VPOS 2.O

Colaborando para construir tu negocio en internet

Especificaciones Técnicas
Single Buy
Versión 1.10

Bancard

Control de cambios

Sección/hoja	Versión	Fecha	Descripción
Pago ocasional/Pág. 8	Versión 1.3	09/08/2018	Se agrega sección “Tarjetas procesadas”
Pago con token/Pág. 18	Versión 1.3	09/08/2018	Se agrega sección “Tarjetas procesadas”
Pagos con débito	Versión 1.3	09/08/2018	Se elimina la sección de pagos con débito.
Catastro de tarjeta/Pág. 26	Versión 1.3	09/08/2018	Se agrega la sección “Flujo de catastro”
Catastro de tarjeta/Pág. 27	Versión 1.3	09/08/2018	Se agrega “Recomendación para el comercio”
	Versión 1.4	18/09/2018	Cambio de pago anónimo por pago ocasional
Código de errores - Pag 50	Versión 1.5	15/10/2018	Anexo Código de errores
Catastro de tarjeta – Pag 22	Versión 1.6	14/01/2019	Recomendación para el comercio.
Single Buy Zimple – Pag 17	Versión 1.7	05/04/2019	Integración Zimple-vPOS
	Version 1.8	30/08/2019	Recomendación para aplicativos
Pag 51	Version 1.8	30/08/2019	Paso a producción
Pag 42	Version 1.9	11/09/2019	Reversas operativas
Pag 6	Version 1.10	25/08/2020	Se agrega la opción de uso de SHA256
Pag 11	Version 1.10	25/08/2020	Se agrega nueva funcionalidad del additional_data para soportar múltiples promociones
Pag 21	Version 1.10	25/08/2020	Se agrega datos de prueba de zimple

Contenido

Introducción

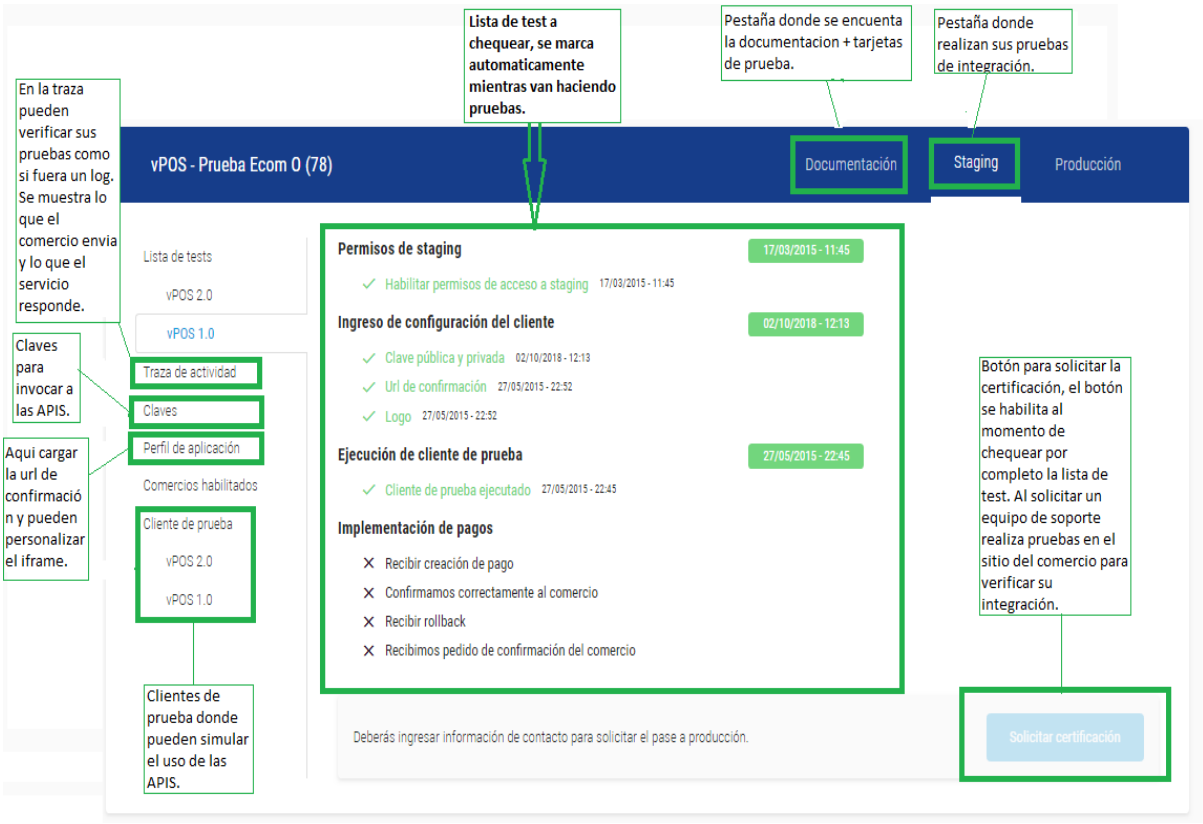
El siguiente documento presenta la información técnica necesaria para comunicarse con el servicio de pasarela de pagos de eCommerce de Bancard o VPOS.

El producto por construir por el comercio podrá ser Web o Mobile. A continuación, se detallan las distintas interacciones con servicios de la API REST, así como redirecciones necesarias a una interfaz de Bancard para solicitar los datos de la tarjeta de crédito.

Adicionalmente a este documento el comercio o desarrollador de la integración con VPOS deberá contar con un acceso al portal de comercio de Bancard: https://comercios.bancard.com.py

En el mismo se le brindará acceso para:

- Acceder al ambiente de staging y producción de vpos
- Acceder a la clave pública y privada. Adicionalmente podrá regenerar ambas claves.
- Modificar información del perfil: Nombre, logo y url de confirmación
- Traza de interacciones entre VPOS y el producto desarrollado por el comercio.
- Checklist con pasos para validar la integración.
- Documentación y ejemplos de códigos en distintos lenguajes.



El vPOS 2.0 cuenta con dos formas de pago, que son los siguiente:

1- **Pago ocasional**: El usuario carga siempre todos los datos de su tarjeta en el formulario realizando así el pago.

   Servicios ofrecidos:

   - **single_buy** - inicia el proceso de pago.
   - **Zimple** – inicia el procesado de pago para zimple.

2- **Pago con token**: El usuario catastra su tarjeta y realiza el pago con un click.

   Servicios ofrecidos:

   - **Cards_new** – Inicia el proceso de catastro de una tarjeta.
   - **Users_cards** – operación que permite listar las tarjetas catastradas por un usuario.
   - **Charge** – operación que permite el pago con un token.
   - **Delete** - operación que permite eliminar una tarjeta catastrada.

Servicios que se utilizan tanto para pago ocasional como para pago con token:

- **single_buy_rollback** - operación que permite cancelar el pago (ocasional o con token).
- **get_single_buy_confirmation** - operación para consulta, si un pago (ocasional o con token) fue confirmado o no.

Servicios ofrecidos por el comercio

- **single_buy_confirm** - operación que será invocada por VPOS para confirmar un pago (ocasional o con token).

El cliente debe ofrecer en una URL pública y de común acuerdo un servicio mediante el cual se notificará la aprobación o cancelación de la transacción de un cliente final, además para funcionar como cliente Web Service de vPOS deberían soportar TLS1. 2 o mayor.

Autenticación

La clave privada y pública permitirán identificar todas las interacciones con los servicios del eCommerce de Bancard. Estas claves serán enviadas por el producto desarrollado por el comercio en todas sus peticiones para identificarse (la clave privada **nunca** viaja en forma plana, sino *hasheada* con otra información en forma de *token*). Ambas pueden ser generadas nuevamente en caso de vulneración.

La clave pública **será única**, y de la forma: [a-zA-Z0-9] {32}. La clave privada no tiene porqué ser única (aunque seguramente lo sea), y de la forma: [a-zA-Z0-9T1] {40}.

Peticiones realizadas por el comercio a VPOS

Las peticiones serán realizadas por POST a una interfaz REST

public_key	Clave pública del comercio.
operation	Datos de la operación que se va a llevar a cabo.

```
{
   "public_key": "[public key]",
   "operation": {
      "token": "[generated token]",
      ...
   }
}
```

Token

El token será generado al momento de realizar la petición, dependiendo de la operación. Será un md5 (32 caracteres). El orden debe ser exactamente como se indica.

También se tiene la opción de utilizar SHA256 en vez de md5.

single buy

md5(private_key + shop_process_id + amount + currency)

SHA256(private_key + shop_process_id + amount + currency)

single_buy confirm

md5(private_key + shop_process_id + "confirm" + amount + currency)

SHA256(private_key + shop_process_id + "confirm" + amount + currency)

single_buy get confirmation

md5(private_key + shop_process_id + "get_confirmation”)

SHA256(private_key + shop_process_id + "get_confirmation”)

single buy rollback

md5(private_key + shop_process_id + "rollback" + "0.00”)

SHA256(private_key + shop_process_id + "rollback" + "0.00”)

El token de confirm para una acción de rollback se genera usando "0.00" para amount.

