
  "price_currency": "MXN",
  "created": "2026-06-25T10:00:00Z",
```

```
  "modified": "2026-06-25T10:05:00Z",
  "status": "READY",
  "expiry": "2026-06-26T10:00:00Z"
}
```

Respuesta (200 OK) - Orden en proceso

```
{
  "network_id": "network_123",
  "price": "1500.00",
  "price_currency": "MXN",
  "created": "2026-06-25T10:00:00Z",
  "modified": "2026-06-25T10:05:00Z",
  "status": "PAYMENT_STARTED",
  "expiry": "2026-06-26T10:00:00Z"
}
```

> ⚠ **VALIDACIÓN DE ESTADO**
>
> Solo debes proceder al siguiente paso si el campo `status` es **`READY`**. Si recibes `CANCELLED`, `COMPLETED` o `EXPIRED`, debes informar al usuario que la orden no puede ser procesada.

---

2. Iniciar pago (bloqueo)

Este es el paso más crítico. Al ejecutar este endpoint, estás indicando que tienes la intención de pagar la orden. El sistema **bloqueará** la orden para tu proveedor y cambiará su estado a `PAYMENT_STARTED`.

Si otro proveedor intenta iniciar el pago de la misma orden mientras está bloqueada, recibirá un error `400` indicando que la transición no está permitida.

Endpoint: `POST /providers/orders/pay-out/{code}/start-payment/`

Petición

```
curl -X POST "https://api.dev.pago46.io/api/v1/providers/orders/pay-out/1234567890/start-payment/" \
  -H "Provider-Key: <TU_PROVIDER_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>" \
  -H "Content-Type: application/json" \
  -d '{
    "network_id": "network_123",
    "price": "1500.00",
    "price_currency": "MXN"
  }'
```

Respuesta (200 OK)

```
{
  "network_id": "network_123",
  "price": "1500.00",
  "price_currency": "MXN",
  "created": "2026-06-25T10:00:00Z",
  "modified": "2026-06-25T10:06:00Z",
  "status": "PAYMENT_STARTED",
  "expiry": "2026-06-26T10:00:00Z"
}
```

Error (400) - Orden no disponible

```
{
  "type": "validation_error",
  "errors": [
    {
      "code": "invalid",
      "detail": "Transition is not allowed",
      "attr": null
    }
  ]
}
```

Campo	Tipo	Descripción
`network_id`	string	ID de la red de pago utilizada por el proveedor. **Requerido**.
`price`	string	Opcional. Si lo envías, debe coincidir con el `price` de la orden.
`price_currency`	string	Opcional. Si lo envías, debe coincidir con el `price_currency` de la orden.

SUBREDES PARA PROVEEDORES

Como proveedor, debes suministrar el `network_id` que identifica la red específica utilizada para procesar esta transacción. Esto permite a Pago46 mantener un registro detallado de qué red procesó cada pago y proporcionar esta información al usuario final.

OPERACIÓN SEGURA

Una vez recibes el `200 OK` con estado `PAYMENT_STARTED`, es seguro proceder a la entrega física del dinero o la gestión interna de fondos. La orden está reservada para ti.

---

3. Confirmar pago (finalización)

Una vez que el dinero ha sido entregado exitosamente al usuario (o la transacción interna ha finalizado), debes confirmar la operación para cerrar la orden definitivamente.

Endpoint: `POST /providers/orders/pay-out/{code}/confirm-payment/`

Petición

```
curl -X POST "https://api.dev.pago46.io/api/v1/providers/orders/pay-out/1234567890/confirm-payment/" \
  -H "Provider-Key: <TU_PROVIDER_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>" \
  -H "Content-Type: application/json" \
  -d '{
    "network_id": "network_123",
    "price": "1500.00",
    "price_currency": "MXN"
  }'
```

Respuesta (200 OK)

