Manual de Integración con API Pública mediante SDK

Versión 1.16

Agosto de 2020

Macro
Cerca, siempre.

Contenido

1. Método de integración.

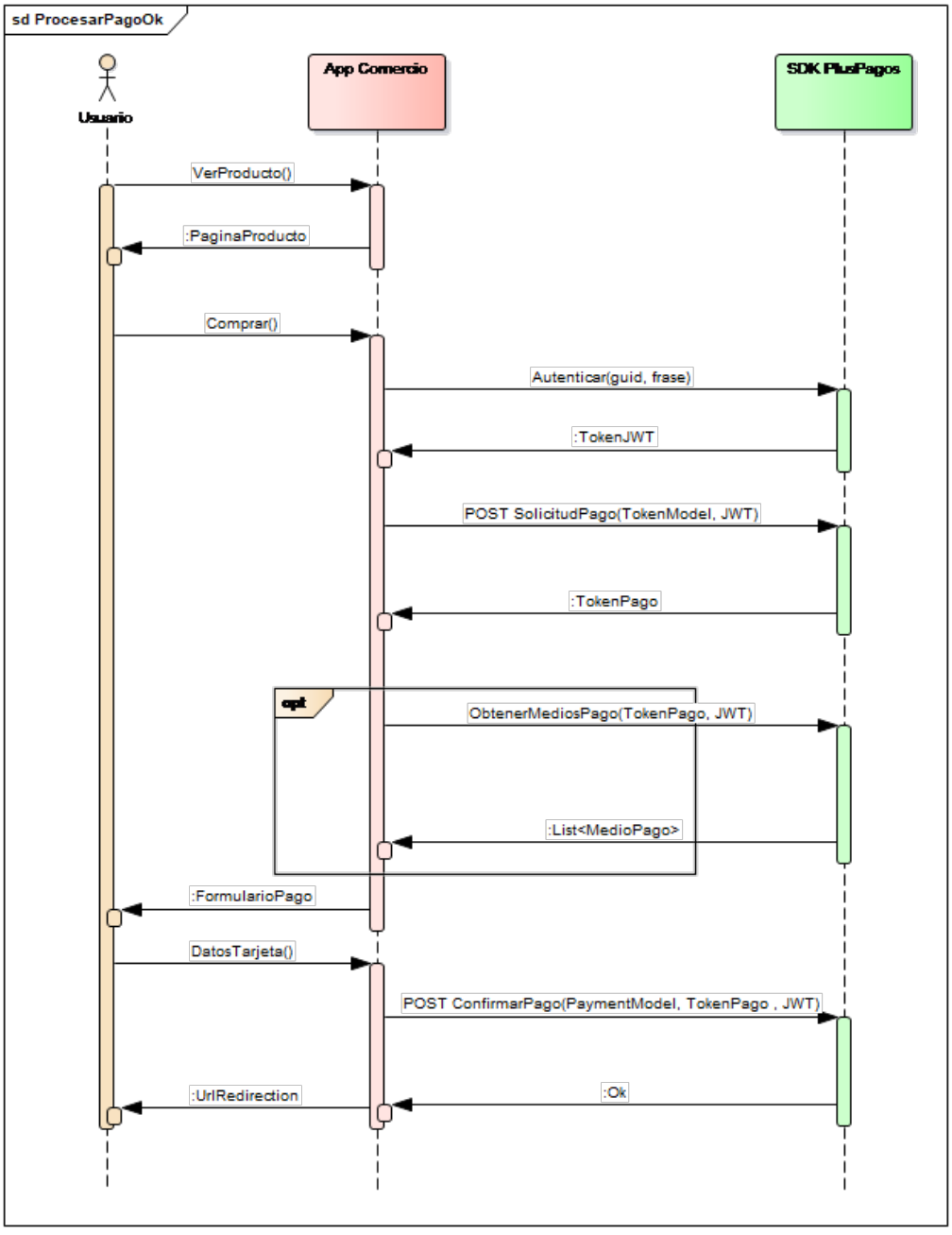
El consumo de los servicios ofrecidos por la **API Pública de Macro Click de Pago** se realiza a través de un SDK. A continuación, se detalla el flujo de integración.