Al momento de generar el token, los números deben ser transformados en cadenas, usar dos dígitos decimales y un punto (“.”) como separador de decimales. ej.:

token = md5(“[private key]" + "3332134" + "130.00" + "130.00”)

cards_new

md5(private_key + card_id + user_id + "request_new_card”)

SHA256(private_key + card_id + user_id + "request_new_card”)

users_cards

md5(private_key + user_id + "request_user_cards”)

SHA256(private_key + user_id + "request_user_cards”)

charge

md5(private_key + shop_process_id + "charge" + amount + currency +alias_token)

SHA256(private_key + shop_process_id + "charge" + amount + currency +alias_token)

delete

md5(private_key + "delete_card" + user_id + card_token)

SHA256(private_key + "delete_card" + user_id + card_token)

Pago ocasional

Tarjetas procesadas

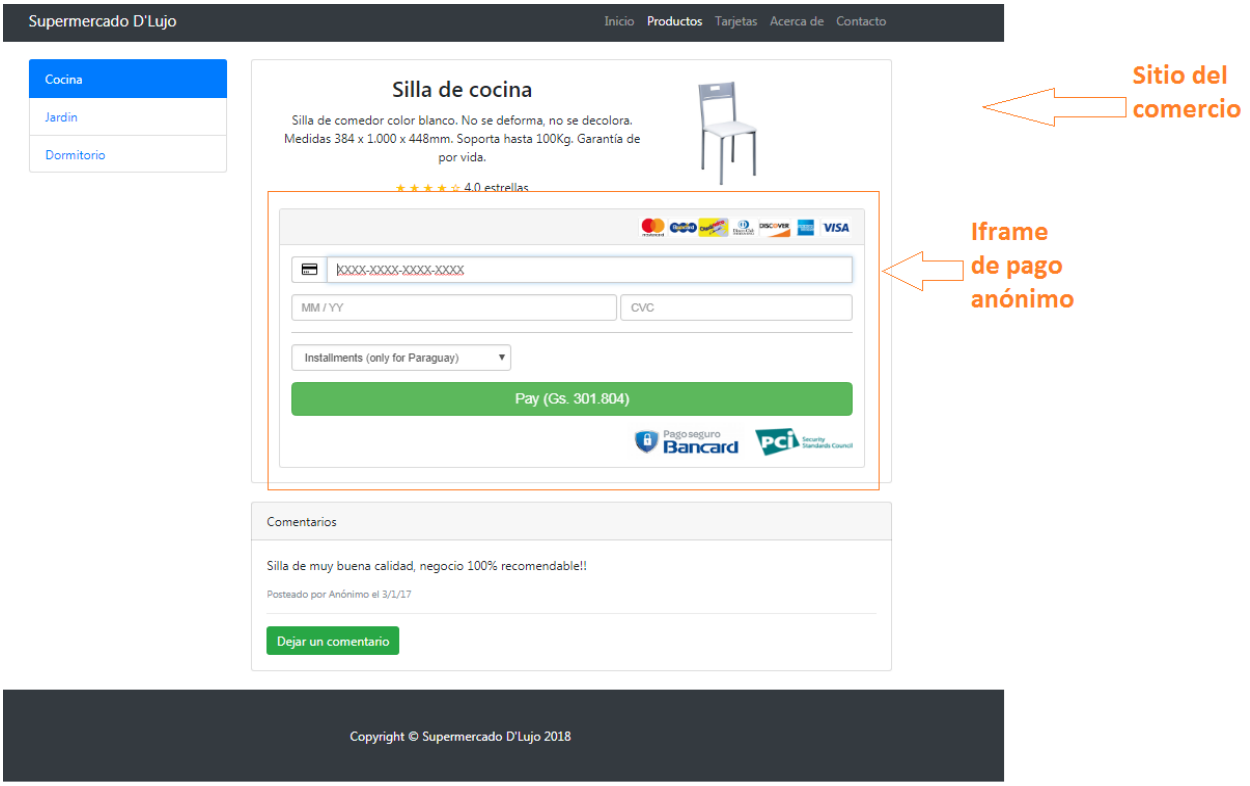
Esta operación acepta:

- Tarjetas de crédito local.
- Tarjetas de crédito internacional.
- Tarjetas de débito internacional.

Esta operación no acepta:

- Tarjeta de débito local.

El comercio carga el iframe de pago seguro de Bancard en su sitio, donde el iframe de pago ocasional queda totalmente integrado sin necesidad de que el cliente salga del sitio del comercio.



Operaciones pago ocasional

El eCommerce de Bancard cuenta con operaciones publicadas como Web Services REST disponibles para los comercios asociados que le permitirán realizar el flujo de un carrito en su sitio.

Single Buy (Pedido de pago)

POST {environment}/vpos/api/0.3/single_buy

Environment:

- **Producción** - https://vpos.infonet.com.py
- **Staging** - https://vpos.infonet.com.py:8888

Token:

md5/SHA256(private_key + shop_process_id + amount + currency)

Operación invocada por el comercio para iniciar el proceso de pago.

Este servicio devolverá un identificador de proceso (process id) que se utilizará para invocar el iframe de pago ocasional. Llamamos **iframe de pago ocasional** al iframe que permite cargar el formulario en el sitio del comercio.

Debe completarse con éxito un Single Buy para habilitación de la correspondiente opción en la Lista de test **->Recibir creación de pago**

Obs1: No se marcará en la lista de test si es que en el json del pedido envían test_client.

Obs2: Si el comercio ya cuenta con el vPOS 1.0 esta operación ya lo tiene implementada, solo deben cambiar el redirect por el iframe de pago ocasional.

El pedido estará compuesto por un JSON con los siguientes elementos:

Elementos de la petición

public_key	Clave pública.	String (50)
operation	Elemento Operation	Operation

Elementos Operation

token	md5/SHA256 de la petición	String (32)
shop_process_id	identificador de la compra.	Entero (15)
amount	Importe en guaraníes.	Decimal (15,2) - separador decimal '.'
currency	Tipo de Moneda.	String (3) - PYG (Gs)
additional_data	Campo de servicio de uso reservado para casos especiales. Opcional	String (100)

description	Descripción del pago, para mostrar al usuario.	String (20)
return_url	URL a donde se enviará al usuario al realizar el pago. Tener en cuenta que, si la tarjeta es rechazada, también se le redirigirá a esta URL.	String (255)
cancel_url	URL a donde se enviará al usuario al cancelar el pago. Opcional, se usará **return_url** por defecto.	String (255)

Descripción de “additional_data”

Este elemento será utilizado para enviar información adicional a validar en el momento de la autorización de la compra. Se empleará para indicar promociones o convenios realizados entre el comercio, Bancard y el emisor.

La estructura de este elemento será:

Dato	Tipo de Dato	Formato	Posición	Alcance	Ejemplo
Entidad	Int (3)	Rellenado con ceros a la izquierda	1-3	TC y TD	099
Marca	String (3)	Rellenado con espacios a la derecha	4-6	TC y TD	‘VS’
Producto	String (3)	Rellenado con espacios a la derecha	7-9	TC y TD	‘ORO’
Afinidad	Int (6)	Rellenado con ceros a la izquierda	10-15	TC	000045

Ejemplo de datos a enviar si el comercio desea validar que:

. la tarjeta sea de una entidad específica: 099

. la tarjeta sea de una entidad y afinidad específica: 099 000045

. la tarjeta sea de una entidad y marca específica: 099VS

. la tarjeta sea de una entidad, marca y producto específico: 099VS ORO

. la tarjeta sea de una marca específica: 000VS

.se puede enviar varias promociones: 099VS ORO000045,099VS,099VS ORO000045 *entre comas(,) sin espacio

Ejemplo petición:

```
{
   "public_key": "[public key]",
   "operation": {
      "token": "[generated token]",
      "shop_process_id": 54322,
      "currency": "PYG",
      "amount": "10330.00",
      "additional_data": "099VS ORO000045,099VS,099VS ORO000045",
      "description": "Ejemplo de pago",
      "return_url": "http://www.example.com/finish",
      "cancel_url": "http://www.example.com/cancel"
   }
}
```

La respuesta estará compuesta por un JSON con los siguientes elementos:

status	Estado de respuesta	String (20)
process_id	Identificador de la compra	String (20)

Ejemplo respuesta:

```
{
  "status": "success",
  "process_id": "i5fn*lx6niQel0QzWK1g"
}
```

Nota:

"El mensaje de respuesta se enviará en el cuerpo (body) de la petición HTTP"

Invocar al iframe de pago ocasional

Una vez obtenido el **process_id en la operación de single_buy**, el usuario podrá incluir en su e-commerce un formulario de checkout embebido, de esta forma la compra se podrá finalizar en su propia aplicación.