```
{
  "network_id": "network_123",
  "price": "1500.00",
  "price_currency": "MXN",
  "created": "2026-06-25T10:00:00Z",
  "modified": "2026-06-25T10:08:00Z",
  "status": "COMPLETED",
  "expiry": "2026-06-26T10:00:00Z"
}
```

Al recibir esta respuesta, Pago46 notificará al comercio que creó la orden mediante el webhook configurado.

Cancelación (opcional)

Si por alguna razón (insuficiencia de fondos en caja, error operativo) no puedes completar el pago después de haber ejecutado `start-payment`, **debes** liberar la orden para no dejarla bloqueada indefinidamente. Solo puedes hacerlo si la orden está en `PAYMENT_STARTED` y tu proveedor fue quien la bloqueó; si otro proveedor la bloqueó, no puedes actuar sobre ella.

Endpoint: `POST /providers/orders/pay-out/{code}/cancel-payment/`

Esto devuelve la orden a `READY`, donde queda disponible para procesarse nuevamente.

```
curl -X POST "https://api.dev.pago46.io/api/v1/providers/orders/pay-out/1234567890/cancel-payment/" \
  -H "Provider-Key: <TU_PROVIDER_KEY>" \
  -H "Message-Date: <TIMESTAMP>" \
  -H "Message-Hash: <HMAC_SIGNATURE>" \
  -H "Content-Type: application/json" \
  -d '{
    "network_id": "network_123",
    "price": "1500.00",
    "price_currency": "MXN"
  }'
```

Resumen de estados

Estado	Descripción	Acción requerida del proveedor
`READY`	La orden está lista para ser pagada.	Puede llamar a `start-payment`.
`PAYMENT_STARTED`	La orden está bloqueada por un proveedor.	Debe entregar el dinero y llamar a `confirm-payment`.
`COMPLETED`	El flujo terminó exitosamente.	No requiere acción; registro histórico.
`CANCELLED`	La orden fue anulada.	No entregar dinero.

Estado del servicio

La **página de estado de Pago46** centraliza el estado operativo de nuestros servicios para que puedas validar la disponibilidad, dar seguimiento a incidentes y consultar los mantenimientos programados en tiempo real.

URL oficial: `https://pago46.statuspage.io`

¿Qué información encontrarás?

- **Estado actual por componente:** Vista rápida de la salud de los servicios clave.
- **Incidentes activos e históricos:** Actualizaciones sobre degradaciones, interrupciones y resolución.
- **Mantenimientos programados:** Ventanas planificadas y su impacto esperado.

Suscripciones y alertas

Si operas en producción, te recomendamos suscribirte para recibir notificaciones proactivas.

Correo electrónico

1. Entra a `https://pago46.statuspage.io`.
2. Haz clic en **Subscribe to updates**.
3. Selecciona **Email**.
4. Ingresa tu correo y confirma la suscripción.

Slack

1. Entra a `https://pago46.statuspage.io`.
2. Haz clic en **Subscribe to updates**.
3. Selecciona **Slack**.
4. Autoriza la integración y elige el canal donde quieras recibir alertas.

> **RECOMENDACIÓN OPERATIVA**
>
> Suscribe al menos un correo de tu equipo de soporte o de guardia y un canal de Slack compartido para tener visibilidad interna ante incidentes o mantenimientos.

Widget de estado en esta documentación

Además de la página de estado, este sitio de documentación ya tiene integrado el widget oficial de Statuspage.

Cuando exista una **interrupción**, degradación o **mantenimiento**, podrás ver señales de estado también desde aquí para tener contexto operativo sin salir de la documentación.

¿Cuándo consultarla?

- Si detectas errores inusuales en llamadas API o caídas de webhook.
- Antes de escalar un incidente, para validar si ya existe una comunicación oficial.
- Al planificar despliegues críticos en producción.

Si necesitas ayuda adicional, puedes escribirnos a contacto@pago46.com.

Elige tu guía

Soy un Comercio

Quiero cobrar pagos o enviar dinero a mis usuarios.

- Cobros (Pay-In)
- Retiros (Pay-Out)
- Retiros en moneda extranjera

Soy un Proveedor de Pago

Tengo puntos físicos y quiero procesar transacciones de Pago46.

- Procesar pagos
- Procesar retiros

Guía de diseño

Esta guía proporciona los recursos visuales y técnicos necesarios para integrar Pago46 en tu plataforma de manera profesional y consistente.

Identidad visual

Mantener la coherencia visual ayuda a generar confianza en el usuario final durante el proceso de pago.

Colores institucionales

Hemos preparado los valores exactos en diferentes formatos para que puedas copiarlos y pegarlos directamente en tu hoja de estilos o software de diseño.

Color	Uso principal	HEX	RGB	HSL	CMYK
Verde	Botones de acción	`#28C339`	`rgb(40, 195, 57)`	`hsl(127, 66%, 46%)`	`79, 0, 71, 24`
Azul oscuro	Textos y fondos	`#07214F`	`rgb(7, 33, 79)`	`hsl(218, 84%, 17%)`	`91, 58, 0, 69`
