



Operadora de Pagos Móviles de México

Módulo de Dispersiones OPM

Manual de integración

Tabla de Contenido

Introducción	3
Arquitectura de la interfaz	3
Requisitos técnicos	3
Tipos de cuentas	4
Algoritmo para generar dígito verificador	6
Flujo de una orden	7
Pasos de integración	8
Obtención de credenciales	8
Llamar endpoint Create Order	8
Ejemplos de código para la generación de firma digital	11
Java	11
PHP	12
Node.js	13
C#	15
Recepción de Webhooks	16
Webhook de órdenes de entrada	17
Webhook de órdenes de salida	22
Devoluciones	23
Conexión de endpoints	23
Diagrama de flujo de transacciones recibidas	24
Diagrama de flujo de transacciones enviadas	25



Introducción

El Módulo de Dispersiones OPM (MDO), es un servicio que permite al cliente realizar transferencias bancarias a través del Sistema de Pagos Electrónicos Interbancarios (SPEI).

Existen dos vías para realizar transacciones utilizando el MDO. A través del dashboard de administración se puede llenar manualmente los datos de la transacción o a través de la conexión de otro sistema a MDO por medio de un API.

Este documento pretende guiar al integrador en el desarrollo de la interfaz entre los sistemas por medio del uso del API expuesta por MDO.

Arquitectura de la interfaz

El sistema está diseñado para funcionar en una arquitectura host to host, es decir, la conexión a MDO deberá realizarse desde el servidor del integrador. Esta conexión se realizará a través de Internet.

El servicio de MDO recibirá las llamadas https iniciadas por el integrador a cualquiera de los endpoints especificados en la documentación de API. De igual manera, el servicio MDO iniciará la comunicación cuando deba enviar una notificación al servidor del integrador.

Requisitos técnicos

Para poder comunicarse con el API de MDO es necesario cumplir con los siguientes requisitos técnicos.

- Deben contar con servicios HTTP expuestos por internet, utilizando el protocolo https y contar con un certificado SSL emitido por una autoridad certificadora.
- Proveer a OPM con las IP's públicas de los servicios desde los cuales se realizarán llamadas a MDO y las IP's públicas de los servicios que recibirán llamadas desde MDO.
- Tener un usuario y contraseña para acceder al frontend de MDO. La cuenta será enviada una vez OPM haya realizado las autorizaciones internas correspondientes. Las rutas del frontend son:
 - Producción: <https://mdo.opm.mx/>
 - UAT: <https://transfercid.com:10443/>



- Contar con un API key y llave pública y privada que deben ser descargadas desde la sección de “Integración” dentro del portal de OPM.

Los servicios de MDO se encuentran expuestos en las siguientes ligas:

- UAT: <https://apiuat.opm.mx>
- PROD: <https://api.opm.mx>

Por ejemplo, para crear una orden, el consumo se haría a la siguiente liga en ambiente UAT.

- <https://apiuat.opm.mx/api/1.0/orders/>

Puntos importantes a considerar.

- Las credenciales de autenticación API key, y llaves públicas y privadas son diferentes en cada ambiente.
- La lista completa de endpoints que pueden consumirse pueden consultarse en el documento de especificación de API.

Tipos de cuentas

Existen dos tipos de cuentas en MDO:

- Cuentas operativas: Son cuentas que pueden realizar transferencias a otros bancos.
- Cuentas virtuales: Solo pueden recibir transacciones.

Existen dos modalidades de MDO para asignar cuentas virtuales derivadas de una cuenta operativa:

- Modalidad cuenta única
- Modalidad subcuentas



Modalidad cuenta única

Las cuentas asignadas por OPM a los integradores con modalidad subcuentas tienen el siguiente formato y está dividida en cinco segmentos.

684 180 PPP XXXXXXXX D (99,999,999 cuentas virtuales)

Primer segmento: dígitos del banco

Segundo segmento: dígitos de la sucursal

Tercer segmento: dígitos del integrador

Cuarto segmento: dígitos de la cuenta virtual

Quinto segmento: dígito verificador

Ejemplos de cuentas virtuales:

684180079 00000001 **5**

...

684180079 99999999 **8**

Los pagos salientes deben ser realizados desde la cuenta con terminación 00000000X (Siendo X el dígito verificador). Las cuentas virtuales no pueden generar pagos desde MDO.

Ejemplo de cuenta:

684 180 001 00000001 7

Modalidad subcuentas

Las cuentas asignadas por OPM a los integradores con modalidad subcuentas tienen el siguiente formato y está dividida en seis segmentos.

684 180 001 CCC XXXXX D (99,999 cuentas virtuales)

Primer segmento: dígitos del banco

Segundo segmento: dígitos de la sucursal

Tercer segmento: dígitos del integrador

Cuarto segmento: dígitos del centro de costos

Quinto segmento: dígitos de la cuenta virtual

Sexto segmento: dígito verificador



Ejemplos de cuentas virtuales:

684180079001 XXXX D

...

684180079999 XXXX D

Todos los clientes en MDO con la modalidad subcuentas tienen asignado el centro de costos 000. Cada centro de costos tiene un saldo independiente. Si se desea tener más centros de costos (001,002,...) es necesario solicitarlo OPM para que se realice el registro ante Banco de México.

Solo es posible modificar los últimos 6 dígitos de la cuenta, teniendo así disponibles hasta 99,999 cuentas virtuales por centro de costos.

Los pagos realizados a las cuentas virtuales afectarán el saldo asociado a su centro de costos. Por ejemplo, si se realiza un pago a la cuenta 6841800130000000016 el saldo de la cuenta 6841800130000000003 será afectado.

Los pagos salientes deben ser realizados desde la cuenta con terminación 00000X (Siendo X el dígito verificador). Las cuentas virtuales no pueden generar pagos desde MDO.

Algoritmo para generar dígito verificador

Se describe este procedimiento mediante un ejemplo, para el número de cuenta 00218003224094670.