El JavaScript para iframe de pago ocasional se encuentra publicado:

src="**https://{enviroment}/checkout/javascript/dist/bancard-checkout-3.0.0.js**"

url formulario= **https://{enviroment}/checkout/new?process_id={process_id}**

Environment

- **Producción** - https://vpos.infonet.com.py
- **Staging** - https://vpos.infonet.com.py:8888

Para levantar el iframe:

```
window. onload = function () {
    Bancard.Checkout. createForm ('iframe-container', process_id ', styles);
  };
```

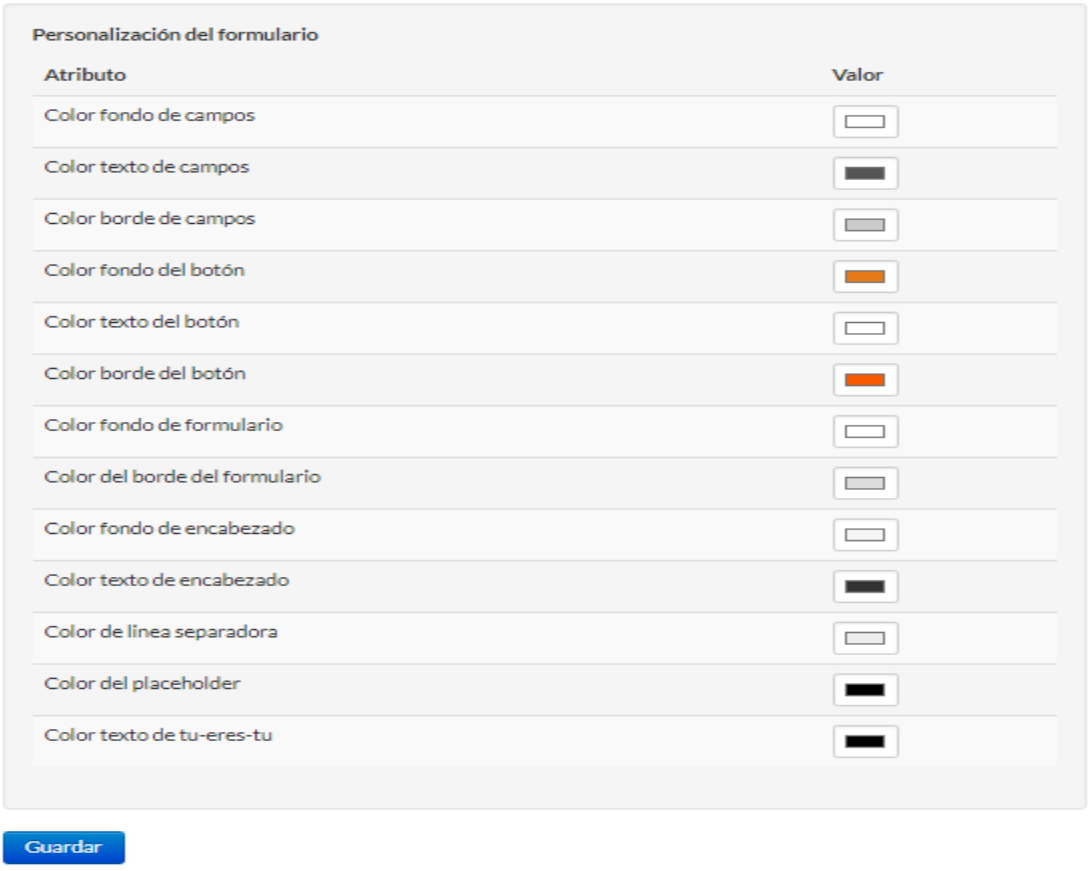
Ejemplo de código html

```
<! DOCTYPE html>

<html lang="en">

 <head>

  <meta charset="UTF-8">

  <title>iFrame</title>

  <script
src="https://vpos.infonet.com.py:8888/checkout/javascript/dist/bancard-
checkout-3.0.0.js"></script>

 </head>

 <script type="application/javascript">

  styles = {

   "form-background-color": "#001b60",

   "button-background-color": "#4faed1",

   "button-text-color": "#fcfcfc",

   "button-border-color": "#dddddd",

   "input-background-color": "#fcfcfc",

   "input-text-color": "#111111",

   "input-placeholder-color": "#111111"

  };

  window. onload = function () {

   Bancard.Checkout.createForm ('iframe-container', 'WR-YY9JmxsEZV3hpVGA7',
styles);

  };

 </script>

 <body>

  <h1 style="text-align: center">iFrame vPos</h1>
```

```
  <div style="height: 300px; width: 500px; margin: auto" id="iframe-container"></div>

 </body>


</html>
```

Panel de personalización

Pueden personalizar el iframe también por medio de una tabla de personalización que se encuentra en el panel de vpos del portal de comercio en el apartado de Perfil de la aplicación.



Luego de que el usuario ingrese los datos de su tarjeta y le da al botón de PAGAR, entonces el vpos realiza un POST a la url de confirmación que el comercio proporciono en el panel de la aplicación.

Es la siguiente operación: Operación de confirmación

Experiencia de compra de un cliente en un sitio con el **iframe de pago ocasional**

Single Buy Zimple (Pedido de pago con Zimple)

POST {environment}/vpos/api/0.3/single_buy

Environment:

- Producción - https://vpos.infonet.com.py
- Staging - https://vpos.infonet.com.py:8888

Token:

md5/SHA256(private_key + shop_process_id + amount + currency)

Operación invocada por el comercio para iniciar el proceso de pago por zimple. Es el mismo servicio que se utiliza para el pago ocasional.

Este servicio devolverá un identificador de proceso (process id) que se utilizará para invocar el iframe de zimple. Llamamos **iframe de pago zimple** al iframe que permite cargar el formulario en el sitio del comercio.

Obs: Si tiene implementado el pago ocasional, para implementar **Zimple**, solo tiene 2 variantes, el **additional_data** y el campo **zimple**.

El pedido estará compuesto por un JSON con los siguientes elementos:

Elementos de la petición

public_key	Clave pública.	String (50)
operation	Elemento Operation	Operation

Elementos Operation

token	md5/SHA256 de la petición	String (32)
shop_process_id	identificador de la compra.	Entero (15)
amount	Importe en guaraníes.	Decimal (15,2) - separador decimal '.'
currency	Tipo de Moneda.	String (3) - PYG (Gs)
additional_data	**Campo donde ira el teléfono celular del usuario con Zimple.**	**String (100)** **Ej: "0981123456"**
description	Descripción del pago, para mostrar al usuario.	String (20)
return_url	URL a donde se enviará al usuario al realizar el pago. Tener en cuenta que, si la tarjeta es rechazada, también se le redirigirá a esta URL.	String (255)
cancel_url	URL a donde se enviará al usuario al cancelar el pago. Opcional, se usará **return_url** por defecto.	String (255)

zimple	**Valor que enviar cuando se quiere invocar el iframe de simple, enviar “S”**	**String (1) Ej: “S”**

Ejemplo petición:

```
{
   "public_key": "[public key]",
   "operation": {
      "token": "[generated token]",
      "shop_process_id": 54322,
      "currency": "PYG",
      "amount": "10330.00",
      "additional_data": "0981123456",
      "description": "Ejemplo de pago",
      "return_url": "http://www.example.com/finish",
      "cancel_url": "http://www.example.com/cancel",
      "zimple": "S"
 }
}
```

La respuesta estará compuesta por un JSON con los siguientes elementos:

status	Estado de respuesta	String (20)
process_id	Identificador de la compra	String (20)

Ejemplo respuesta:

```
{
   "status": "success",
   "process_id": "i5fn*lx6niQel0QzWK1g"
}
```

Nota:

"El mensaje de respuesta se enviará en el cuerpo (body) de la petición HTTP"

Invocar al iframe de pago con zimple

Una vez obtenido el **process_id**, el usuario podrá incluir en su e-commerce un formulario de checkout embebido, de esta forma la compra se podrá finalizar en su propia aplicación.

El JavaScript para iframe de pago con zimple se encuentra publicado:

src="https://{enviroment}/checkout/javascript/dist/bancard-checkout-3.0.0.js">

url formulario= https://{enviroment}/checkout/zimple/new?process_id={process_id}

Environment

- Producción - https://vpos.infonet.com.py
- Staging - https://vpos.infonet.com.py:8888

Para levantar el iframe:

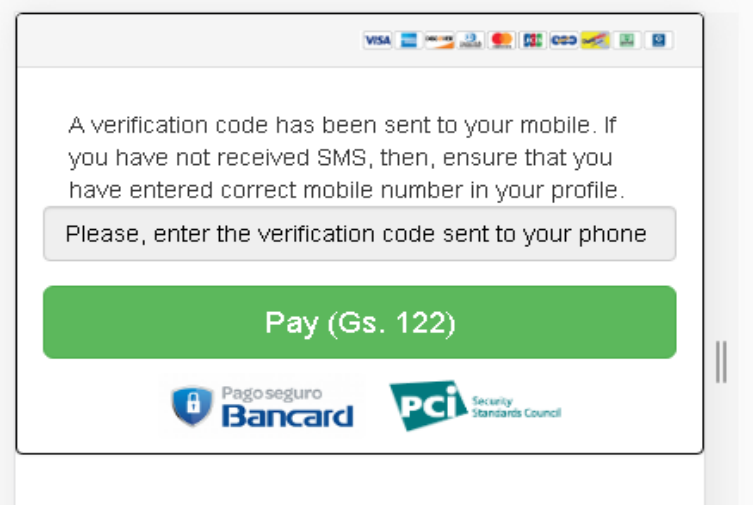
```
window. onload = function () {
    Bancard.Zimple.createForm ('iframe-container', process_id ', styles);
  };
```

Ejemplo de código html

```
<! DOCTYPE html>

<html lang="en">

 <head>

  <meta charset="UTF-8">

  <title>iFrame</title>

  <script
src="https://vpos.infonet.com.py:8888/checkout/javascript/dist/bancard-
checkout-3.0.0.js"></script>

 </head>

 <script type="application/javascript">

  styles = {

   "form-background-color": "#001b60",

   "button-background-color": "#4faed1",

   "button-text-color": "#fcfcfc",

   "button-border-color": "#dddddd",

   "input-background-color": "#fcfcfc",

   "input-text-color": "#111111",

   "input-placeholder-color": "#111111"

  };

  window. onload = function () {

   Bancard.Zimple.createForm ('iframe-container', 'WR-YY9JmxsEZV3hpVGA7',
styles);

  };

 </script>

 <body>

  <h1 style="text-align: center">iFrame vPos</h1>
```

```
<div style="height: 300px; width: 500px; margin: auto" id="iframe-container"></div>

</body>

</html>
```

Ejemplo de iframe Zimple



Flujo para pago con Zimple

- Enviar el pedido de single_buy con las variantes para Zimple.
- El servicio enviará un código al teléfono cargado en el campo additional_data.
- Levantar el iframe Zimple.
- El usuario debe cargar el código que llego a su teléfono en el iframe.
- Al confirmar el pago se debitará de su billetera Zimple.