Blanco	Fondos claros	`#FFFFFF`	`rgb(255, 255, 255)`	`hsl(0, 0%, 100%)`	`0, 0, 0, 0`

Tipografía

Nuestra fuente oficial es **Lato**. Facilita la legibilidad en interfaces de pago y dispositivos móviles.

- **Uso en botones:** Negrita (Bold), **16px**, centrado, letter-spacing: 0px.
- **Fallback:** En caso de no poder cargar **Lato**, utiliza sans-serif.

> **OBTENER LA TIPOGRAFÍA**
>
> Si no cuentas con la tipografía en tu proyecto, puedes descargarla o importarla gratuitamente a través de Google Fonts o encontrarla en Adobe Fonts.

Logotipos base

Activos oficiales con fondo transparente. Utiliza la versión **SVG** siempre que sea posible para garantizar la máxima nitidez.

Versión	Color	SVG	PNG
Full	**Verde**	**Descargar:** svg	**Descargar:** sm · md · lg
	Azul	**Descargar:** svg	**Descargar:** sm · md · lg
	Blanco	**Descargar:** svg	**Descargar:** sm · md · lg

Versión	Color	SVG	PNG
Isotipo	Verde	 Descargar: svg	 Descargar: sm · md · lg
	Azul	 Descargar: svg	 Descargar: sm · md · lg
	Blanco	 Descargar: svg	 Descargar: sm · md · lg

Catálogo de botones

Ofrecemos botones listos para usar en múltiples idiomas, tamaños y esquemas de color. Selecciona tu preferencia a continuación para obtener los enlaces de descarga directa.

Español

Variantes con los textos **"Efectivo"** y **"Pagar con"**.

Verde / Azul

Tamaño	Full (Efectivo)	Full (Pagar con)	Iso (Efectivo)	Iso (Pagar con)
xs	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
lg	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG

Verde / Blanco

Tamaño	Full (Efectivo)	Full (Pagar con)	Iso (Efectivo)	Iso (Pagar con)
xs	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
lg	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG

Azul / Verde

Tamaño	Full (Efectivo)	Full (Pagar con)	Iso (Efectivo)	Iso (Pagar con)
xs	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
lg	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG

Azul / Blanco

Tamaño	Full (Efectivo)	Full (Pagar con)	Iso (Efectivo)	Iso (Pagar con)
xs	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
lg	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG

Blanco / Azul

Tamaño	Full (Efectivo)	Full (Pagar con)	Iso (Efectivo)	Iso (Pagar con)
xs	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	SVG / PNG	SVG / PNG	SVG / PNG	SVG / PNG
lg	SVG / PNG	SVG / PNG	SVG / PNG	SVG / PNG

Blanco / Verde

Tamaño	Full (Efectivo)	Full (Pagar con)	Iso (Efectivo)	Iso (Pagar con)
xs	SVG / PNG	SVG / PNG	SVG / PNG	SVG / PNG

Tamaño	Full (Efectivo)	Full (Pagar con)	Iso (Efectivo)	Iso (Pagar con)
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
lg	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG

English

Variants with **"Cash"** and **"Pay with"** labels.

Green / Blue

Size	Full (Cash)	Full (Pay with)	Iso (Cash)	Iso (Pay with)
xs	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
lg	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG

Green / White

Size	Full (Cash)	Full (Pay with)	Iso (Cash)	Iso (Pay with)
xs	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
lg	SVG / PNG	SVG / PNG	SVG / PNG	SVG / PNG

Blue / Green

Size	Full (Cash)	Full (Pay with)	Iso (Cash)	Iso (Pay with)
xs	SVG / PNG	SVG / PNG	SVG / PNG	SVG / PNG

Size	Full (Cash)	Full (Pay with)	Iso (Cash)	Iso (Pay with)
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
lg	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG

Blue / White

Size	Full (Cash)	Full (Pay with)	Iso (Cash)	Iso (Pay with)
xs	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
lg	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG

White / Blue

Size	Full (Cash)	Full (Pay with)	Iso (Cash)	Iso (Pay with)
xs	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
sm	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
md	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG
lg	SVG / PNG	SVG / PNG	SVG / PNG	SVG / PNG

White / Green

Size	Full (Cash)	Full (Pay with)	Iso (Cash)	Iso (Pay with)
xs	SVG / PNG	SVG / PNG	SVG / PNG	SVG / PNG
sm	SVG / PNG	SVG / PNG	SVG / PNG	SVG / PNG
md	SVG / PNG	SVG / PNG	SVG / PNG	SVG / PNG

Size	Full (Cash)	Full (Pay with)	Iso (Cash)	Iso (Pay with)
lg	 SVG / PNG	 SVG / PNG	 SVG / PNG	 SVG / PNG

DESCARGA MASIVA

Puedes descargar el paquete completo con todas las variantes e idiomas en el siguiente enlace.

📦 Descargar Todos los Activos (.zip)

Ejemplos de implementación

Copia y pega estos fragmentos de código para integrar el botón rápidamente en tu frontend. Estos ejemplos utilizan la variante **Efectivo + Logo Full (Fondo Verde / Texto y Logo Azul)** en su versión SVG.

HTML & CSS