Considerar los siguientes factores de ponderación para cada dígito:

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Ponderación	3	7	1	3	7	1	3	7	1	3	7	1	3	7	1	3	7

El dígito verificador se calcula de la siguiente manera:

1. Multiplicar cada dígito del número de cuenta por el factor de ponderación respectivo:

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Cuenta	0	0	2	1	8	0	0	3	2	2	4	0	9	4	6	7	0
Ponderación	3	7	1	3	7	1	3	7	1	3	7	1	3	7	1	3	7
Resultado	0	0	2	3	56	0	0	21	2	6	28	0	27	28	6	21	0

2. Tomar módulo 10 de cada resultado obtenido en el paso 1:

Resultado	0	0	2	3	56	0	0	21	2	6	28	0	27	28	6	21	0
Módulo 10	0	0	2	3	6	0	0	1	2	6	8	0	7	8	6	1	0

3. Sumar los resultados de cada una de las operaciones módulo realizadas en el paso 2.

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Cuenta	0	0	2	1	8	0	0	3	2	2	4	0	9	4	6	7	0
Ponderación	3	7	1	3	7	1	3	7	1	3	7	1	3	7	1	3	7
Resultado	0	0	2	3	56	0	0	21	2	6	28	0	27	28	6	21	0

4. Tomar el módulo 10 de la suma calculada en el paso 3:

$$A = 50 \text{ mod } 10 = 0$$

5. Tomar el valor A obtenido en el paso 4 y restarlo de 10.

$$B = 10 - A = 10 - 0 = 10$$

6. El Dígito Verificador es el resultado de obtener el módulo 10 del número B calculado en el paso 5:

$$\text{Dígito Verificador} = \text{mod } 10 \ 10 = 0$$

7. La CLABE se obtiene al agregar al número de cuenta original el dígito verificador calculado. De esta manera, en este ejemplo, la CLABE es 002180032240946700

Flujo de una orden

Una orden es una transferencia de dinero. El flujo de envío de una orden es el siguiente:

- Servidor de integrador envía orden a MDO por medio del API.
- Servicio MDO registra la orden y responde síncronamente con la información de la orden y un id único.



- Si la cuenta del remitente tiene saldo suficiente, el saldo se reduce del saldo general y se suma al saldo en tránsito.
- En caso de no tener saldo suficiente, la orden queda en espera y se continúa con el flujo al recibir el próximo depósito.
- El Servicio MDO envía de manera asíncrona la orden al Banco de México.
- El Servicio MDO envía una notificación por medio del webhook de actualización de estado de orden, informando que la orden se encuentra en estado "sent".
- Banco de México entrega la orden al participante del SPEI destinatario.
- Banco de México avisa a MDO si la orden fue recibida o rechazada.
- Si la orden fue recibida, el monto es restado del saldo en tránsito.
- Si la orden fue rechazada, el monto es restado del saldo en tránsito y sumado al saldo general.
- MDO avisa al servidor del integrador por medio de un webhook el estado de la orden.

Pasos de integración

Para empezar a utilizar el servicio MDO es necesario tener 4 puntos cubiertos que detallaremos a continuación.

Obtención de credenciales

Al crear la cuenta un administrador de OPM debe entregar las credenciales de acceso al integrador. Las credenciales incluyen:

- Usuario y contraseña del dashboard
- Llaves pública y privada para el API (RSA PEM) y API Key. Se descargan desde la sección de integración del dashboard.

Llamar endpoint Create Order

Para poder utilizar este endpoint debes tener la cuenta de la que será retirado el dinero. Puede ser la cuenta del centro de costos principal o de un centro de costos adicional.

La orden debe contener una firma generada a partir de la llave privada y toda la información de la orden.



Para realizar órdenes exitosas la cuenta debe contar con saldo suficiente, el saldo en ambiente UAT solo puede ser otorgado por medio de una solicitud, el horario para realizar fondeos es de 9:00 a 6:00 CDT. El integrador debe especificar a que cuenta se realizará el depósito, puede ser una cuenta operativa o una cuenta virtual.

Las órdenes de pago realizadas en el entorno de pruebas UAT deben realizarse a la cuenta beneficiaria 884180884987878785 del banco “GEMTRANSFER”.

Para firmar una orden se deberá realizar una secuencia de caracteres llamada cadena original la cual debe ser formada de acuerdo a los siguientes criterios:

1. La cadena original deberá empezar y terminar con un doble símbolo pipe (||)
2. Cada uno de los campos de la cadena original deberán ser separados por el símbolo pipe (|)

La cadena original siempre contendrá los siguientes campos ubicados en el orden que le corresponda:

- A. Nombre del beneficiario (Obligatorio)
 - B. RFC/Curp del beneficiario (Opcional, si no se cuenta con el, sustituir con cadena vacía)
 - C. Banco del beneficiario (Obligatorio)
 - D. Cuenta del beneficiario (Obligatorio)
 - E. Tipo de cuenta del beneficiario (Obligatorio, el valor debe ser 40 o 101)
 - F. Cuenta ordenante (Obligatorio)
 - G. Referencia numérica (Obligatorio)
 - H. Fecha de operación (Obligatorio, debe especificarse en el formato tYYYY-MM-DD. Es necesario transformar el timestamp del JSON del request, ejemplo: 1635830800175 -> 2021-11-02, es importante observar que la fecha debe estar formateada en timezone UTC)
 - I. Tipo de pago (Obligatorio)
 - J. Concepto (Obligatorio)
 - K. Monto (Obligatorio, revisar que el formato sea decimal, ej: 1000.00)
4. En caso de que no se cuente con alguno de los datos anteriores de los cuales su presencia no sea obligatoria, podrá ser omitido siempre y cuando se coloque la posición que le corresponda dentro de la cadena original.

||nombreBeneficiario||BancoDelBeneficiario|CuentaDelBeneficiario...

5. Algunas características especiales a tomar en consideración para la construcción de la cadena original son las siguientes:
 - a. No se deben usar separadores de miles.
 - b. Siempre deben ser incluidos dos dígitos decimales.
 - c. Se usará el carácter punto (.) como separador decimal.
6. La codificación de la cadena deberá ser UTF-8

Composición de cadena original:

||nombre del beneficiario|rfc/curp del beneficiario|banco del beneficiario|cuenta del beneficiario|tipo de cuenta de beneficiario|**cuenta ordenante**|referencia numérica|día de pago|tipo de pago|concepto|monto||

Ejemplo real de cadena original:

||Juan Perez
Perez|JPPM801|28L56|40012|012180026050203448|40|684180009000000004|1234568|2021-10-19|1|Payment|1.11||

La cadena original se utilizará para generar un hash basado en el algoritmo de digestión SHA-256 (<https://es.wikipedia.org/wiki/SHA-2>) que se debe utilizar para realizar la generación de la firma a través del algoritmo asíncrono RSA (<https://es.wikipedia.org/wiki/RSA>) con el contenido de la llave privada otorgada.