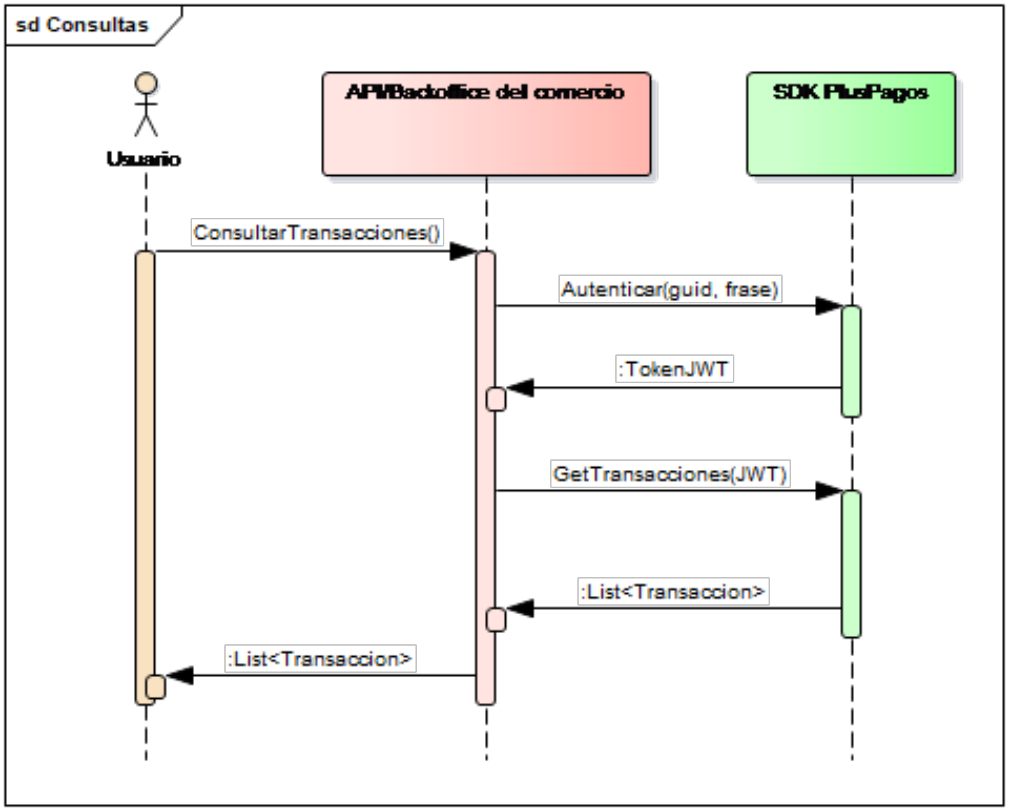
1.1 Introducción

A través del SDK, Macro Click de Pago permite la utilización de dos sets de servicios.

- Servicios de pago: provee los métodos necesarios obtener el token de pago, los medios de pago disponibles para su comercio y por último ejecutar el pago correspondiente.
- Servicios de consulta: permite obtener listados de transacciones e información de una transacción particular a través del identificador de la transacción en el comercio.

Diagramas de secuencia





1.2 Instalación

• NET Framework

El presente SDK es compatible con las versiones 3.5 a 4.6.1 de NET Framework.

Instalar el siguiente paquete Nuget:

```
PM> Install-Package PlusPagosConnector.NetFramework
```

Versión a la fecha: 4.3.0

Agregar los siguientes using:

```
using PlusPagosConnector.NetFramework;
using PlusPagosConnector.NetFramework.Models;
```

Especificar la utilización del protocolo de seguridad TLS 1.2:

```
System.Net.ServicePointManager.SecurityProtocol = System.Net.SecurityProtocolType.Tls12;
```

• NET Core

Instalar el siguiente paquete Nuget:

```
PM> Install-Package PlusPagosConnector
```

Versión a la fecha: 4.2.0

Agregar los siguientes using:

```
using PPlusPagosConnector;
using PlusPagosConnector.Models;
```

Especificar la utilización del protocolo de seguridad TLS 1.2:

```
System.Net.ServicePointManager.SecurityProtocol = System.Net.SecurityProtocolType.Tls12;
```

De aquí en adelante, los pasos son los mismos tanto para integrar con versiones de NET Framework como con versiones de NET Core.

• Ambientes

El SDK permite la utilización de dos ambientes: sandbox y producción que deberán indicarse de la siguiente manera:

```
PPConnector ppConnector = new PPConnector(AmbienteHelper.Ambiente.SANDBOX);
```

```
PPConnector ppConnector = new PPConnector(AmbienteHelper.Ambiente.PRODUCTION);
```

• HealthCheck

Este método permite conocer el estado actual de la API de Macro Click de Pago.

```
var status = ppConnector.HealthCheck();
```

Response

Código	Response	
200	code	200
	status	true
	message	Servicio disponible.
	data	Versión 1.0
400	code	400
	status	false
	message	Servicio no disponible.
	data	-

• Responses

Todas las respuestas de la API Pública de Macro Click de Pago coinciden con siguiente esquema:

Response		
Campo	**Descripción**	**Tipo**
status	Estado en el que finalizó el request	bool
code	Status Code del request	int
message	Mensaje informativo correspondiente a la operación realizada.	string
data	Contenido útil para el comercio. Se proveen los modelos necesarios para realizar la conversión del contenido.	string

1.2.1 Flujo de un pago

1. Inicializar el conector:

```
PPConnector ppConnector = new PPConnector(AmbienteHelper.Ambiente.SANDBOX);
```

El mismo recibe como parámetro el ambiente en el que se está desarrollando la integración (Sandbox o Producción). El conector permitirá el acceso a los métodos expuestos por la API Pública de Macro Click de Pago.

2. Obtener token de acceso

Para poder interactuar con los métodos expuestos por Macro Click de Pago, se deberá obtener un identificador de acceso que identifica unívocamente al comercio.
Para ello se debe invocar el método GetAuthenticationToken (), que recibe como parámetro el modelo AuthenticationModel.

AuthenticationModel			
Campo	**Descripción**	**Tipo**	**Requerido**
frase	Frase generada por Macro Click de Pago al registrar el comercio.	string	SI
guid	Identificador del comercio generado por Macro Click de Pago al registrar el comercio.	string	SI

```
var model = new AuthenticationModel()
{
  frase = "TX9GDIZNkfmoDtsZhbxo7T5ehF/taKsEyYSnAp8PKIA=",
  guid = "4fb0b8f9-0047-40e3-86a5-8b06168cea1e"
};

var access_token = ppConnector.GetAuthenticationToken(model);
```

Response

Código	Response	
200	code	200
	status	true
	message	Identificación del comercio correcta.
	data	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYmYiOjE1NDExMDYzODQsImV4cCI6MTU0MTEwNjk4NCwiaXNzIjoiaHR0cDovL2JvdG9uLmFwaWF1dGguZGV2LnNvbHVjaW9uZXNhbmRpbmFzLmNvbS5hci8iLCJhdWQiOiJodHRwOi8vYm90b24uYXBpcHVibGljYS5kZXYuc29sdWNpb25lc2FuZGluYXMuY29tLmFyLyJ9.WAvZxsE1oTQlZzAkzc0-HEYfMOrslP1s809q2lJM38M
401	code	401
	status	false
	message	No se pudo identificar al comercio.
	data	-

De ahora en adelante, se deberá pasar el identificador de acceso a todos los métodos que se deseen utilizar.

3. Obtener token de pago

Para poder realizar una transacción de pago se deberá obtener un identificador del pago. Para ello se debe invocar el método GetPaymentToken (), que recibe como parámetro un modelo TokenModel junto con el token obtenido anteriormente y la SecretKey que provee Macro Click de Pago al momento de registrar el comercio.