```
<button class="p46-btn-pay">
  <!-- Recomendamos usar el SVG directamente para mejor calidad -->
  <img
    src="/img/buttons/green-and-dark-blue-contrast/logo-full/md/button-cash-ES.svg"
    alt="Pagar con Pago46"
    class="p46-button-img"
  />
</button>

<style>
  .p46-btn-pay {
    background: none;
    border: none;
    padding: 0;
    cursor: pointer;
    transition: transform 0.2s ease, filter 0.2s ease;
  }

  .p46-btn-pay:hover {
    filter: brightness(90%);
  }

  .p46-btn-pay:active {
    transform: scale(0.98);
  }

  .p46-btn-pay:disabled {
    opacity: 0.5;
    cursor: not-allowed;
  }

  .p46-button-img {
    height: 48px; /* Ajusta según tu diseño */
    display: block;
  }
</style>
```

React (Tailwind)

```
import React from "react";

const Pago46Button = ({ onClick, disabled }) => {
  return (
    <button
```

```
      onClick={onClick}
      disabled={disabled}
      className="transition-all hover:brightness-90 active:scale-95 disabled:opacity-50 disabled:cursor-not-allowed"
    >
      <img
        src="/img/buttons/green-and-dark-blue-contrast/logo-full/md/button-cash-ES.svg"
        alt="Pagar con Pago46"
        className="h-[48px] block"
      />
    </button>
  );
};

export default Pago46Button;
```

Restricciones y malas prácticas

Para evitar el rechazo de la integración durante el proceso de certificación, **no realices** las siguientes acciones:

- **Modificar proporciones:** No estires ni comprimas el logo o el isotipo.
- **Alterar colores:** No cambies el verde institucional por otros tonos de verde o colores ajenos a la marca.
- **Usar tipografía incorrecta:** No utilices fuentes con serifa (serif) como Times New Roman en el botón.
- **Usar contraste deficiente:** No utilices el logo verde sobre fondos de colores similares que dificulten la lectura.

CERTIFICACIÓN

Antes de pasar a producción, nuestro equipo revisará que la implementación del botón cumpla con estos lineamientos para asegurar la mejor tasa de conversión para tu comercio.

Elige tu guía

Soy un Comercio

Quiero cobrar pagos o enviar dinero a mis usuarios.

- Cobros (Pay-In)
- Retiros (Pay-Out)
- Retiros en moneda extranjera

Soy un Proveedor de Pago

Tengo puntos físicos y quiero procesar transacciones de Pago46.

- Procesar pagos
- Procesar retiros