Obs: El teléfono de prueba es 0981123456 y el código OTP para las pruebas es 1234 para una transaccion aprobada.

Luego de que el usuario ingrese los datos de su tarjeta y le da al botón de PAGAR, entonces el vpos realiza un POST a la url de confirmación que el comercio proporciono en el panel de la aplicación.

Es la siguiente operación: Operación de confirmación

Catastro y Pago con token

Esta es una opción totalmente nueva para el comercio, donde se puede catastrar una tarjeta y realizar el pago con el token generado en el catastro.

Vpos 2.0 contará con la opción de catastro de tarjetas dentro de un **iframe de catastro** siempre en el ambiente seguro de Bancard cumpliendo con las normas PCI.

Para poder catastrar una tarjeta, el servicio de vpos 2.0 se integra a un motor desarrollado en Bancard al cual llamamos “Tu eres tú”

Tarjetas procesadas

Esta operación acepta:

- Tarjetas de crédito local.
- Tarjeta de débito local.

Esta operación no acepta:

- Tarjetas de crédito internacional.
- Tarjetas de débito internacional.

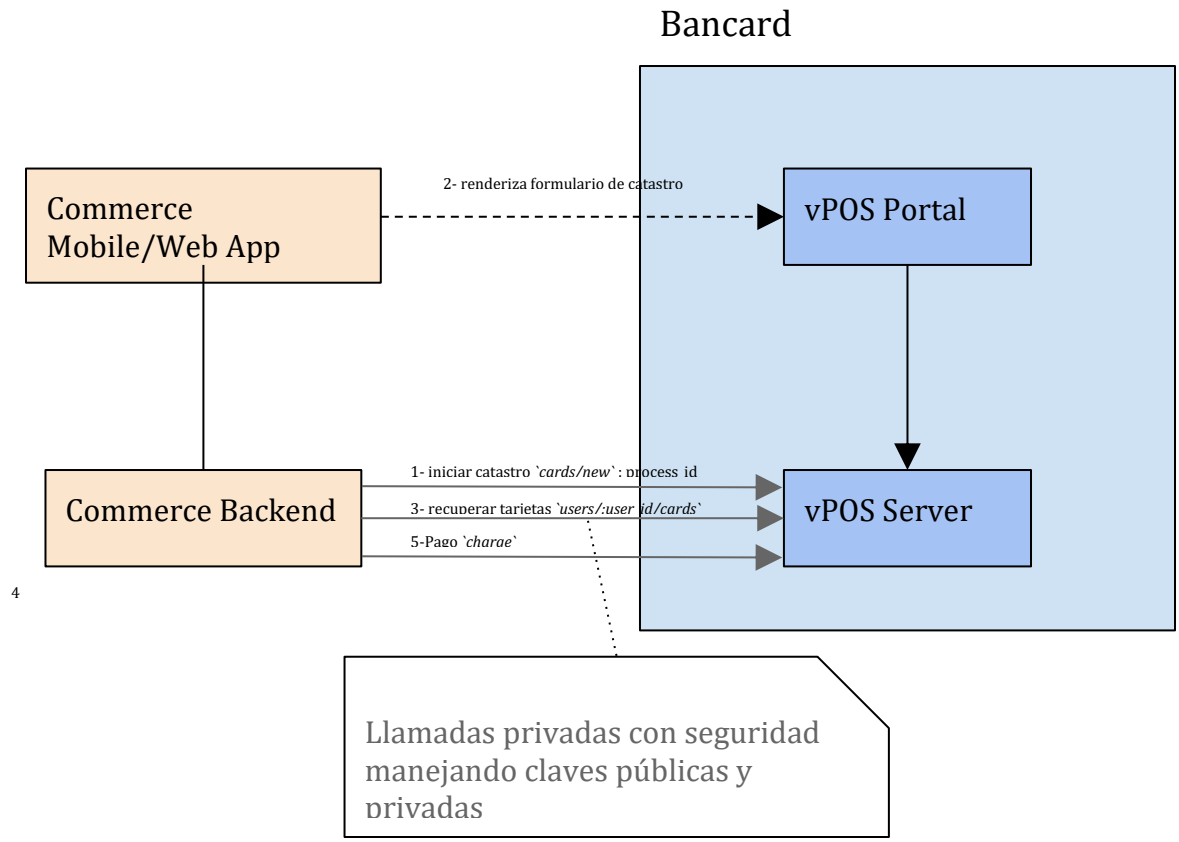
Tu eres tú

El proyecto **es un sistema de KYC** para clientes de Bancard que permitirá obtener mediante fuentes internas unas series de preguntas relacionados con el cliente, para que este a su vez puedan ser contestadas mediante selecciones múltiples o que el mismo cliente pueda responder, tipeando la respuesta en cuadros de textos. Mediante este proceso podremos medir con el motor si el cliente final es quien dice ser dando un porcentaje de aciertos. Aplicando políticas establecidas y con parámetros exclusivos, podemos aplicar el tipo de complejidad necesaria por tipo de cliente.

Una vez que el motor de Tu eres tú dé una respuesta exitosa sobre el usuario y tarjeta, entonces podrá catastrar su tarjeta.

Arquitectura planteada

Se plantea un e-commerce genérico con un backend (Commerce Backend), frontend web (Commerce Web App) y mobile (Commerce Mobile App). Los comercios pueden acceder a la API Rest de vPOS (vPOS Service) y al portal de vPOS (vPOS Portal) ambos dos instalados en Bancard cumpliendo las normas PCI.



Operaciones para catastro y pago con token

Catastro de Tarjeta (Cards_new)

POST {environment}/vpos/api/0.3/cards/new

Environment:

- **Producción** - https://vpos.infonet.com.py
- **Staging** - https://vpos.infonet.com.py:8888

Token:

md5/SHA256(private_key + card_id + user_id + "request_new_card")

Operación invocada por el comercio para iniciar el proceso de catastro.

Este servicio devolverá un identificador de proceso (process id) que se utilizará para invocar el iframe de catastro. Llamamos **iframe de catastro** al iframe que permite generar un token de tarjeta para pagos con un click.

Debe completarse con éxito un Cards_new para habilitación de la correspondiente opción en la Lista de test -> Solicitud de catastro

Obs: No se marcará en la lista de test si es que en el json del pedido envían test_client.

El pedido estará compuesto por un JSON con los siguientes elementos:

Elementos de la petición

public_key	Clave pública.	String (50)
operation	Elemento Operation	Operation

Elementos Operation

token	md5/SHA256 de la petición	String (32)
card_id	identificador de la tarjeta del usuario	Entero (19)
user_id	Identificador del usuario	Entero (19)
user_cell_phone	Teléfono del usuario	String (255)
user_mail	Mail del usuario	String (255)
return_url	URL a donde se enviará al usuario al realizar el pago. Tener en cuenta que, si la tarjeta es rechazada, también se le redirigirá a esta URL.	String (255)

Los atributos card_id, user_id, user_cell_phone y user_mail son obligatorios y son brindados para asociar el pedido de catastro de tarjeta a un usuario con una referencia interna del comercio.

Un usuario del comercio(user_id) pueden tener N tarjetas asociadas(card_id).

Un comercio no puede asociar varios usuarios con la misma tarjeta.

Ejemplo petición:


```
{
   "public_key": "kR6oAQoIYCqUZLAivLQgac3lO7mv5bXZ",
   "operation": {
      "token": "69bd9ef382cb47e796ebe9f6b6b850ba",
      "card_id": 1,
      "user_id": 966389,
      "user_cell_phone": 0919876543,
      "user_mail": "gustavo.rolfi@gmail.com",
      "return_url": "http://micomercio.com/resultado/catastro",
   }
}
```


La respuesta estará compuesta por un JSON con los siguientes elementos:

status	Estado de respuesta	String (20)
process_id	Identificador de la compra	String (20)

Ejemplo respuesta:


```
   {
      "status": "success",
      "process_id": "i5fn*lx6niQel0QzWK1g"
   }
```


Obs: Tener en cuenta que el servicio podría devolver algún dato adicional a futuro.

Recomendación para el comercio

Antes de invocar el IFrame de Bancard indicar al usuario el siguiente mensaje:

> Estimado usuario,
> A continuación usted pasará **por única vez** por un proceso de validación con preguntas de seguridad. Para iniciar favor tener en cuenta las siguientes recomendaciones:
> 1- Verifique su número de cédula de identidad
> 2- Tenga a mano sus tarjetas de crédito y débito activas
> 3- Verifique el monto y lugar de sus últimas compras en comercios o extracciones en cajeros

Invocar al iframe de catastro de tarjeta

El usuario podrá embeber dentro de su propio sitio o app un formulario para el ingreso de información sensible de tarjetas. Bancard creó una librería JavaScript para poder hacerlo de manera simple y transparente. Por más información bancard-checkout-js.

Una vez que se tiene el process_id los pasos para realizar la integración son:

1. Incluir bancard-checkout.js
2. Iniciar contenedor con código JavaScript

1. Incluir bancard-checkout.js

El JavaScript para iframe de catastro se encuentra publicado:

src="**https://{enviroment}/checkout/javascript/dist/bancard-checkout-3.0.0.js**">

url formulario= **https://{enviroment}/checkout/register_card/new?process_id={process_id}**

Environment

- **Producción** - https://vpos.infonet.com.py
- **Staging** - https://vpos.infonet.com.py:8888

2. Iniciar contenedor con código JavaScript

Para montar el formulario de catastro en el sitio web, se debe ejecutar Bancard.Cards.createForm indicando el id del contenedor, process_id y un conjunto de opciones que incluyen los estilos asociados al elemento embebido.