TokenModel			
Campo	**Descripción**	**Tipo**	**Requerido**
Comercio	Identificador del comercio generado por Macro Click de Pago al registrar el comercio.	string	SI
SucursalComercio	Identificador de la sucursal otorgado por Macro Click de Pago al registrar el comercio.	string	NO
Productos	Array de productos que se van a pagar.	String []	NO
TotalOperacion	Monto de la operación. Los últimos dos caracteres representan los decimales. Ej.: $ 2.00 → 200	string	SI
TransaccionComercioId	Identificador ÚNICO de la transacción en el comercio.	string	SI
IP	Dirección IP del cliente que está realizando la compra. Se deberá calcular cada vez que se haga una transacción de pago.	string	SI

Importante: El campo IP es un mecanismo de seguridad que sirve para verificar que el cliente que solicita el pago, sea el mismo que ejecuta el pago, por lo tanto, es responsabilidad del implementador realizar el cálculo de la misma en las dos etapas del pago (paso 3 y paso 5). En la **Referencia 3**, se indica un ejemplo de cómo obtener la dirección IP.

```
var tokenModel = new TokenModel ()
     {
          Comercio = "2c4179ef-24db-493d-afed-7c59392bd556",
        SucursalComercio = null,
        Productos = new String[] {"Producto1", "Producto2"},
        TotalOperacion = "20000",
        TransaccionComercioId = "231945",
        Ip = "140.32.56.3"
     };
```

```
var payment_token = ppConnector.GetPaymentToken(tokenModel, access_token,
"comercio_bafa1264-ad90-443f-b994-df156ee6357c")
```

Response

Código	Response	
200	code	200
	status	true
	message	Token de pago generado correctamente.
	data	bf7f438a-90ab-41dd-a3a1-004464881f1d
400	code	400
	status	false
	message	Referencia 1
	data	-
400	code	400
	status	false
	message	La dirección IP del comprador no puede ser nula o vacía
	data	-
500	code	500
	status	false
	message	Tiempo de espera del servidor excedido. SQL Timeout
	data	-

4. Obtención de medios de pago (OPCIONAL)

El comercio podrá obtener los medios de pago disponibles para operar en Macro Click de Pago y deberá pasar el código del mismo a la hora de ejecutar el pago. Para ello deberá realizar una llamada al método GetPaymentMethods() pasando como parámetro el identificador de acceso obtenido en el paso 2.

```
var mediosPago = ppConnector.GetPaymentMethods(access_token);
```

Response

Código	Response	
200	code	200
	status	true
	message	Medios de pago listados correctamente.
	data	[{"MedioPagoId":1023,"Nombre":"Visa Credito"," Descripcion"}, {"MedioPagoId":1024,"Nombre":"Maestro","Descripcion":""}, {"MedioPagoId"1027,
400	code	400
	status	false
	message	Error al listar los medios de pago.
	data	-

5. Ejecución de un pago

Para ejecutar el pago de una transacción el comercio deberá hacer uso de el token de pago generado en el paso 3 e invocar al método ExecutePayment() que recibe como parámetro un modelo del tipo PaymentModel, el identificador de acceso y la SecretKey provista por Macro Click de Pago al registrar el comercio.

PaymentModel			
Campo	**Descripción**	**Tipo**	**Requerido**
DatosTarjeta	Objeto del tipo DatosTarjeta	DatosTarjeta	SI
AceptaHabeasData	Control de la aceptación de Habeas Data	bool	SI
AceptTerminosyCondiciones	Control de la aceptación de Términos y condiciones.	bool	SI
CantidadCuotas	Numero de cuotas en las que se va a pagar la compra.	int	SI
Hash	-	-	NO
IPCliente	IP del cliente que está realizando la compra	string	SI
MedioPagoId	Identificador del medio de pago.		

DatosTarjeta			
Campo	**Descripción**	**Tipo**	**Requerido**
AñoVencimiento	Año en que caduca la tarjeta	string	SI
MesVencimiento	Mes en que caduca la tarjeta	string	SI
CodigoTarjeta	CVV – Código de seguridad de la tarjeta	string	SI
DocumentoTitular	Número de documento del titular de la tarjeta	string	SI
Email	Email del usuario que realiza la compra. A este email se enviará la confirmación del pago.	string	SI
FechaNacimientoTitular	Fecha de nacimiento del titular de la tarjeta. En forma DDMMAAAA	string	SI
NumeroPuertaResumen	N.º del domicilio dónde recibe el resumen de la tarjeta el titular de la misma.	string	NO
NumeroTarjeta	Número de la tarjeta	string	SI
TipoDocumento	Tipo de Documento del Titular de la tarjeta (DNI)	string	SI
TitularTarjeta	Nombre y apellido del titular de la tarjeta tal cual sale en la tarjeta.	string	SI

Importante: El campo IP es un mecanismo de seguridad que sirve para verificar que el cliente que solicita el pago, sea el mismo que ejecuta el pago, por lo tanto, es responsabilidad del implementador realizar el cálculo de la misma en las dos etapas del pago (paso 3 y paso 5). En la Referencia 3, se indica un ejemplo de cómo obtener la dirección IP.

```
var paymentModel = new PaymentModel ()
{
    DatosTarjeta = new DatosTarjeta ()
    {
       AñoVencimiento = "20",
       MesVencimiento = "08",
       CodigoTarjeta = "617",
       DocumentoTitular = "25123456",
       Email = "email@gmail.com",
       FechaNacimientoTitular = "12031993",
       NumeroPuertaResumen = "20",
       NumeroTarjeta = "4304968001555104",
       TipoDocumento = "DNI",
       TitularTarjeta = "JUAN PEREZ"
     },
    AceptaHabeasData = true,
    AceptTerminosyCondiciones = true,
    CantidadCuotas = 1,
    Hash = null,
    IPCliente = "140.32.56.3",
    MedioPagoId = 1023
};
```

```
var response_payment = ppConnector.ExecutePayment(paymentModel, payment_token, access_token, "comercio_bafa1264-ad90-443f-b994-df156ee6357c");
```