Ejemplo de generación de firma digital en PHP:
<https://www.php.net/manual/en/function.openssl-sign.php>

Ejemplo del cuerpo del request.

```
{
  "amount": 123.00,
  "beneficiaryAccount": "012180026050203448",
  "beneficiaryAccountType": 40,
  "beneficiaryBank": "40012",
  "beneficiaryName": "Juan Perez Perez",
  "beneficiaryUid": "",
  "concept": "Payment",
```

```

    "numericalReference": 1234568,
    "payerAccount": "684180009000000004",
    "payerAccountType": 40,
    "payerName": "TEST",
    "paymentDay": 1716114930765,
    "paymentType": 1,
    "sign": "OH15jjW810eFruIRhdoc2IOuy3/wH..."
}

```

Ejemplos de código para la generación de firma digital

Java

```

import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Base64;

class SignerExample{

    public static void main(String[] args) throws NoSuchAlgorithmException,
    InvalidKeySpecException, InvalidKeyException, SignatureException {
        String
        privateKeyString="MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBCkwggSjAgEAAoI
        BAQCaLcr/xXKglvNgeJ/GwYGCT3+D1nnZ1+zZsSuwwZ9lwsXtJPoSv/H6gm2BUuKv
        Nk/tu9IlePPI3nNDiOTNViZwcJtn9nZapfzccMDTerK4J3xen4iXjKIPKE5+Oy+4AFr2h
        jyL3XmW2NupR6o4eigVutAKGKto3AA3/OoXOc0OhQn6hl+8DCBqDsigM3S/BwK
        hzpQq0xyfIG+EYYw2459VLjj8B24H7Qb/1hZLp7to/ISfmvGnj61SL5TGkGcljmc9h5Tt
        D/jjuTfvRQbDrEET109Z5JrfN4PU0dZjdboVk9iRDGxGO7lgtXUmySYJqCrP/E5JI+Cil
        uCHw4iaMelDAgMBAAECggEAQCdp3r/EEIo6hxvqMAP45cY6oxstnM3SSksEiQoo
        Q1h4j1Lhk8e24qw+8J8SoCGTZLLQIKHvqujDQYjG3AJLAGJ65+mQGewTqY5pThY01
        /7mK2FqrtTf4jq9Xp02H/RV5Vq5+uDvZStbS7b8R3/dCh2vz2Owf3N7y9A7f+qXPsdI
        ATPljKoYMjamg5dE/hwGJZ3Ua/J8dM0R8auh3RRKqtRLtWAPnMvczRtirioXWOa/z
        SA8dptnOpufv9rdEI4qsVR7KKTha3ENNGsCURJdjAA5kckVv9YXbZpk6aa164JT7K
        NLddTwR9NEw8zo+QIU3zSlohytO3oqNHaTN+WfsQKBgQD1NclM4llwI30JDrVvb
        mc4iwBgwu24qAW8SWy6WRZnW5pthuH06gmoVDZfL5aXWk4adALJQMASIRe
        VKsBF8satOz6gsvPnci2P15nhCafRqrym0TOM5dYJTSerj5WimoWLKYq/KB0M4Hg

```

```
O5IPEN0aZ9bwYJfXcF658txuqJeqoXwKBgQCg9pGjuMLLi3IRVQTz5bo0J1Pvr41xk
LChx/sdJfjayHPL97Ybqhs61pRu8SO7nsyGCKwQOuDTGq/ZWYMRMBw42+W9v+H
1ULQJNNMKk7DSyMjldZwf5VSRdRQ0cKuxsr2mPVNzZF9wDbhQseSIRiCfjGnPOS
5v3DqXPAYaz+S5nQKBgDP7O/o9G0W9mLcD1DiUGfOIm+XnTMe5bsXUduUmH5a
3l4wibOljfvqdJ42UJANf1HDNKQ4K2Oy+8SXhEzOA3Nc6WPNUHEBzNu9oUCmhh
17uJ+HhNWFoE7CTF5bJ8Hmw8iBnJQ+S7F2fdDQMA1bkNOZMbpxb57vJIWbAlm
+CbDVXAoGARK0vMoWQRzw17QMzW8S2j63/+BIU7VFoq34hvCwJY3+HOI12G5O
WjqIjLrqgHlsAtGda0onshuWqBbhVXa4DPJdGDJGkHQDT7+Gj8a1WEnksjgqCFH
HKhU1I058HAohatmUQB/nZZnvUGknsUeBp5ukH9CxXjc6PCrVJFwOAeVkcGyEA
u0sorozfeNk+eK2qtI3You1JgV47AdXDhZnQaaX6NdSAitPZJKZa3PcRsbFPQuPlN/S
5GMO89gdNhg/NrULCQyZYd1S60z4w+1iG44ge3qMbg2MJfdsZZX5jrnpGfctPrqM
DW8l+wr9UOecEnTnQBeJfox6pexded/HQdwoieo=";
```

```
String originalString = "|Juan Perez
Lopez|JPL8012123M2|684|684180999000000000|40|6841800170000000009|12345|
2021-04-20|1|Pago de ejemplo|1.00|";
```

```
byte[] originalStringBytes = originalString.getBytes();
```

```
KeyFactory kf = KeyFactory.getInstance("RSA");
PKCS8EncodedKeySpec keySpecPKCS8 = new
PKCS8EncodedKeySpec(Base64.getDecoder().decode(privateKeyString));
PrivateKey privateKey = kf.generatePrivate(keySpecPKCS8);

Signature signer = Signature.getInstance("SHA256withRSA");
signer.initSign(privateKey);
signer.update(originalStringBytes);
String signString = Base64.getEncoder().encodeToString(signer.sign());
System.out.println(signString);
}
}
```