Ejemplo de invocación:

```
window. onload = function () { Bancard.Cards.createForm('iframe-container','[PROCESS_ID]', styles); };
```

Ejemplo código html

```
<! DOCTYPE html>

<html lang="en">

 <head>

  <meta charset="UTF-8">

  <title>iFrame</title>

  <script
src="https://vpos.infonet.com.py:8888/checkout/javascript/dist/bancard-
checkout-3.0.0.js"></script>

 </head>

 <script type="application/javascript">

  styles = {

   "form-background-color": "#001b60",

   "button-background-color": "#4faed1",

   "button-text-color": "#fcfcfc",

   "button-border-color": "#dddddd",

   "input-background-color": "#fcfcfc",

   "input-text-color": "#111111",

   "input-placeholder-color": "#111111"

  };

  window. onload = function () {

   Bancard.Cards. createForm ('iframe-container', 'V3aeqBmb7.70L76Q_*DK',
styles);

  };

 </script>
```

```
 <body>

  <h1 style="text-align: center">iFrame vPos</h1>

  <div style="height: 300px; width: 500px; margin: auto" id="iframe-container"></div>

 </body>

</html>
```

Recomendación para aplicativos que implementen catastro de tarjetas

Para comercios que implementen en su aplicativo Android tener en cuenta que para implementar el iframe se tiene que agregar las siguientes líneas de código.

```
CookieSyncManager.getInstance().startSync();

CookieManager cookieManager = CookieManager.getInstance();

cookieManager.setAcceptCookie(true);

CookieManager.getInstance().setAcceptThirdPartyCookies(registrarTarjetaWebView, true);
```

Esto es porque para el iframe necesitamos en algunas situaciones aceptar cookies y si no se tiene seteado para aceptarlas entonces el aplicativo rechaza.

Obs1: En IOS aplicativos no existe ese inconveniente.

Obs2: En Safari a partir de cierta versión se controla mediante una opción (“Prevent cross-site tracking.”) que no se pueda setear cookies desde un iFrame, en el navegador se encuentra esa opción, si desmarcan eso de su navegador ya no se da el inconveniente. Chrome también tiene esa opción, pero no viene marcado por defecto.

Flujo de catastro

Para el catastro con tarjeta de crédito:

- Se cargan los datos de la tarjeta (nro., fecha de expiración, cvv)
- Se carga cedula
- Aparece la trivia de tu eres tu

Para el catastro con tarjeta de débito:

- Se cargan los datos de la tarjeta (no, fecha de expiración, dato adicional)
- Se carga cedula

- Aparece la trivia de tu eres tu

Al momento de levantar el iframe, les aparece la opción de cargar los datos de la tarjeta y al dar siguiente les pedirá cargar un numero de cedula valido, para luego ir a la trivia de tu eres tú.

La cedula válida para las tarjetas es: 9661000 (Para las pruebas)

Tarjetas de prueba

Nombre: MasterCard	Nombre: Visa	Nombre: Bancard
Número: 5418630110000014	Número: 4907860500000016	Número: 8601010000000013
Vencimiento: 8/21	Vencimiento: 8/21	Vencimiento: 8/21
Código de seguridad: 258	Código de seguridad: 599	Código de seguridad: N/D

Les saldrán 3 preguntas, que deben responder correctamente para que se les catastre.

Este son las preguntas que pueden salirle con sus respuestas correctas:

Pregunta	Respuesta
¿Dónde recibe su extracto de algunas de sus tarjetas de crédito?	"Brasilia 765 c/ Siria"
¿Cuál es su fecha de nacimiento?	"Dec, Feb, Set"
¿Cuál fue su última transacción?	"2017/11/28 15:54:48 - Stock - 6000.00"
¿algunas de estas es tu tarjeta de crédito?	"Medalla - Credicard - 4058"
¿algunas de estas es tu tarjeta de débito?	"Medalla - Mastercad - 5789"

Mensajes de respuesta del iframe de catastro

- El mensaje del iframe si contestan todas las preguntas correctamente seria:

```
{
  "status": " add_new_card_success ",
  "description": null
}
```

- El mensaje del iframe si no contestan correctamente todas las preguntas sería:

```
{
  "status": " add_new_card_fail ",
  "description": "No se ha catastrado la tarjeta. Para continuar con el catastro favor comuníquese con el CAC de Bancard. *288/4161000"
}
```

Obs: **Tener en cuenta que el servicio podría devolver algún dato adicional a futuro.**

Recomendación para el comercio

- Al momento de habilitar la opción de catastro de tarjetas, indicar a su cliente que para catastrar debe responder unas preguntas de seguridad que sirven para velar por la seguridad de sus datos.
- Con este aviso previo al catastro, los clientes finales no se verán sorprendidos a la hora de catastrar por las preguntas de seguridad.
- Los usuarios finales estarán atentos a que les saldrán preguntas de seguridad y que esto es algo que realizara por una única vez y que pueden sentirse seguros con cargar sus datos en el sitio del comercio.

Recuperar Tarjetas catastradas de un usuario (users_cards)

POST {environment}/vpos/api/0.3/users/**user_id**/cards

Environment:

- **Producción** - https://vpos.infonet.com.py
- **Staging** - https://vpos.infonet.com.py:8888

Token:

md5/SHA256(private_key + user_id + "request_user_cards")

Operación invocada por el comercio para obtener las tarjetas catastradas de un usuario.

Debe completarse con éxito un users_cards para habilitación de la correspondiente opción en la Lista de test **-> Recibir tarjetas del usuario**

Obs: No se marcará en la lista de test si es que en el json del pedido envían test_client.

El pedido estará compuesto por un JSON con los siguientes elementos:

Elementos de la petición

public_key	Clave pública.	String (50)
operation	Elemento Operation	Operation

Elementos Operation

token	md5/SHA256 de la petición	String (32)

El **user_id** debe ser el mismo que el comercio ingresó en la operación anterior (POST cards/new)

Ejemplo petición:

```
{
   "public_key": "kR6oAQoIYCqUZLAivLQgac3lO7mv5bXZ",
   "operation": {
      "token": "69bd9ef382cb47e796ebe9f6b6b850ba"
   }
}
```

La respuesta estará compuesta por un JSON con los siguientes elementos:

status	Estado de respuesta	String (20)
cards	Elemento cards	Cards []

Array cards []

alias_token	Alias token temporal para realizar el pago	String (255)
card_masked_number	Tarjeta enmascarada	String (255)
expiration_date	Fecha espiración de la tarjeta	String (255)
card_brand	Marca de la tarjeta	String (255)
card_id	Identificador de la tarjeta	String (255)

card_type	Tipo de la tarjeta (credit o debit)	String (20)

Ejemplo respuesta:


```
{
  "status": "success"
  "cards": [{
    "alias_token": "c8996fb92427ae41e4649b934ca495991b7852b855",
    "card_masked_number": "5418********0014",
    "expiration_date": "08/21",
    "card_brand": "MasterCard",
    "card_id": 1
  }...]
}
```


El **alias_token** retornado permite realizar pagos con la tarjeta catastrada con la operación **charge**.

Es importante destacar, que el token tiene validez para una sola operación y su tiempo de vida (ttl) es del orden de los minutos.

Obs: Tener en cuenta que el servicio podría devolver algún dato adicional a futuro.

Pago con token(charge)

POST {environment}/vpos/api/0.3/charge

Environment

- **Producción** - https://vpos.infonet.com.py
- **Staging** - https://vpos.infonet.com.py:8888

Token:

md5/SHA256(private_key + shop_process_id + "charge" + amount + currency + alias_token)

El comercio podrá establecer un cargo luego de obtener las tarjetas de un usuario, para esto deberá invocar a esta operación de charge.

Debe completarse con éxito un Charge para habilitación de la correspondiente opción en la Lista de test **-> Pago con alias token**

Obs: No se marcará en la lista de test si es que en el json del pedido envían test_client.

El pedido estará compuesto por un JSON con los siguientes elementos:

Elementos de la petición

public_key	Clave pública.	String (50)
operation	Elemento Operation	Operation

Elementos Operation

token	md5/SHA256 de la petición	String (32)
shop_process_id	identificador de la compra.	Entero (15)
amount	Importe en guaraníes.	Decimal (15,2) - separador decimal '.'
currency	Tipo de Moneda.	String (3) - PYG (Gs)
number_of_payments	Cantidad de cuotas	-Debito, siempre deben enviar 1, ya que cuotas no aplica para débito. -Crédito: el comercio puede implementar un combo box donde el usuario elija la cantidad de cuotas a pagar, esto financia la entidad de la tarjeta del usuario, al comercio siempre le llega el monto total. Si envía 1

		es que se realiza en un solo pago.
additional_data	Campo de servicio de uso reservado para casos especiales. Opcional	String (100)
Alias_token	alias token obtenido de la operación de recuperar tarjetas	String (255)

Ejemplo petición:


```
{
   "public_key": "kR6oAQoIYCqUZLAivLQgac3lO7mv5bXZ",
   "operation": {
      "token": "f9aa075da613ee2b62e6712c1ed537f2",
      "shop_process_id": 60361,
      "amount": "723215.00",
      "number_of_payments": 1,
      "currency": "PYG",
      "additional_data": "",
          "description": "descripción 1",
      "alias_token": "c8996fb92427ae41e4649b934ca495991b7852b855"
   }
}
```


El alias_token es el obtenido al recuperar la lista de tarjetas de un usuario bajo el atributo con el mismo nombre.