Response

Código	Response	
200	code	200
	status	true
	message	Pago realizado correctamente.
	data	{\"TransaccionComercioId\":\"028028\",\"TransaccionPlataformaId\":\"24336\",\"-Tipo\": \"PAGO\",\"Monto\":\"20,00\",\"Estado\":\"REALIZADA\",\"Detalle\":\"Pago realizado exitosamente.\",\"MetodoPago\":\"2\",\"MedioPago\":\"8\",\"EstadoId\":\"3\", \"Cuotas\":\"1\",\"InformacionPagador\":{\"Email\":\"test@gmail.com\", \"Nombre\":\"test\",\"NumeroDocumento\":\"25123456\",\"Telefono\":null,\"TipoDocumento\": \"DNI\"},\"Resultado\":{\"CodigoInterno\":\"00\",\"Descripcion\":\"Pago correcto\"}}

Código	Response	
400	code	400
	status	false
	message	Referencia 2
	data	{\"TransaccionComercioId\":\"058704\",\"TransaccionPlataformaId\":\"24337\",\"Tipo\":\"PAGO\",\"Monto\":\"20,00\",\"Estado\":\"RECHAZADA\",\"Detalle\":\"El pago fue rechazado.\",\"MetodoPago\":\"2\",\"MedioPago\":\"8\",\"EstadoId\":\"4\",\"Cuotas\":\"1\",\"InformacionPagador\":{\"Email\":\"test@gmail.com\",\"Nombre\":\"Test\",\"NumeroDocumento\":\"25123456\",\"Telefono\":null,\"TipoDocumento\":\"DNI\"},\"Resultado\":{\"CodigoInterno\":\"54\",\"Descripcion\":\"Tarjeta expirada\"}}
500	code	500
	status	false
	message	Tiempo de espera del servidor excedido. SQL Timeout
	data	-

1.2.2 Flujo de una consulta

1. Inicializar el conector:

```
PPConnector ppConnector = new PPConnector (AmbienteHelper.Ambiente.SANDBOX);
```

El mismo recibe como parámetro el ambiente en el que se está desarrollando la integración (Sandbox o Producción). El conector permitirá el acceso a los métodos expuestos por la API Pública de Macro Click de Pago.

2. Obtener token de acceso

Para poder interactuar con los métodos expuestos por Macro Click de Pago, se deberá obtener un identificador de acceso que identifica unívocamente al comercio.
Para ello se debe invocar el método GetAuthenticationToken (), que recibe como parámetro el modelo AuthenticationModel.

AuthenticationModel	
Campo	Descripción
frase	Frase generada por Macro Click de Pago al registrar el comercio.
guid	Identificador del comercio generado por Macro Click de Pago al registrar el comercio.

```
var model = new AuthenticationModel ()
{
   frase = "TX9GDIZNkfmoDtsZhbxo7T5ehF/taKsEyYSnAp8PKIA=";
   guid = "4fb0b8f9-0047-40e3-86a5-8b06168cea1e"
};

var access_token = ppConnector.GetAuthenticationToken(model);
```

Response

Código	Response	
200	code	200
	status	true
	message	Identificación del comercio correcta.
	data	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYmYiOjE1NDExMDYzODQsImV4cCI6MTU0MTEwNjk4NCwiaXNzIjoiaHR0cDovL2JvdG9uLmFwaWF1dGguZGV2LnNvbHVjaW9uZXNhbmRpbmFzLmNvbS5hci8iLCJhdWQiOiJodHRwOi8vYm90b24uYXBpcHVibGljYS5kZXYuc29sdWNpb25lc2FuZGluYXMuY29tLmFyLyJ9.WAvZxsE1oTQlZzAkzc0-HEYfMOrslP1s809q2lJM38M
401	code	401
	status	false
	message	No se pudo identificar al comercio.
	data	-

De ahora en adelante, se deberá pasar el identificador de acceso a todos los métodos que se deseen utilizar.

3. Obtener el listado de transacciones del comercio

El comercio podrá obtener un listado de las transacciones y el estado en el que se encuentran, invocando al método GetTransactions () que recibe como parámetro el identificador de acceso.
Filtros:

- TransaccionId – string
- Fecha desde – string (dd/MM/yyyy)
- Fecha Hasta – string (dd/MM/yyyy)
- EstadoTransaccion – string
- TransaccionComercioId – string
- NumeroSucursal – string
- Caja – string (código)
- Pagina – int
- Cantidad – int

Importante: los filtros por Fecha y TransaccionComercioId son obligatorios.

```
var transacciones = ppConnector.GetTransactions(access_token, transaccionId, fechaDesde, fechaHasta, estadoTransaccion, transaccionComercioId, numeroSucursal, caja, pagina, cantidad);
```

Response

Código	Response	
200	code	200
	status	true
	message	Transacciones listadas correctamente.
	data	“{\”Transacciones\”: [{“TransaccionId”:7638,”Estado”:”RECHAZADA”, ”Fecha”:”2018-09-19T13:14:46.527278”,”Moneda”:”ARS”, ”Monto”:”2510,51”,”Productos”:”Producto 1, Producto 2”,”Sucursal”: null, ”MedioPagoId”:”1024”,”MedioPago”:”Maestro”, “TransaccionComercioId”:”431787”}, {“TransaccionId”:7639,”Estado”: “REALIZADA”,”Fecha”:”2018-09-19T13:17:41.417821”,”Mone-da”:”ARS”,”Monto”:”234,44”,”Productos”: “test”, “Sucursal”: null,”MedioPagoId”:”1023”,”MedioPago”:”Visa Credito”, ”TransaccionComercioId”:”34343443”,”Result”:{“Codigo”:”00”, ”Descripcion”:”Pago correcto”}, {“TransaccionId”:7642, “Estado”:”EXPIRADA”,”Fecha”:”2018-09-26T15:27:28.4327796”, ”Moneda”:”ARS”,”Monto”:”200,00”, “Productos”: aaa, bbb”,”Sucursal”: null, “MedioPagoId”: null, “MedioPago”: null, “TransaccionComercioId”:”613419”,”Result”:{“Codigo”:”00”,”Descripcion”: ”Pago correcto”}], \”Pagina\”:3,\”Cantidad\”:2,\”TotalRows\”:1068}”
400	code	400
	status	false
	message	Error al listar las transacciones.
	data	-
400	code	400
	status	false
	message	“Debe ingresar al menos el dato TransaccionComercioId o FechaDesde/Hasta”
	data	-