PHP

```
<?php
$private_key = <<<EOD
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBAwggSjAgEAAoIBAQCAlCr/xXKglvNge
J/GwYGCT3+D1nnZl+zZsSuwwZ9lwsXtJPoSv/H6gm2BUuKvNk/tu9IlePPI3nNDiOTN
ViZwcJtn9nZapfzccMDTerK4J3xen4iXjKIPKE5+Oy+4AFr2hjyL3XmW2NupR6o4eigV
```

```
utAKGKto3AA3/OoXOc0OhQn6hl+8DCBqDsigM3S/BwKhzpQq0xyfIG+EYYw2459VLj
j8B24H7Qb/1hZLp7to/ISfmvGnj61SL5TGkGcljmc9h5TtD/jjuTfvRQbDrEET109Z5JrfN4
PU0dZjdboVvk9iRDGxGO7lgtXUmySYJqCrP/E5JI+CiluCHw4iaMeIDAgMBAAECggEA
QCdp3r/EEIo6hvxqMAP45cY6oxstnM3SSksEiQooQ1h4j1Lhk8e24qw+8J8SoCGTZLLQ
IKHvqujDQYjG3AJLAGJ65+mQGewTqY5pThY0l/7mK2FqrtTf4jq9Xp02H/RV5Vq5+uD
vZStbS7b8R3/dCh2vz2Owf3N7y9A7f+qXPsdIATPljKoYMjamg5dE/hwGJZ3Ua/J8dMO
R8auh3RRKqtRLtWAPnMvzRtirioXWOa/zSA8dptnOpufv9rdEI4qsVR7KKTha3ENN
gsCURJdjAA5kckVv9YXbZpk6aa164JT7KNLddTwR9NEw8zo+QIU3zSlohytO3oqNHa
TN+WfsQKBgQD1NclM4llw130JDrVvbm4iwBgwu24qAW8SWy6WRZnW5pthuH06
gmoVDZfL5aXWk4adALJQMASIReVKsBF8satOz6gsVpnci2P15nhCafRqym0TOM5d
YJTSerj5WimoWLKYq/KB0M4HgO5IPEN0aZ9bwYJfXcF658txuqJeqoXwKBgQCg9p
GjuMLLi3IRVQTz5bo0J1Pvr41xkLChx/sdJfjayHPL97Ybqhs61pRu8SO7nsyGCKwQOuD
TGq/ZWYMRMBw42+W9v+H1ULQJNNMKk7DSyMjldZwf5VSRdRQ0cKuxsr2mPVNz
ZF9wDbhQseSIRiCfjGnPOS5v3DqXPAYaz+S5nQKBgDP7O/o9GOW9mLcD1DiUGfOI
m+XnTMe5bsXUduUmH5a3l4wibOljfvqdJ42UJANfIHDNKQ4K2Oy+8SXhEzOA3Nc6
WPNUHEBzNu9oUCmhh17uJ+HhNWFoE7CTF5bJ8Hmw8iBnJQ+S7F2fdDQMA1bkN
OZMbpxb57vJIWbAlm+CbDVXAoGARK0vMoWQRzw17QMzW8S2j63/+BIU7Vfoq34
hvCwJY3+HOI2G5OWjqljLrqgHlsAtGda0onshuWqBbhVXa4DPJdGDJGkHQDT7+Gj
8a1WEnksjgqCFHHKhU1I058HAohatmUQB/nZZnvUGknsUeBp5ukH9CxXjc6PCRVJF
wOAeVkcGyEAu0soroZfeNk+eK2qt13You1JgV47AdXDhZnQaaX6NdSAitPZJKZa3PcR
sbfPQuplN/S5GMO89gdNhG/NrULCQyZYd1S60z4w+1iG44ge3qMbg2MJfdsZZX5jrnP
GfctPPrqMDW8l+wr9UOecEnTnQBeJfox6pexded/HQdwoieo=
```

-----END PRIVATE KEY-----

EOD;

```
$originalString="||Juan Perez
Lopez|JPL8012123M2|684|684180999000000000|40|684180017000000009|12345|20
21-04-20||Pago de ejemplo|1.00||";
```

```
openssl_sign($originalString, $signature, $private_key, OPENSSL_ALGO_SHA256);
```

```
echo base64_encode($signature);
```

Node.js

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

```
crypto.generateKeyPair('rsa', {
  'modulusLength': 4096,
```

```
'publicKeyEncoding': {
  'type': 'spki',
  'format': 'pem',
},
'privateKeyEncoding': {
  'type': 'pkcs8',
  'format': 'pem'
}
},()=>{});
```

```
const privateKey = crypto.createPrivateKey({
  'key': "-----BEGIN PRIVATE KEY-----\n" +
```

```
"MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBCwggSjAgEAAoIBAQCALcr/xXKgIvNgeJ/G
wYGCT3+D1nnZ1+zZsSuwwZ9lwsXtJPoSv/H6gm2BUuKvNk/tu9IlePPI3nNDiOTNViZwcJt
n9nZapfzccMDTerK4J3xen4iXjKIPKE5+Oy+4AFr2hjyL3XmW2NupR6o4eigVutAKGKto3A
A3/OoXOc0OhQn6hl+8DCBqDsigM3S/BwKhzpQq0xyfIG+EYYw2459VLjj8B24H7Qb/1hZ
Lp7to/ISfmvGnj61SL5TGkGcljmc9h5TtD/jjuTfvRQbDrEET109Z5JrfN4PU0dZjdboVk9iRD
GxGO7lgtXUmySYJqCrP/E5JI+CiluCHw4iaMelDAgMBAAECggEAQCdp3r/EElo6hvxvqMA
P45cY6oxstnM3SSksEiQooQ1h4j1Lhk8e24qw+8J8SoCGTZLLQIKHvqujDQYjG3AJLAGJ65
+mQGewTqY5pThY01/7mK2FqrTf4jq9Xp02H/RV5Vq5+uDvZStbS7b8R3/dCh2vz2Owf3N
7y9A7f+qXPsdIATPljKoYMjamg5dE/hwGJZ3Ua/J8dM0R8auh3RRKqtRLtWAPnMvzcRtiri
oXWOa/zSA8dptnOpufv9rdEI4qsVR7KKTha3ENNGsCURJdjAA5kckVv9YXbZpk6aa164JT
7KNLddTwR9NEw8zo+QIU3zSl0hytO3oqNHaTN+WfsQKBgQD1NclM4Ilw130JDrVvbmC4
iwBgwu24qAW8SWy6WRZnW5pthuH06gmoVDZfL5aXWk4adALJQMASiReVKsBF8sat
Oz6gsvPnci2P15nhCafRqym0TOM5dYJTserj5WimoWLKYq/KBOM4HgO5IPEN0aZ9bw
YJfXcF658txuqJeqoXwKBgQCg9pGjuMLLi3IRVQTz5bo0J1Pvr41xkLChx/sdJfjayHPL97Yb
qhs61pRu8SO7nsyGCKwQOuDTGq/ZWYMRMBw42+W9v+H1ULQJNNMKk7DSyMjldZwf
5VSRdRQ0cKuxsr2mPVNzZF9wDbhQseSIRiCfjGnPOS5v3DqXPAYaz+S5nQKBgDP7O/o9
G0W9mLcD1DiUGfOIm+XnTMe5bsXUduUmH5a3l4wibOljfvqdJ42UJANf1HDNKQ4K2Oy
+8SXhEzOA3Nc6WPNUHEBzNu9oUCmhhh17uJ+HhNWFOE7CTF5bJ8Hmw8iBnJQ+S7F2f
dDQMA1bkNOZMbpXb57vJIWbAlm+CbDVXAoGARK0vMoWQRzw17QMzW8S2j63/+BIU
7VFoq34hvCwJY3+HOI12G5OWjqIjLrqgHlsAtGda0onshuWqBbhVXa4DPJdGDJGkHQD
T7+Gj8a1WEnksjgqCFHHKhU1I058HAohatmUQB/nZZnvUGknsUeBp5ukH9CxXjc6PCRV
JFwOAeVkcGyEAu0sorozfeNk+eK2qtI3You1JgV47AdXDhZnQaaX6NdSAitPZJKZa3PcRs
bfPQupIN/S5GMO89gdNhg/NrULCQyZYdIS60z4w+liG44ge3qMbg2MJfdsZZX5jrnpGfct
pPrqMDW8I+wr9UOecEnTnQBeJfox6pexded/HQdwoieo=\n" +
  "-----END PRIVATE KEY-----",
  'format': 'pem',
  'type': 'pkcs8'
});
```

```
const originalString = "||Juan Perez
Lopez|JPL8012123M2|684|684180999000000000|40|6841800170000000009|12345|2021-
04-20||Pago de ejemplo|1.00||";

const signer = crypto.createSign('RSA-SHA256');
signer.update(originalString);

console.log(signer.sign(privateKey).toString('base64'));
```