Ejemplo respuesta: La respuesta ya viene en el response del request. El tiempo de respuesta es en segundos.


```
{ “operation": {
    "token": "[generated token]",
    "shop_process_id": "12313",
    "response": "S",
    "response_details": "respuesta S",
    "extended_response_description": "respuesta extendida",
    "currency": "PYG",
    "amount": "10100.00",
    "authorization_number": "123456",
    "ticket_number": "123456789123456",
    "response_code": "00",
    "response_description": "Transacción aprobada.",
    "security_information": {
      "customer_ip": "123.123.123.123",
      "card_source": "I",
      "card_country": "Croacia",
      "version": "0.3",
      "risk_index": "0"
    }}}
```


Eliminar tarjeta

DELETE {environment}/vpos/api/0.3/users/**user_id**/cards

Environment

- **Producción** - https://vpos.infonet.com.py

● **Staging** - https://vpos.infonet.com.py:8888

Token:

md5/SHA256(private_key + "delete_card" + user_id + alias_token)

Se podrá eliminar una tarjeta a un usuario, para esto se deberá invocar a la siguiente operación.

Debe completarse con éxito un delete para habilitación de la correspondiente opción en la Lista de test -> Eliminar tarjeta del usuario

Obs: No se marcará en la lista de test si es que en el json del pedido envían test_client.

El pedido estará compuesto por un JSON con los siguientes elementos:

Elementos de la petición

public_key	Clave pública.	String (50)
operation	Elemento Operation	Operation

Elementos Operation

token	md5/SHA256 de la petición	String (32)
alias_token	alias token obtenido de la operación de recuperar tarjetas	String (255)

Ejemplo petición:

```
{
  "public_key": "kR6oAQoIYCqUZLAivLQgac3lO7mv5bXZ",
  "operation": {
    "token": "f9aa075da613ee2b62e6712c1ed537f2",
```

```
    "alias_token": "c8996fb92427ae41e4649b934ca495991b7852b855"
  }
}
```

El alias_token es el obtenido al recuperar la lista de tarjetas de un usuario bajo el atributo con el mismo nombre.

El **user_id** debe ser el mismo que el comercio ingresó en la operación de recuperar las tarjetas.

La respuesta estará compuesta por un JSON con los siguientes elementos:

status	Estado de respuesta	String (20)

Ejemplo respuesta:

```
{
  "status": "success"
}
```

Obs: Tener en cuenta que el servicio podría devolver algún dato adicional a futuro.

Operaciones comunes para pago ocasional y pago con token

Buy Single Confirm (Operación de confirmación de una transaccion)

POST [URL] (Definida por el comercio)

Esta acción es invocada por VPOS al finalizar una transacción. Tiene como objetivo confirmar o cancelar un pago. Este será el único medio por el cual el cliente tendrá la certeza de que el usuario completó satisfactoriamente una transacción.

Bancard realizará una petición POST a la url de confirmación que el comercio cargo en su panel de aplicación de vpos en el portal de comercios, enviando el JSON en el cuerpo del pedido o body.

El comercio deberá responder con status 200 a la operación, como se muestra más abajo en el ejemplo de respuesta. Si el comercio no responde con status 200 dentro de los siguientes 60 segundos, vPOS cerrará la conexión y se marcará como inválida la confirmación en la traza y con una indicación del timeout en reemplazo de lo que debió ser la respuesta del comercio. **Si el comercio no responde con 200 eso no**

significa que la transaccion haya quedado denegada, siempre deben realizar la consulta para verificar el estado en que quedo la transaccion.

Esta operación realiza el vpos para pagos ocasionales y para pagos con token.

Nota:

Si el producto Web o Mobile desarrollado por el Comercio inicia la operación de compra (single buy) y no recibe la confirmación (single buy confirm) por parte del VPOS, puede invocar a la operación de consulta (single buy get confirmation) para saber en qué estado quedo la transacción y actualizar en su sistema o también puede invocar a la operación de reversa (single buy rollback) para evitar inconsistencias en su sistema. El tiempo de espera recomendado es de 10 minutos.

Debe completarse con éxito un Buy Single Confirm para habilitación de la correspondiente opción en la Lista de test **-> Confirmamos correctamente al comercio.**

Obs1: No se marcará en la lista de test si es que en el json del pedido envían test_client.

Obs2: Si el comercio ya cuenta con el vPOS 1.0 ya está preparado para recibir la respuesta de los pagos por la url de confirmación cargada en el perfil de la aplicación del comercio.

El pedido estará compuesto por un JSON con los siguientes elementos:

Elementos de la petición

operation	elemento Operation	Elemento

Elementos Operation

token	md5/SHA256 de la petición	String (32)
shop_process_id	identificador interno del comercio	Entero (15)
response	Indicador de detalle procesado	String (1) - S o N
response_details	Descripción del proceso	String (60)

amount	Importe en guaraníes.	Decimal (15,2) - separador decimal '.'
currency	Tipo de Moneda.	String (3) - PYG (Gs)
authorization_number	Código de autorización.	String (6) - Solo si la transacción es aprobada.
ticket_number	Identificador de autorización.	Int (15)
response_code	Código de respuesta de la transacción.	String (2) - 00 (transacción aprobada) - 05 (Tarjeta inhabilitada) - 12 (Transacción inválida) - 15 (Tarjeta inválida) - 51 (Fondos insuficientes)
response_description	Descripción de la respuesta de transacción.	String (40)
extended_response_description	Descripción extendida de la respuesta de transacción.	String (100)
security_information	Elemento SecurityInformation	Elemento

Elementos SecurityInformation

card_source	Local o Internacional	String (1) - L (Local) - I (Internacional)
customer_ip	Ip del cliente que ingresa los datos de pago	String (15)

card_country	País de origen de la tarjeta	String (30)
version	Version de la API	String (5)
risk_index	Indicador de riesgo.	Int (1)

Ejemplo petición:


```
{
   "operation": {
      "token": "[generated token]",
      "shop_process_id": "12313",
      "response": "S",
      "response_details": "respuesta S",
      "extended_response_description": "respuesta extendida",
      "currency": "PYG",
      "amount": "10100.00",
      "authorization_number": "123456",
      "ticket_number": "123456789123456",
      "response_code": "00",
      "response_description": "Transacción aprobada.",
      "security_information": {
         "customer_ip": "123.123.123.123",
         "card_source": "I",
         "card_country": "Croacia",
         "version": "0.3",
```

```
        "risk_index": "0"
      }
    }
}
```

Notas:

Información índice de riesgos

- El atributo de “risk_index”
  Consiste en un índice de riesgo de la transacción en tiempo real, este campo devolverá un número que indicará al comercio el riesgo de la transacción en tiempo real de acuerdo con la siguiente tabla:

Escala	Riesgo
0	No se puede generar el riesgo en tiempo real
1	Bajo
2	Bajo
3	Bajo
4	Medio
5	Medio
6	Medio
7	Alto

8	Alto
9	Alto

El **índice de riesgo** será generado para las transacciones que se realicen con **tarjeta de crédito local.**
Para las transacciones con tarjetas internacionales el campo risk_index mostrará 0.
Para las transacciones con tarjetas de débito el campo risk_index mostrará 0.
Para las transacciones con tarjetas de crédito de otra procesadora (cabal, panal) mostrará 0.
El campo risk_index mostrará 0 cuando no se puede generar el índice de riesgo en tiempo real.

Acciones del comercio:

- Para una transacción con Riesgo Bajo, el comercio puede estar tranquilo con la transacción.
- Para una transacción con Riesgo Medio, el comercio puede pedir datos de seguridad al cliente para verificar si la transacción le corresponde.
- Para una transacción con Riesgo Alto, el comercio debe verificar la transacción con el cliente y llamar a Bancard en caso de no tener respuesta del cliente, puede escribir un correo a riesgos@bancard.com.py y asegurar la transacción. En caso de que el comercio tenga que entregar una mercadería, favor primero verificar con Bancard si es una transacción legitima.

Ejemplo respuesta esperada por el comercio:

```
{
  "status": "success"

}
```

Single Buy Rollback (Operación de reversa de transacción)

POST {environment}/vpos/api/0.3/single_buy/rollback

Environment

- **Producción** - https://vpos.infonet.com.py
- **Staging** - https://vpos.infonet.com.py:8888

Token:

md5/SHA256(private_key + shop_process_id + "rollback" + "0.00")

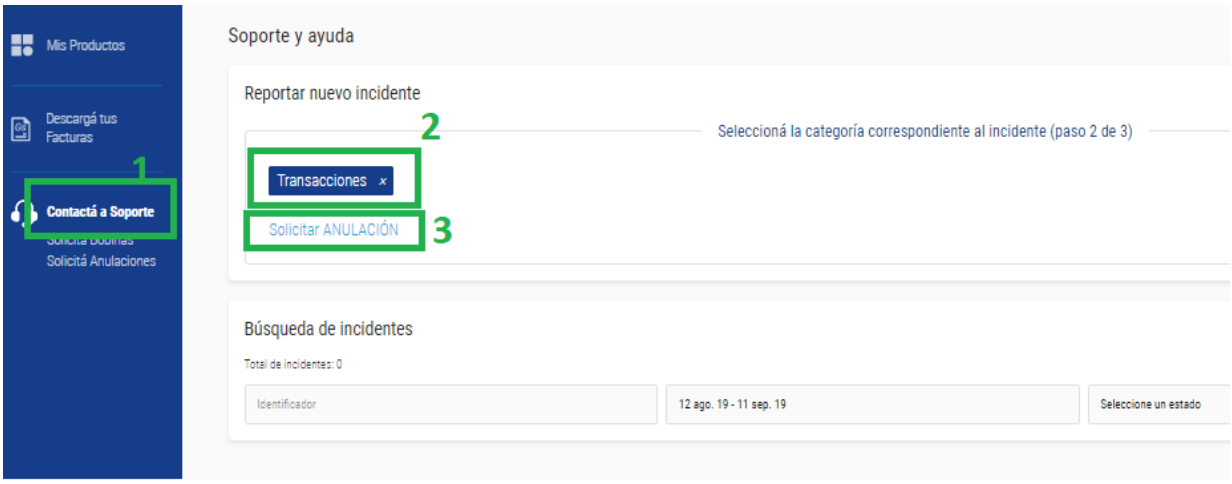
Esta operación se puede utilizar para pago ocasional y para pagos con token.