Estados Transacción		
Código	**Valor**	**Descripción**
1	CREADA	Transacción creada (no procesada)
2	EN_PAGO	Transacción en pago
3	REALIZADA	Transacción procesada con éxito
4	RECHAZADA	Transacción rechazada
5	ERROR_VALIDACION_HASH_TOKEN	Ocurrió un error en la validación de la firma enviada al crear la transacción
6	ERROR_VALIDACION_HASH_PAGO	Ocurrió un error en la validación de la firma al intentar pagar la transacción
7	EXPIRADA	Transacción expirada
8	CANCELADA	Transacción cancelada por el usuario
9	DEVUELTA	Transacción devuelta
10	PENDIENTE	Debin creado correctamente, esperando aprobación.
11	VENCIDA	Transacción vencida porque el DEBIN expiró.

4. Obtener el detalle de una transacción en particular

El comercio podrá obtener el detalle de una transacción en particular invocando al método GetTransactionByTxComercioId () que recibe como parámetro el identificador de acceso y el identificador de la transacción en el comercio.

```
var transaccion = ppConnector.GetTransactionByTxComercioId(access_token, "063337");
```

Response

Código	Response	
200	code	200
	status	true
	message	Transaccion identificada como <063337> listada correctamente.
	data	{"TransaccionId":7759,"Estado":"REALIZADA","Fecha":"2018-10-25T10:59:08.7145689","Moneda":"ARS","Monto":"20,00","Productos":"Producto1, Producto2, ","Sucursal": null,"MedioPagoId":"1023","MedioPago":"Visa Credito","TransaccionComercioId":"063337","Result":{"Codigo":"00","Descripcion":"Pago correcto"}
400	code	400
	status	false
	message	Error al listar la transacción solicitada.
	data	-

1.2.3 Órdenes de pago

1. Inicializar el conector:

```
PPConnector ppConnector = new PPConnector (AmbienteHelper.Ambiente.SANDBOX);
```

El mismo recibe como parámetro el ambiente en el que se está desarrollando la integración (Sandbox o Producción). El conector permitirá el acceso a los métodos expuestos por la **API Pública de Macro Click de Pago.**

2. Obtener token de acceso

Para poder interactuar con los métodos expuestos por Macro Click de Pago, se deberá obtener un identificador de acceso que identifica unívocamente al comercio.
Para ello se debe invocar el método GetAuthenticationToken (), que recibe como parámetro el modelo AuthenticationModel.

AuthenticationModel	
Campo	**Descripción**
frase	Frase generada por Macro Click de Pago al registrar el comercio.
guid	Identificador del comercio generado por Macro Click de Pago al registrar el comercio.

```
var model = new AuthenticationModel ()
{
   frase = "TX9GDIZNkfmoDtsZhbxo7T5ehF/taKsEyYSnAp8PKIA=",
   guid = "4fb0b8f9-0047-40e3-86a5-8b06168cea1e"
};

var access_token = ppConnector.GetAuthenticationToken(model);
```

Response

Código	Response	
200	code	200
	status	true
	message	Identificación del comercio correcta.
	data	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYmYiOjE1NDExMDYzODQsImV4cCI6MTU0MTEwNjk4NCwiaXNzIjoiaHR0cDovL2JvdG9uLmFwaWF1dGguZGV2LnNvbHVjaW9uZXNhbmRpbmFzLmNvbS5hci8iLCJhdWQiOiJodHRwOi8vYm90b24uYXBpcHVibGljYS5kZXYuc29sdWNpb25lc2FuZGluYXMuY29tLmFyLyJ9.WAvZxsE1oTQlZzAkzc0-HEYfMOrslP1s809q2lJM38M
401	code	401
	status	false
	message	No se pudo identificar al comercio.
	data	-

De ahora en adelante, se deberá pasar el identificador de acceso a todos los métodos que se deseen utilizar.

Dar de alta una caja

El comercio podrá dar de alta cajas para sus sucursales, invocando al método Caja() que recibe como parámetro el identificador de acceso, un objeto del tipo CajaModel y la secretkey provista por Macro Click de Pago al registrar el comercio.

CajaModel			
Campo	**Descripción**	**Tipo**	**Requerido**
Nombre	Nombre de la caja	string	SI
Codigo	Identificador alfanumérico de la caja	string	SI
SucursalComercioId	Identificador de la sucursal en el que se creará la caja	int	SI
Fixed_amount	Enviar el valor true si la caja opera con ordenes de pago o false en caso contrario.	Bool	SI

```
var model_caja = new CajaModel()
{
      Codigo = "ABC123",
      Fixed_Amount = true,
      Nombre = "Caja 2",
      SucursalComercioId = 3016
};

var caja = ppConnector.Caja(model_caja, access_token.data, "comercio_bafa1264-ad90-443f-b994-df156ee6357c");
```

Response

Código	Response	
201	code	201
	status	true
	message	[Nombre caja] creada correctamente.
	data	{ "CajaId":29, "GUID":"1f255edd-d354-490c-9a2b-5cf54b686bff", "Codigo":"ABC123", "Nombre":"Caja de ejemplo", "Fixed_Amount":true, "Habilitado":true, "TSCreate":"2019-09-18T22:36:40.2640222-03:00", "TSModificado":null, "TSEliminado":null, "SucursalComercioId":3016 }