C#

```
using System;
using System.Security.Cryptography;
using System.Text;

class SignerExample {
    static void Main(string[] args) {
        //Eliminar header (-----BEGIN PRIVATE KEY-----) y footer (-----END
PRIVATE KEY-----) de llave
        string privateKeyString =
"MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBAwggSjAgEAAoIBAQCALcr/xXKgIvNgeJ/GwYGCT3
+D1nnZ1+zZsSuwwZ9IwsXtJPoSv/H6gm2BUuKvNk/tu9IIePP13nNDiOTNViZwcJtn9nZapfzcc
MDTerk4J3xen4iXjKIPKE5+Oy+4AFr2hjyL3XmW2NupR6o4eigVutAKGKto3AA3/OoX0c00hQn6
hl+8DCBqDsigM3S/BwKhzpQq0xyfIG+EYYw2459VLjj8B24H7Qb/1hZLp7to/ISfmvGnj61SL5T
GkGcIjmc9h5TtD/jjuTfvRQbDrEET109Z5JrFN4PU0dZjdboVk9iRDGxG07IgtXUmySYJqCrP/E
5JI+CiluCHw4iaMe1DAgMBAAECggEAQCdp3r/EEIo6hxvqMAP45cY6oxstnM3SSksEiQooQ1h4j
1Lhk8e24qw+8J8SoCGTZLLQlKHvqujDQYjG3AJLAGJ65+mQGewTqY5pThY01/7mK2FqrTtf4jq9
Xp02H/RV5Vq5+uDuVZStbS7b8R3/dCh2vz20wf3N7y9A7f+qXPdIATPljKoYMjamg5dE/hwGJZ3
Ua/J8dM0R8auh3RRKqRLtWAPnMvzcRtirioXW0a/zSA8dptnOpufv9rdEI4qsVR7KKTha3ENNg
sCURJdjAA5kckVv9YXbZpk6aa164JT7KNLddTwR9NEw8zo+QlU3zSl0hytO3oqNHaTN+WfsQKBg
QD1NclM4Ilw130JDrVvbmC4iwbgu24qAW8Swy6WRZnW5pthuH06gmoVDZfL5aXWk4adALJQMAS
IReVksBF8satOz6gsvPnci2P15nhCafRqrym0TOM5dYJTSerj5WimowLKYq/KB0M4Hg05IPEN0a
Z9bwYJfXcF658txuqJeqoXwKBgQCg9pGjuMLLi3IRVQTz5bo0J1Pvr41xkLChx/sdJfjayHPL97
Ybqhs61pRu8S07nsyGCKwQOuDTGq/ZWYMRMBw42+W9v+H1ULQJNNMKk7DSyMjldZwf5VSRdRQ0c
Kuxsr2mPVNzZF9wDbhQseSIRiCfjGnPOS5v3DqXPAYaz+S5nQKBgDP70/o9G0W9mLcd1DiUGf0l
m+XnTMe5bsXUduUmH5a3l4wibOIjfvqdJ42UJANf1HDNKQ4K20y+8SXhEzOa3Nc6WPNUHEBzNu9
oUCmhh17uJ+HhNWFOE7CTF5bJ8Hmw8iBnJQ+S7F2fdDQMA1bkNOZMbpXb57vJlWbA1m+CbDVXAo
GARK0vMoWQRzw17QMzW8S2j63/+BIU7VFOq34hvCwJY3+H0I12G50WjqIjLrqgHlsAtGda0onsh
```

```
uWqBbhVXa4DPJdGDJGkHQDT7+Gj8a1WEnksjgqCFHHKhU1I058HAohatmUQB/nZZnvUGknsUeBp
5ukH9CxXjc6PCrvJFw0AeVkcGyEAu0sorozfeNk+eK2qtI3You1JgV47AdXDhZnQaaX6NdSAitP
ZJKZa3PcRsbfpQuplN/S5GMo89gdNhg/NrULCQyZYd1S60z4w+1iG44ge3qMbg2MJfdsZZX5jrn
pGfctpPrqMDW8I+wr9U0ecEnTnQBeJfox6pexded/HQdwoieo=";
    string originalString = "||Juan Perez
Lopez|JPL8012123M2|684|684180999000000000|40|684180017000000009|12345|2021-
04-20|1|Pago de ejemplo|1.00||";
    byte[] originalStringBytes =
Encoding.UTF8.GetBytes(originalString);

    RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();

    rsa.ImportPkcs8PrivateKey(Convert.FromBase64String(privateKeyString), out
    _);

    SHA256 sha256 = SHA256.Create();
    byte[] signatureBytes = rsa.SignData(originalStringBytes, sha256);
    string signString = Convert.ToBase64String(signatureBytes);
    Console.WriteLine(signString);
}

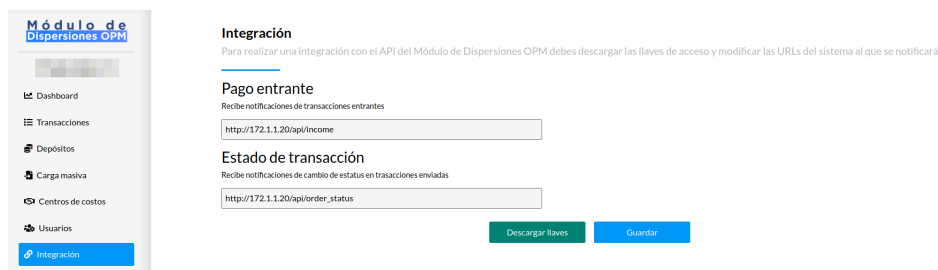
}
```

Recepción de Webhooks

Existen dos webhooks a través de los cuales MDO enviará información al servidor del integrador. El primero es el webhook de órdenes de entrada, es decir: notifica los depósitos a los centros de costo. El segundo es el webhook de cambios de estado de órdenes enviadas, es decir: notifica cuando una orden que ha sido enviada, es liquidada, rechazada o cancelada.