La operación de Rollback deberá ser enviada en los siguientes casos:

- Para realizar un reverso de pago.
- Para transacciones canceladas por el usuario en la página de vPos.
- Para transacciones abandonadas o no culminadas por el usuario.

La operación de rollback solo puede enviar en el día en el que se realizo la operación, a esto lo denominamos reversas automáticas, las que se aplican antes que impacte en el extracto del cliente final.

Si quieren reversar una operación que ya impacto en el extracto, deben ingresar su pedido de anulación por el canal oficial, portal de comercios/soporte/anulaciones:



La operación del Rollback será satisfactoria mientras la transacción no haya sido CUPONADA (confirmada en el extracto del cliente). Si el JSON devuelve status: error y key: "TransactionAlreadyConfirmed", el comercio deberá realizar el proceso manual de pedido de reversión de una transacción cuponada a tramitar en el Área Comercial de Bancard.

El rollback devolverá un estado general "status". "success" indica que el pedido será notificado para cancelar. "error" indica que por alguna razón el pedido no puede continuar. Las posibles causas de error son:

- **InvalidJsonError -** Error en el JSON enviado
- **UnauthorizedOperationError -** Las credenciales enviadas no tienen permiso para la operación rollback.
- **ApplicationNotFoundError -** No existen permisos para las credenciales enviadas.
- **InvalidPublicKeyError** - Existe un error sobre la clave pública enviada.
- **InvalidTokenError** - El token se generó en forma incorrecta.

- **InvalidOperationError** - El JSON enviado no es válido. No cumple con los tipos o limites definidos.
- **BuyNotFoundError** - No existe el proceso de compra seleccionado
- **PaymentNotFoundError** - No existe un pedido de pago para el proceso seleccionado. Esto quiere decir que el cliente no pagó este pedido y deberá tomarse como una respuesta correcta para dichas situaciones.
- **AlreadyRollbackedError** - Ya existe un pedido de rollback previo.
- **PosCommunicationError** - Existen problemas de comunicación con el componente de petición de rollback.
- **TransactionAlreadyConfirmed** - Transacción Cuponada (Confirmada en el extracto del cliente)

En el caso de que una compra sea iniciada por el producto desarrollado por el comercio, pero no se finalice por el usuario o no se obtenga respuesta de parte de vpos luego de 10 minutos, se debería invocar un **Get Buy Single Confirmation** para conocer el estado del pedido. Si el pago todavía no ha sido realizado, el comercio puede optar por realizar un rollback del pedido invocando a la operación **Single Buy Rollback**.

Nota:

Debe completarse con éxito un Single Buy Rollback manual en un caso de transacción aprobada para habilitación de la correspondiente opción en la Lista de test ->Recibir rollback

Obs1: No se marcará en la lista de test si es que en el json del pedido envían test_client.

Obs2: Si el comercio ya cuenta con el vPOS 1.0 esta operación ya lo tiene implementada.

El pedido estará compuesto por un JSON con los siguientes elementos:

Elementos de la petición

public_key	Clave pública	String
operation	Elemento Operation	Elemento

Elemento Operation

token	md5/SHA256 de la petición	String

shop_process_id	Identificador interno del comercio	String

Ejemplo petición:

```
{
   "public_key": "[public key]",
   "operation": {
      "token": "[generated token]",
      "shop_process_id": "12313"
   }
}
```

La respuesta estará compuesta por un JSON con los siguientes elementos:

Elementos de la respuesta

status	Estado de respuesta	String Valores posibles: - success - error
messages	Array de elemento Message	Array

Elemento Message

key	Clave de respuesta	String Valores posibles:

		- InvalidJsonError - UnauthorizedOperationError - ApplicationNotFoundError - InvalidPublicKeyError - InvalidTokenError - InvalidOperationError - BuyNotFoundError - PaymentNotFoundError - AlreadyRollbackedError - PosCommunicationError - RollbackSuccessful - TransactionAlreadyConfirmed
level	Nivel de despliegue del mensaje	String Valores posibles: - info - error
dsc	Descripción de respuesta	String

Ejemplo respuesta:

```
{
  "status": "success",
  "messages": [
 {
        "key": "RollbackSuccessful",
        "level": "info"
```

```
        "dsc": "Rollback correcto.",
    }
    ]
}
```

Get Buy Single Confirmation(Operación de consulta de una transacción)

POST {environment}/vpos/api/0.3/single_buy/confirmations

Environment

- **Producción** - https://vpos.infonet.com.py
- **Staging** - https://vpos.infonet.com.py:8888

Token:

md5/SHA256(private_key + shop_process_id + "get_confirmation")

Esta acción es invocada por el comercio para consultar si existió o no una confirmación.

Debe completarse con éxito un Get Buy Single Confirm para habilitación de la correspondiente opción en la Lista de test **-> Recibimos pedido de confirmación del comercio**

Obs1: No se marcará en la lista de test si es que en el json del pedido envían test_client.

Obs2: Si el comercio ya cuenta con el vPOS 1.0 esta operación ya lo tiene implementada.

El pedido estará compuesto por un JSON con los siguientes elementos:

Elementos de la petición

public_key	Clave pública.	String (50)
operation	Elemento Operation	Operation

Elementos Operation

token	md5/SHA256 de la petición	String (32)
shop_process_id	identificador de la compra.	Entero (15)

La respuesta estará compuesta por un JSON con los siguientes elementos:

Elementos de la respuesta

status	Estado de respuesta	String Valores posibles: - success - error
confirmation	Información de confirmación	Elemento SingleBuyConfirmation
messages	Array de elemento Message	Array

Elemento Message

key	Clave de respuesta	String Valores posibles: - InvalidJsonError - UnauthorizedOperationError - ApplicationNotFoundError - BuyNotFoundError - InvalidPublicKeyError - InvalidTokenError

		- InvalidOperationError - PaymentNotFoundError - AlreadyRollbackedError
level	Nivel de despliegue del mensaje	String Valores posibles: - info - error
dsc	Descripción de respuesta	String

Elemento SingleBuyConfirmation

token	MD5/SHA256 de la petición	String (32)
shop_process_id	Identificador interno del comercio	Entero (15)
response	Indicador de detalle procesado	String (1) - S o N
response_details	Descripción del proceso	String (60)
amount	Importe en guaraníes.	Decimal (15,2) - separador decimal '.'
currency	Tipo de Moneda.	String (3) - PYG (Gs)
authorization_number	Código de autorización.	String (6) - Solo si la transacción es aprobada.
ticket_number	Identificador de autorización.	Int (15)

response_code	Código de respuesta de la transacción.	String (2) - 00 (transacción aprobada) - 05 (Tarjeta inhabilitada) - 12 (Transacción inválida) - 15 (Tarjeta inválida) - 51 (Fondos insuficientes)
response_description	Descripción de la respuesta de transacción.	String (40)
extended_response_description	Descripción extendida de la respuesta de transacción.	String (100)
security_information	Elemento SecurityInformation	Elemento

Elementos SecurityInformation

card_source	Local o Internacional	String (1) - L (Local) - I (Internacional)
customer_ip	Ip del cliente que ingresa los datos de pago	String (15)
card_country	País de origen de la tarjeta	String (30)
version	Version de la API	String (5)
risk_index	Indicador de riesgo.	Int (1)

Ejemplo petición:

```
{
```

```
    "public_key": "[public key]",
    "operation": {
      "token": "[generated token]",
      "shop_process_id": "12313"
    }
  }
```


Ejemplo respuesta:


```
{
  "status": "success"
  "confirmation": {
    "token": "[generated token]",
    "shop_process_id": "12313",
    "response": "S",
    "response_details": "respuesta S",
    "extended_response_description": "respuesta extendida",
    "currency": "PYG",
    "amount": "10100.00",
    "authorization_number": "123456",
    "ticket_number": "123456789123456",
    "response_code": "00",
    "response_description": "Transacción aprobada.",
    "security_information": {
      "customer_ip": "123.123.123.123",
      "card_source": "I",
      "card_country": "Croacia",
```

```
        "version": "0.3",
        "risk_index": "0"
    }
  }
}
```

Restricciones del comercio

A continuación, se presentan restricciones que debe contemplar el comercio que desarrolle la integración con el eCommerce de Bancard.

Interfaz de respuesta

Luego de que el usuario ingresa sus datos de tarjeta y se confirma al comercio por medio de la operación “Buy Single Confirm” el comercio debe desplegar una interfaz de respuesta con la aprobación de la transacción.