Código	Response	
400	code	400
	status	false
	message	Ocurrió un error al crear la caja para la Sucursal.
	data	null
403	code	403
	status	false
	message	El servidor rechazó la conexión con el servicio.
	data	-

Dar de baja una caja

• Por código de caja

El comercio podrá eliminar una caja, invocando al método **DeleteCajaByCodigo()** que recibe como parámetro el identificador de acceso y código asignado para identificar a la caja.

```
var eliminar_caja = ppConnector.DeleteCajaByCodigo(access_token.data, "ABC123");
```

Response

Código	Response	
200	code	200
	status	true
	message	Caja con código <codigo_caja> se eliminó correctamente.
	data	-
400	code	400
	status	false
	message	Ocurrió un error al eliminar la caja.
	data	null
403	code	403
	status	false
	message	El servidor rechazó la conexión con el servicio.
	data	-

• Por identificador de caja

El comercio podrá eliminar una caja invocando al método **DeleteCajaById()** que recibe como parámetro el identificador de acceso y el identificador interno de la caja en Macro Click de Pago. Esta información se devuelve al consultar por una caja en el campo CajaId.

```
var eliminar_caja = ppConnector.DeleteCajaById(access_token.data, 245);
```

Response

Código	Response	
200	code	200
	status	true
	message	Caja con código <codigo_caja> se eliminó correctamente.
	data	-
400	code	400
	status	false
	message	Ocurrió un error al eliminar la caja.
	data	null
403	code	403
	status	false
	message	El servidor rechazó la conexión con el servicio.
	data	-

Obtener detalle de una caja

• Por código de caja

El comercio podrá consultar el detalle de una caja creada con anterioridad invocando al método **GetDetailCajaByCodigo( )** que recibe como parámetros el identificador de acceso y el código de la caja que se quiere consultar.

```
var detalleCaja = ppConnector.GetDetailCajaByCodigo(access_token.data, "ABC123");
```

Response

Código	Response	
200	code	200
	status	true
	message	Caja con código <codigo_caja> se eliminó correctamente.
	data	{ "CajaId":46, "GUID":"aff25c6d-1d8e-43f1-94f8-cd2f27438cfa", "Codigo":"TEST1234", "Nombre":"Caja Lucia", "Fixed_Amount":true, "SucursalComercioId":2013 }
400	code	404
	status	false
	message	No se encontró caja con código <codigo_caja>.
	data	null

• Por identificador de caja

El comercio podrá consultar el detalle de una caja creada con anterioridad invocando al método **GetDetailCajaById( )** que recibe como parámetros el identificador de acceso y el identificar de la caja en Macro Click de Pago, que se quiere consultar.

```
var detalleCaja = ppConnector.GetDetailCajaById(access_token.data, 245);
```

Response

Código	Response	
200	code	200
	status	true
	message	Caja con ID <identificador_caja> obtenida correctamente.
	data	{ "CajaId":46, "GUID":"aff25c6d-1d8e-43f1-94f8-cd2f27438cfa", "Codigo":"TEST1234", "Nombre":"Caja Lucia", "Fixed_Amount":true, "SucursalComercioId":2013 }
400	code	404
	status	false
	message	No se encontró caja con ID <identificador_caja>.
	data	null

Generar una orden de pago

Si el comercio desea cobrar con QR, deberá generar una orden de pago para una caja. Para ello, deberá invocar al método **Order()** y enviar como parámetro un objeto del tipo OrderModel, el identificador de acceso, la secretkey provista por Macro Click de Pago al registrar el comercio, el identificador de la caja en la que se creará la orden y opcionalmente, un tiempo de expiración para esa orden.

OrdenModel			
Campo	**Descripción**	**Tipo**	**Requerido**
MontoTotal	Monto por pagar. Los últimos dos posiciones representan los decimales. Ej.: 12,00 → "1200"	string	SI
IdTransaccionInterno	Identificador alfanumérico de la orden	string	SI
Productos	Descripción de los productos a pagar	string	NO
UrlNotificacion	URL a la que se notifica, en caso de dejar vacío si notifica a la configurada al dar de alta el comercio en Macro Click de Pago	string	NO

La orden expira a un tiempo definido por Macro Click de Pago al ser creada. Si se desea expirarla a un tiempo diferente, se deberá enviar al invocar el método de creación de orden, en el parámetro **ttlPreference** en segundos.

Creación de orden sin establecer **ttlPreference**, expirando según tiempo máximo default de **Macro Click de Pago**:

```
var order = new OrderModel()
 {
     MontoTotal = "100", // $1,00
     IdTransaccionInterno = "ORDEN001",
     Productos = "Producto1, Producto2"
 };
var result = ppConnector.Order(order, "ABC123", access_token.data, "comercio_bafa1264-ad90-
443f-b994-df156ee6357c");
```

Creación de orden estableciendo **ttlPreference**:

```
 var order = new OrderModel()
 {
     MontoTotal = "100", // $1,00
     IdTransaccionInterno = "ORDEN001",
     Productos = "Producto1, Producto2"
 };
var result = ppConnector.Order(order, "ABC123", access_token.data, "comercio_bafa1264-ad90-
443f-b994-df156ee6357c", 120);
```

• El **ttlPreference** debe ser un valor entero expresado en segundos. En este ejemplo, 120 segundos de expiración para la orden creada.