La comunicación general de los eventos se realiza vía HTTP(s) emitiendo una llamada a la dirección de recepción definida con un mensaje en estructura JSON para todos los eventos.

Los webhooks se configuran en la sección de integración del dashboard de MDO.



Una vez establecida la comunicación con el servicio, se tomará en cuenta las acciones a implementar de acuerdo al código HTTP de respuesta de la comunicación:

- HTTP 2xx:
 - Se recibió correctamente el mensaje
- HTTP 4xx.
 - El mensaje es rechazado ó existe algún motivo por el cual no se pudo recibir la operación.
- HTTP 5xx:
 - Se interpreta como un error interno, la transacción es rechazada.

Webhook de órdenes de entrada

Este webhook (Llamada HTTP tipo post) informará al servidor del integrador que un depósito fue recibido a través de SPEI. El integrador deberá responder si acepta o no el depósito. En caso de rechazar el depósito, se informará del rechazo al Banco de México y el dinero será regresado al remitente.

Especificación de request

El contenido del mensaje de la notificación contendrá la información de la orden entrante que será descrita a continuación:

- A. Tipo
- B. Datos
 - a. Nombre del beneficiario (string<40>)
 - b. RFC/Curp beneficiario (string<13>)
 - c. Cuenta del beneficiario (string<40>)
 - d. Banco del beneficiario (string<5>)
 - e. Tipo de cuenta del beneficiario (int<32>)
 - f. Nombre del ordenante (string<40>)
 - g. RFC/Curp del ordenante (string<13>)
 - h. Cuenta del ordenante (string<18>)

- i. Banco del ordenante (string<5>)
- j. Tipo de cuenta del ordenante (int<32>)
- k. Monto (decimal<18,2>)
- l. Concepto (string<40>)
- m. Llave de rastreo (string<30>)
- n. Referencia numérica (int<7>)
- o. Fecha de operación (string<10>)
- p. Timestamp de recepción (int<32>)
- q. Firma (string<500>). Creada con la siguiente cadena original (amount debe ser convertido a decimal, ejemplo 1000->1000.00):
 ||beneficiaryName|beneficiaryUid|beneficiaryAccount|beneficiaryBank|
 beneficiaryAccountType|payerName|payerUid|payerAccount|payerBank|
 payerAccountType|amount|concept|trackingKey|numericalReference||

Ejemplo del contenido de una llamada al webhook de depósitos.

```
{
  "data": {
    "beneficiaryBank": "684",
    "payerBank": "012",
    "amount": 1,
    "payerUid": "ABCD900824B52",
    "beneficiaryAccountType": 40,
    "concept": "TEST",
    "sign":
      "ejxxu44ByH6HdCnqvIxT2TvXCIsvY3NHNR31Znf/6PmRHSM4nh01ppRPLlpOHMU0BYud0u99oV
      xT+CgH15Jrw2VwjVqkM8I30XgwEni69+30U1Sv/wDwdA0n/bFgwOSTzzq4iXPE4Sddpjivnkjx1
      J93JZEYHUCvL5cPokPMAj+Sq9AP5v2dWHct0+WpC8NewTuSJ3s7CfJhRK2grUKrRpsDqq5gKFvJ
      9uuttk3gdKtURphAiPusBcQh9koYAWd+I3eC0qtah1tmxbfLJ31/8izMe9oS8JynRkmm7C7aGEs
      ppi/4ydqEmghQvFc0MrgH/9xcAKd3NVW7b1b6q3a6VA==",
    "payerAccount": "012180029064952818",
    "numericalReference": 211210,
    "payerAccountType": 102,
    "trackingKey": "MBAN01002111030095198330",
    "beneficiaryName": "TEST CLIENT",
    "beneficiaryAccount": "684180001000000002",
    "beneficiaryUid": "WOWN9907552B5",
    "payerName": "JUAN PEREZ PEREZ",
    "operationDate": "2021-12-30",
  }
}
```

```
"receivedTimestamp": 1640880657000
},
"type": "supply"
}
```

Especificación de response

La orden de pago será aceptada o rechazada de acuerdo a la respuesta obtenida al realizar la llamada de webhook.

Las respuestas deberán tener la siguiente estructura:

Nombre campo	Obligatoriedad	Tipo de dato	Descripción
returnCode	Mandatorio	Numérico, máximo 10 caracteres.	0 =Orden aceptada <>0 = Orden rechazada
errorDescription	Opcional	Alfanumérico, máximo 100 caracteres	Una breve descripción del error o la causa del rechazo.
metadata	Opcional	Objeto	{ "internalTxId": "1266", "accountId": "723489" }
cepBeneficiaryName	Opcional	Alfanumérico, máximo 40 caracteres	Nombre del beneficiario asociado a la cuenta virtual. Este dato se utilizará para generar el CEP emitido por Banxico. Para hacer uso de esta funcionalidad debe aprobarse previamente por OPM.

cepBeneficiaryUid	Opcional	Alfanumérico, 12 caracteres, 13 caracteres o valor "ND"	RFC del beneficiario asociado a la cuenta virtual. Este dato se utilizará para generar el CEP emitido por Banxico. Para hacer uso de esta funcionalidad debe aprobarse previamente por OPM.
-------------------	----------	---	---

Catálogo de códigos de retorno(returnCode)

Código de respuesta / returnCode	Descripción	Acción
0	Transacción Exitosa	Acción: Concluye el flujo de la operación, y se notifica ante Banxico el CDA (confirmación de abono para generar CEP).
4	Excede el límite de saldo autorizado de la cuenta	Acción: Devolución ante Banxico con la causa de devolución: "Excede el límite de saldo autorizado de la cuenta"
6	Cuenta inexistente	Acción: Devolución ante Banxico con la causa de devolución: "Cuenta inexistente"
7	Error en datos	Acción: Devolución ante Banxico con la causa de devolución: "Tipo de pago erróneo"
12	Transacción duplicada	Acción Devolución ante Banxico. "Clave de rastreo repetida por Participante Emisor y día de operación"
13	Beneficiario no reconoce el pago	Acción: Devolución ante Banxico con la causa de devolución: "Beneficiario no reconoce el pago"
99	Error interno	Acción: Devolución ante Banxico. "Cuenta inexistente"

Ejemplos de respuesta exitosa:

```
{
  "returnCode": 0
}
```

```
{
  "returnCode": 0,
  "cepBeneficiaryName": "Joel García Perez",
  "cepBeneficiaryUid": "RACJ950315LU7"
}
```

```
{
  "returnCode": 0,
  "metadata": {
    "internalTxId": "1266",
    "accountId": "723489"
  }
}
```

Ejemplos de respuesta con rechazo de transacción

```
{
  "returnCode": 4,
  "errorDescription": "Excede límite de saldo"
}
```

Consideraciones:

- Si el código HTTP de respuesta es diferente a 200, la transacción será rechazada con el motivo "Cuenta inexistente"

- Si el código HTTP de respuesta del integrador es un código 200, el cuerpo del mensaje coincide con la especificación de respuesta y el return code es diferente a 0, la orden será rechazada y se especificará el motivo de devolución especificado en la respuesta.
- Si el código HTTP de respuesta del integrador es un código 200 y el cuerpo del mensaje no cuenta con la llave “returnCode”, la orden será aceptada.