En esta interfaz se deben respetar las siguientes restricciones:

- Se deben indicar los datos de la transacción:
  - Fecha y Hora
  - Número de pedido (shop_process_id)
  - Importe (amount)
  - Descripción de la Respuesta (response_description)
- **No** debe mostrarse al usuario:
  - Código de autorización (authorization_number)
  - Código de respuesta (response_code)
  - Respuesta extendida (extended_response_description)
  - Información de seguridad (security_information)

Notas generales

- El Comercio debe incluir en su aplicación la sección de **Contacto**, de manera que el cliente pueda

evacuar consultas referentes a las compras del ecommerce.

- La aplicación podrá registrar todos datos del cliente que requiera el comercio, salvo todos aquellos que se relacionen a sus tarjetas de crédito (Número de tarjeta, código de seguridad, vencimiento, etc.)
- El logo para utilizar por el comercio debe idealmente tener un ancho de 173 píxeles y un alto 55 píxeles. El ancho mínimo es de 85 píxeles.
- Para la comunicación con nuestro web en ambiente de desarrollo deben tener habilitado el puerto 8888.

Formato de mensajería – JSON

Para enviar y recibir información se empleará el formato JSON (JavaScript Object Notation).

Información sobre JSON - http://json.org/

Validación de JSON - http://jsonlint.com/

Al consumir un JSON enviado por el VPOS debe prestarse especial atención a los caracteres especiales (ej. tildes). El mismo será enviado utilizando el standard "\uXXXX" (Donde X es un digito hexadecimales)

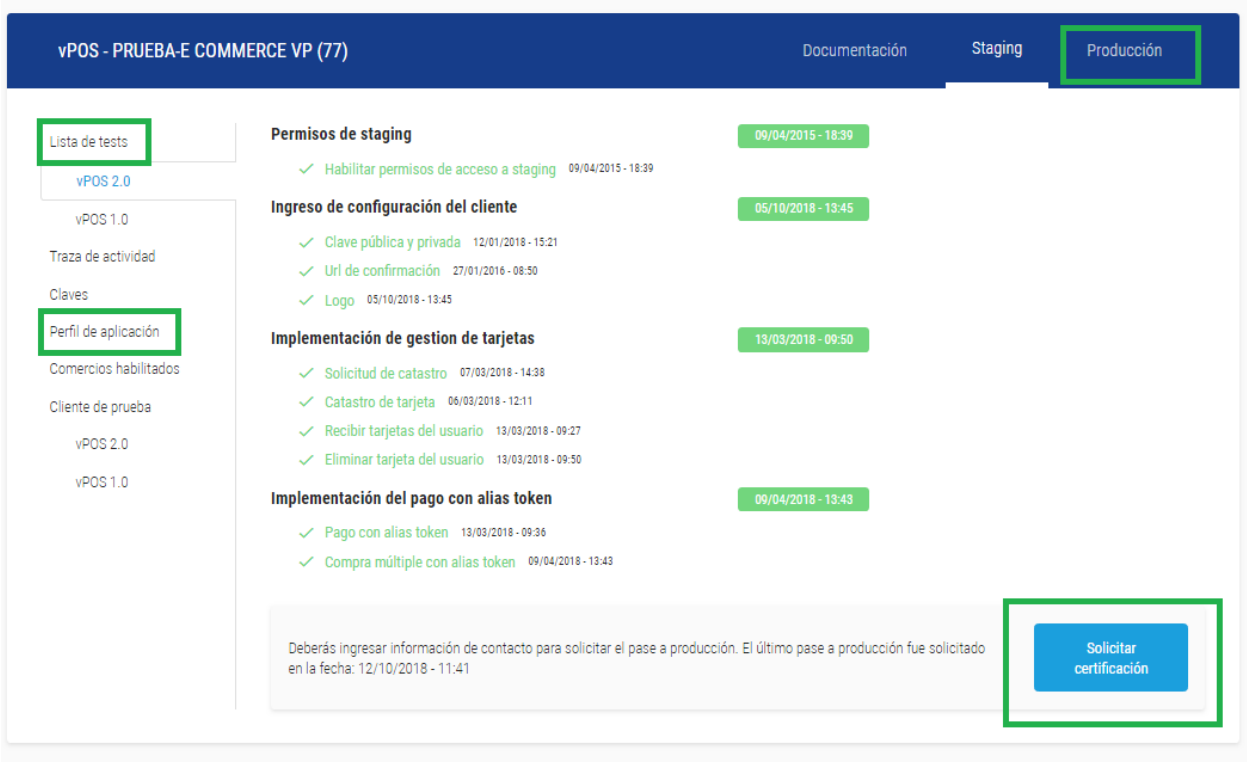
Los JSON enviados y recibidos por Bancard y el comercio deberán realizarse mediante una petición POST enviando el JSON en el cuerpo del pedido o body.

Solicitud de pase a producción

El comercio deberá completar la lista de test para solicitar la certificación y próximo paso a producción, los pasos a producción son los siguientes:

- El comercio debe completar su lista de test, todos los campos deben tener chequeado, mientras hacen sus pruebas cada ítem de la lista de test se marca en verde.
- Al tener la lista de test completamente chequeado, se habilita el botón de **"Solicitar certificación"**
- En el botón el comercio carga la url a certificar y un usuario/contraseña si se necesita para realizar las pruebas en su sitio
- El pedido de certificación llega al equipo de soporte, donde hacen compras de prueba en la url dada, si vemos que la integración se encuentra ok entonces se le da el acceso a producción.
- Se habilita una pestaña de producción donde el comercio tiene las claves de producción y puede configurar su perfil de aplicación en producción

- También el comercio debe cambiar las urls de las apis por las de producción.



Código de errores – Vpos 2.0

Código de errores Vpos 2.0	
Card errors (Código Errores para el catastro)	
CardAlreadyRegisteredByUserError	'The user has already registered the card.'
InvalidCiError	"The user's ci does not match with card's ci"
CardRequestAlreadyProcessedError	"The card request with process id #{@process_id} has already been processed."
CardInvalidDataError	'The data for the card is not correct.'
NewCardRequestNotFoundError	"New card request not found for process id: #{@process_id}"
CardNotFoundError	'The card does not exist'
CardAliasTokenExpiredError	'The card alias token has expired.'
CardBlockedError	'The card for the user is blocked.'
InvalidCardStatus	'The given status is incorrect

Buy errors (Código de errores para pedido de pago)

BuyNotFoundError	'Buy Not Found'
InvalidAmountError	"Amount attribute must be greater than zero."

Application errors (Código de errores para APIS vpos)	
ApplicationNotFoundError	'Application not found'
InvalidTokenError	'Invalid token'
InvalidPublicKeyError	'Invalid Public key'
PublicKeyNotFoundError	'Public key not found'
ApplicationCommunicationError	
ApplicationCredentialNotFoundError	"The credential for the application was not found."
CantCreateApplicationCredentialError	"The credential for the application could not be created."

Código de errores en los pagos	
0	APROBADA
2	CONSULTE SU EMISOR - CONDICION ESPECIAL
3	NEGOCIO INVALIDO
4	RETENGA TARJETA
5	NO APROBADO
6	ERROR DE SISTEMA
7	RECHAZO POR CONTROL DE SEGURIDAD
8	TRANSACCION FALLBACK RECHAZADA
12	TRANSACCION INVALIDA
13	MONTO INVALIDO
14	TARJETA INEXISTENTE
15	EMISOR INEXISTENTE, NO HABILITADO P/NEGOC
17	CANCELADO POR EL CLIENTE
19	INTENTE OTRA VEZ
22	SOSPECHA DE MAL FUNCIONAMIENTO
33	TARJETA VENCIDA
34	POSIBLE FRAUDE - RETENGA TARJETA
35	LLAME PROCESADOR/ADQUIRENTE
36	TARJETA/CUENTA BLOQUEADA POR LA ENTIDAD
37	MES NACIMIENTO INCORRECTO-TARJ.BLOQUEADA
38	3 CLAVES EQUIVOCADAS - TARJETA BLOQUEADA
39	NO EXISTE CUENTA DE TARJETA DE CREDITO
40	TIPO DE TRANSACCION NO SOPORTADA
41	TARJETA PERDIDA - RETENGA TARJETA
42	NO APROBADO - NO EXISTE CUENTA UNIVERSAL
43	TARJETA ROBADA - RETENGA TARJETA
45	NO EXISTE LA CUENTA

46	EMISOR/BANCO NO RESPONDIO EN 49 SEGUNDOS
49	OPERACION NO ACEPTADA EN CUOTAS
51	NO APROBADA-INSUF.DE FONDOS
54	TARJETA VENCIDA
55	CLAVE INVALIDA
57	NO APROBADA - TARJETA INADECUADA
58	NO HABILITADA PARA ESTA TERMINAL
59	NO APROBADO - POSIBLE FRAUDE
60	NO APROBADO - CONSULTE PROCESADOR ADQUIR
61	EXCEDE MONTO LIMITE
62	NO APROBADA - TARJETA RESTRINGIDA
63	VIOLACION DE SEGURIDAD
65	EXCEDE CANTIDAD DE OPERACIONES
66	LLAMAR SEGURIDAD DEL PROCESADOR ADQUIR.
70	NEGOCIO INHABILITADO POR FALTA DE PAGO
71	OPERACIÓN YA EXTORNADA
72	FECHA INVALIDA
73	CODIGO DE SEGURIDAD INVALIDO
92	EMISOR DESCONECTADO - PROBLEMAS EN LINEA
94	TRANSACCION DUPLICADA - NO APROBADO

Soporte para la integración

Ingrese al sitio: https://comercios.bancard.com.py, utilice la opción de menú Soporte, y el servicio vPOS.

Si se trata de una consulta de integración utilice la opción Pruebas de integración (desarrollo), estas consultas serán atendidas de lunes a viernes en horario de oficina.