Código	Response	
201	code	201
	status	true
	message	La orden de pago se creó correctamente.

	data	{
		"CajaId":29,
		"GUID":"1f255edd-d354-490c-9a2b-5cf54b686bff",
		"Codigo":"ABC123",
		"Nombre":"Caja de ejemplo",
		"Fixed_Amount":true,
		"Habilitado":true,
		"TSCreate":"2019-09-18T22:36:40.2640222-03:00",
		"TSModificado":null,
		"TSEliminado":null,
		"SucursalComercioId":3016
		}

400	code	400
	status	false
	message	No se encontró una caja con el código [codigo_caja].
	data	-
400	code	400
	status	false
	message	La URL de notificación no tiene un formato válido.
	data	-
400	code	400
	status	false
	message	IdTransaccionInterno no puede ser nulo.
	data	-
400	code	400
	status	false
	message	TtlPreference no tiene un formato válido.
	data	-
400	code	400
	status	false
	message	Ocurrió un error al crear la orden de pago no se pudo crear.
	data	-

Consultar por una orden

El comercio tiene la posibilidad de consultar el detalle de una orden que haya creado con anterioridad. Para ello deberá invocar al método GetOrderById() y enviar como parámetro el identificador de acceso (access_token.data) y el identificador de la orden en Macro Click de Pago. Este identificador se obtiene al crear una orden, en el campo OrdenId.

```
var detalleOrden = ppConnector.GetOrdenById(access_token.data, "23");
```

Response

Código	Response	
200	code	200
	status	true
	message	Orden de pago con ID <23> obtenida correctamente.

	data	{
		"OrdenId":265,
		"IdTransaccionInterno":"ORDEN_12",
		"UrlNotificacion":null,
		"MontoTotal":10.00,
		"Productos":"Producto1, Producto2",
		"CajaId":54,
		"FechaCreacion":"2020-01-03T12:59:02.6081937",
		"FechaExpiracion":"2020-02-02T12:59:02.6041596",
		"NombreCaja":"Lucia 10",
		"Estado":"PENDIENTE"
		}

400	code	404
	status	false
	message	No se encontró una orden con ID <23>.
	data	-

Consultar la orden vigente por caja

• Consultar por código de caja

El comercio tiene la posibilidad de consultar por la orden que se encuentre vigente en una caja. Para ello deberá invocar al método GetOrder() y enviar como parámetro el identificador de acceso (access_token.data) y el código que identifica a la caja.

```
var orden = ppConnector.GetOrder(access_token.data, "ABC123");
```

Código	Response	
200	code	200
	status	true
	message	Ultima orden pendiente de la caja con codigo [codigo de la caja] se obtuvo correctamente.
	data	{ "OrdenId":29, "IdTransaccionInterno":"ORDEN001", "UrlNotificacion": null, "MontoTotal": 12.40 , "Productos": "Producto1, Producto2", "CajaId": "29", "FechaCreacion":"2019-09-18T22:36:40.2640222-03:00", "FechaExpiracion":"2019-09-18T22:50:40.2640222-03:00", "NombreCaja": "ABC123", "Estado": "PENDIENTE" }
400	code	404
	status	false
	message	No se encontró ninguna orden vigente con el codigo ingresado.
	data	-

ESTADO DE UNA ORDEN		
1	PENDIENTE	Orden pendiente de pago
2	APROBADA	Orden pagada con éxito
3	RECHAZADA	Orden con pago rechazado
4	EXPIRADA	Orden vencida o expirada

• Consultar por ID de caja

El comercio tiene la posibilidad de consultar por la orden que se encuentre vigente en una caja. Para ello deberá invocar al método GetOrderByCajaId() y enviar como parámetro el identificador de acceso (access_token.data) y el identificador de la caja, devuelto al momento de crear la caja en el campo **CajaId**.

```
var orden = ppConnector.GetOrderByCajaId(access_token.data, 29);
```

Código	Response	
200	code	200
	status	true
	message	Ultima orden pendiente de la caja con ID [id de la caja] se obtuvo correctamente.

	data	{
		"CajaId":29,
		"GUID":"1f255edd-d354-490c-9a2b-5cf54b686bff",
		"Codigo":"ABC123",
		"Nombre":"Caja de ejemplo",
		"Fixed_Amount":true,
		"Habilitado":true,
		"TSCreate":"2019-09-18T22:36:40.2640222-03:00",
		"TSModificado":null,
		"TSEliminado":null,
		"SucursalComercioId":3016
		}

400	code	404
	status	false
	message	No se encontró ninguna orden vigente con el codigo ingresado.
	data	-

Eliminar una orden

El comercio podrá eliminar una orden de una caja, invocando al método DeleteOrder() y enviando como parámetros el identificador de acceso, junto con el identificador de la caja.

```
var result = ppConnector.DeleteOrder(access_token.data, "ABC123");
```

Código	Response	
200	code	200
	status	true
	message	Orden eliminada correctamente.
	data	-
400	code	404
	status	false
	message	No se encontró una orden para eliminar en la caja con código [codigo_caja]
	data	-

• JAVA – ANDROID

• En un proyecto gradle, incluir lo siguiente en el archivo build.gradle:

```
repositories {
    jcenter()
}
dependencies {
    implementation 'ar.com.pluspagos:ppconnector-java:0.2.13'
    implementation 'com.squareup.retrofit2:retrofit:2.5.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.5.0'
}
```

• En un proyecto Maven, incluir lo siguiente en el archivo pom.xml:

```
<project>
    ...
    <dependencies>
    <dependency>
          <groupId>ar.com.pluspagos</groupId>
          <artifactId>ppconnector-java</artifactId>
          <version>0.2.13</version>
          <type>pom</type>
    </dependency>
    <dependency>
          <groupId>com.squareup.retrofit2</groupId>
          <artifactId>retrofit</artifactId>
          <version>2.5.0</version>
    </dependency>
    <dependency>
      <groupId>com.squareup.retrofit2</groupId>
      <artifactId>converter-gson</artifactId>
      <version>2.5.0</version>
    </dependency>
    </dependencies>
    <repositories>
    <repository>
          <id>central</id>
          <name>bintray</name>
          <url>https://jcenter.bintray.com</url>
    </repository>
    <repositories>
    ...
</project>
```