Webhook de órdenes de salida

Este webhook (Llamada HTTP tipo POST) informará cuando exista un cambio de estado en las órdenes enviadas. A través de este webhook se informará del cambio de estado de todas las órdenes que hayan sido enviadas.

El contenido del mensaje de la notificación contendrá el id de la orden que se está actualizando y el estado actual de la misma:

- A. Tipo
- B. Datos
 - a. Id de orden
 - b. Estatus de orden
 - c. Detalle
 - i. Sólo se envía en caso de rechazo (estado “canceled o “returned”)
 - d. Firma. Creada con la siguiente cadena original: ||id|status|detail| (En caso de no existir “detail”, sustituir por cadena vacía)

Estatus posibles de la tracción.

- pending (La transacción se encuentra en la cola de envíos, si la cuenta no tiene suficiente saldo, la transacción esperará a tener el saldo suficiente para enviarla)
- sent (La transacción fue enviada al sistema SPEI y está en espera de una respuesta)
- scattered (La transacción ha sido liquidada)
- canceled (La transacción fue cancelada)
- returned (El banco receptor rechazó la transacción)

```
{
  "data": {
    "id": "98588b7d-59f2-41cb-ad51-7a18a0a5e2fe",
    "status": "sent"
  },
  "sign":
    "EoCts2MZjTTsPAOkOU/+rrj4PQn2rV2HT4B1Z7brZ0X0X89Pos2G7V6tj59ViwF43Uczp"
```

```
6HeVS+MvS+1691GPPErATmRMiybycXRRWmU7Z18X6oN5MOZYa8sjJ3l/45TuiUvsVs
DDIBQ0v4bSXOOdrplINEbDnspGpgx6lYLPwClzopD6UWi0T3kNEeNAMsxhwFBrr
0B6VKWBoCihBCYgH6EmL0s/rnlvHNqE/AACgwX+RzM/D9VkY9euTVHlqdechHC4eA
ZJJq0k4AnaxP8NVp2SVePX/CgRDvYJjgxBhlOIKeHcg8iaXiHwtQo/DWYSfjC2Cv7Ahf
YrmrkUFNOng==",
  "type": "orderStatus"
}
```

Devoluciones

Existe la posibilidad de realizar una orden de pago, que el banco beneficiario acepte la transacción y minutos después la rechacé por algún proceso de validación interna. A continuación se describe el escenario.

- MDO envía la transacción a EF-SPEI si hay suficiente dinero en la cuenta
- MDO informa al integrador por medio del webhook enviando el estado “sent”
- El banco beneficiario acepta la transacción y MDO envía el estado “scattered”
- El banco beneficiario cambia de opinión y rechaza la transacción
- MDO modifica la propiedad errorDetail de la transacción de salida para describir la razón de la devolución.
- MDO crea una nueva transacción entrante con la misma llave de rastreo y monto de la transacción original.
- MDO envía actualización de orden por medio de webhook

El detalle de la devolución es guardado en la propiedad “detail”.

Conexión de endpoints

Existen múltiples endpoints que pueden ser integrados con la interfaz MDO, para ver la lista exhaustiva y los detalles de los parámetros de cada endpoint referirse a la documentación del API.

Diagrama de flujo de transacciones recibidas

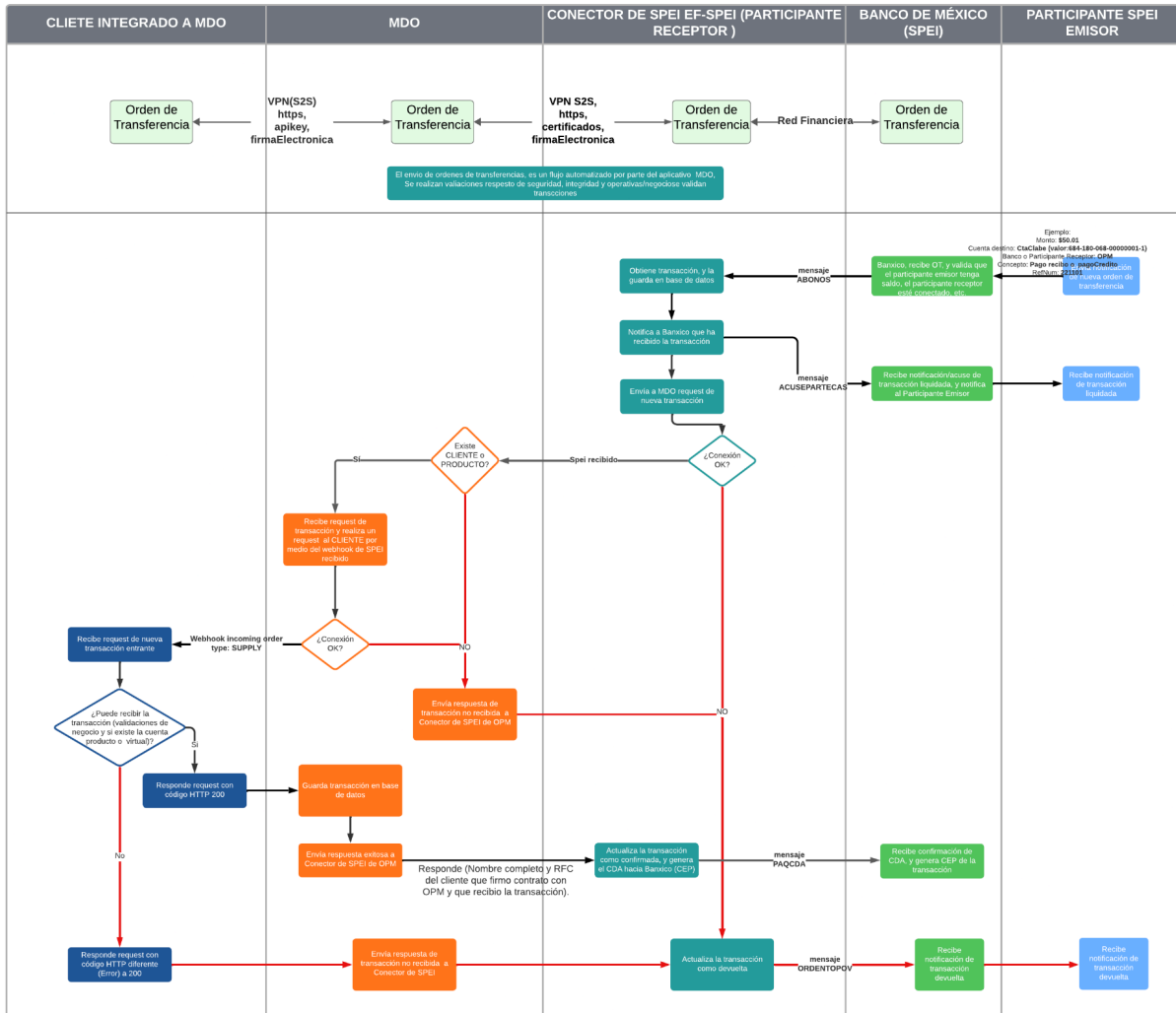
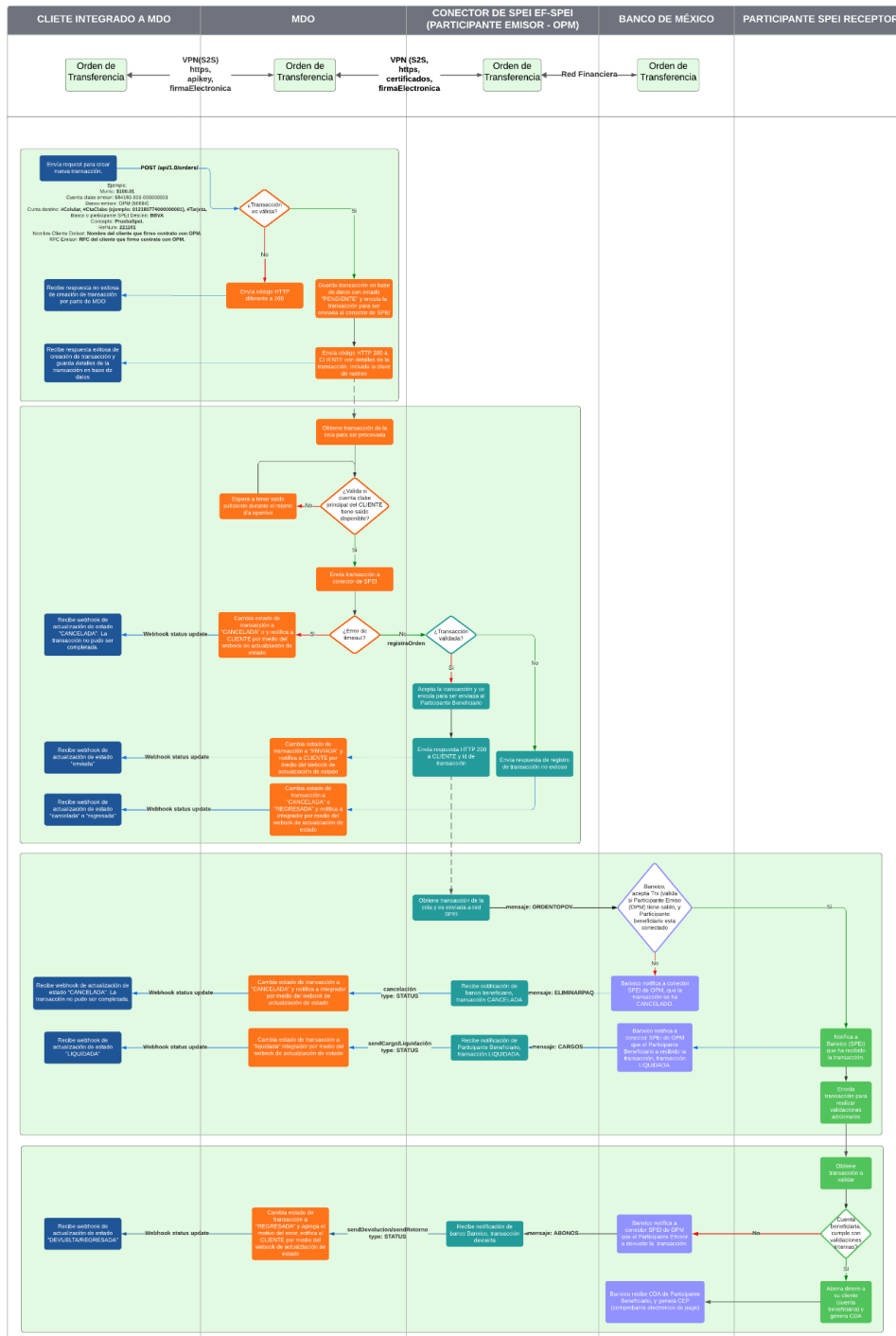


Diagrama de flujo de transacciones enviadas



Flujo: Envío de transacciones desde API - MDO hacia otros Participantes de SPEI

Fecha de creación: 20/Nov/2023



Control de cambios

Versión	Fecha	Modificaciones
1.0	04/13/2021	Creación del documento
1.1	04/23/2021	Se agregan ejemplos de código para la generación de firma de órdenes
1.2	07/05/2021	Se agrega especificación de construcción de firma de transacciones de entrada
1.3	27/10/2021	Se agregan estados faltantes de transacciones de salida. Se modifica ejemplo de cadena original.
1.4	05/11/2021	Se agrega algoritmo de generación de dígito verificador. Se agrega proceso de devoluciones.
1.5	29/11/2021	Especificación de construcción de cadena original en webhook de transacción entrante.
1.6	03/03/2022	Se agrega especificación de sustitución de acentos.
1.7	01/02/2023	Se agrega especificación de retornos personalizados.
1.8	07/03/2023	Se agrega ejemplo de generación de firma en C#
1.9	22/03/2023	Se agrega la llave beneficiaryUid en la documentación de webhook de orden entrante.
2.0	16/02/2024	Se realiza modificación en especificación de webhook de notificación de estado. Se especifica en que estados se envía la llave "detail".
2.1	29/04/2024	Se agrega especificación de datos de retorno en webhook de pago entrante para generar CEP.
2.2	09/06/2024	Corrección en documentación de generación de cadena.
2.3	05/11/2024	Se agregan diagramas de flujo de transacciones enviadas y recibidas.
2.4	22/04/2025	Se agrega tipo de devolución 13

2.5	21/07/2025	Comunicación por Internet en vez de VPN
-----	------------	---