Imports

```
import ar.com.pluspagos.ppconnector.*;
import ar.com.pluspagos.ppconnector.models.*;
```

Definición de datos de conexión

```
String comercio = "573cc80e-5916-47cd-8028-b288932052a0";
String frase = "K74kjShQOdC6tnbPUocgRdZOT4QUo38QauCcULFYX14=";
String secretKey = "WaveApps_77cf0de1-99be-4f43-893d-e5880edbf208";7
```

• Ambientes

El SDK permite la utilización de dos ambientes: sandbox y producción que deberán indicarse de la siguiente manera:

```
PPConnector.init(AmbienteHelper.Ambiente.SANDBOX, comercio, frase);
```

```
PPConnector.init(AmbienteHelper.Ambiente.PRODUCTION, comercio, frase);
```

Tests

Todas las respuestas de la API Pública de Macro Click de Pago coinciden con siguiente esquema:

Response		
Campo	**Descripción**	**Tipo**
status	Estado en el que finalizó el request	bool
code	Status Code del request	int
message	Mensaje informativo correspondiente a la operación realizada.	string
data	Contenido útil para el comercio. Se proveen los modelos necesarios para realizar la conversión del contenido.	string

El siguiente método permite visualizar las respuestas por consola:

```
private static void log(String titulo, Response r) {
    System.out.println(titulo);
    String l = "code:\t\t" + r.getCode() + "\n" +
            "status:\t\t" + r.isStatus() + "\n" +
            "message:\t" + r.getMessage() + "\n" +
            "data:\t\t" + r.getData();
    System.out.println(l);
    System.out.println();
}
```

• HealthCheck

Este método permite conocer el estado actual de la API de Macro Click de Pago.

```
PPConnector.healthCheck(new PPCallback() {
    @Override
    public void onFinished(Response response) {
        log("HealthCheck", response);
    }
});
```

Código	Response	
200	code	200
	status	true
	message	Servicio disponible.
	data	Versión 1.0
400	code	400
	status	false
	message	Servicio no disponible.
	data	-

1.2.1 Flujo de un pago

1. Inicializar el conector:

```
PPConnector.init(AmbienteHelper.Ambiente.SANDBOX, comercio, frase);
```

El mismo recibe como parámetro el ambiente en el que se está desarrollando la integración (Sandbox o Producción), el identificador del comercio y la frase provista por Macro Click de Pago. El conector permitirá el acceso a los métodos expuestos por la API Pública de Macro Click de Pago.

2. Obtener token de pago

Para poder realizar una transacción de pago se deberá obtener un identificador del pago. Para ello se debe invocar el método getPaymentToken (), que recibe como parámetro un modelo TokenModel junto con la SecretKey que provee Macro Click de Pago al momento de registrar el comercio.

TokenModel			
Campo	**Descripción**	**Tipo**	**Requerido**
Comercio	Identificador del comercio generado por Macro Click de Pago al registrar el comercio.	string	SI
SucursalComercio	Identificador de la sucursal otorgado por Macro Click de Pago al registrar el comercio.	string	NO
Productos	Array de productos que se van a pagar.	String []	NO
TotalOperacion	Monto de la operación. Los últimos dos caracteres representan los decimales. Ej.: $ 2.00 → 200	string	SI
TransaccionComercioId	Identificador ÚNICO de la transacción en el comercio.	string	SI
IP	Dirección IP del cliente que está realizando la compra. Se deberá calcular cada vez que se haga una transacción de pago.	string	SI

Importante: El campo IP es un mecanismo de seguridad que sirve para verificar que el cliente que solicita el pago, sea el mismo que ejecuta el pago, por lo tanto, es responsabilidad del implementador realizar el cálculo de la misma en las dos etapas del pago (paso 2 y paso 4). En la **Referencia 3**, se indica un ejemplo de cómo obtener la dirección IP.

```
String publicIp = "207.248.125.4";
String urlDominio = "www.google.com";

TokenModel tokenModel = new TokenModel();
tokenModel.setComercio(comercio);
tokenModel.setIp(publicIp);
tokenModel.setUrlDominio(urlDominio);
tokenModel.setSucursalComercio(null);
tokenModel.setProductos(new String[] {"Product 1", "Product 2"});
tokenModel.setTotalOperacion("200");
tokenModel.setTransaccionComercioId(UUID.randomUUID().toString());


    PPConnector.getPaymentToken(tokenModel, secretKey, new PPCallback() {
          @Override
          public void onFinished(Response resPT) {
                //realizar pago
          }
```

Response

Código		Response
200	code	200
	status	true
	message	Token de pago generado correctamente.
	data	bf7f438a-90ab-41dd-a3a1-004464881f1d
400	code	400
	status	false
	message	Referencia 1
	data	-

3. Obtención de medios de pago (OPCIONAL)

El comercio podrá obtener los medios de pago disponibles para operar en Macro Click de Pago y deberá pasar el código de este a la hora de ejecutar el pago. Para ello deberá realizar una llamada al método GetPaymentMethods().

```
PPConnector.getPaymentMethods(tokenModel, secretKey, new PPCallback() {
@Override
public void onFinished(Response res) {
   //medios de pago
}
});
```

Response

Código	Response	
200	code	200
	status	true
	message	Medios de pago listados correctamente.
	data	[{"MedioPagoId":1023,"Nombre":"Visa Credito"," Descripcion"}, {"MedioPagoId":1024,"Nombre":"Maestro","Descripcion":""}, {"MedioPagoId"1027, "Nombre":"Visa Débito","Descripcion":""}]
400	code	400
	status	false
	message	Error al listar los medios de pago.
	data	-

4. Ejecución de un pago

Para ejecutar el pago de una transacción el comercio deberá hacer uso del token de pago generado en el paso 2 e invocar al método executePayment() que recibe como parámetro un modelo del tipo PaymentModel, el identificador de acceso y la SecretKey provista por Macro Click de Pago al registrar el comercio.

PaymentModel			
Campo	**Descripción**	**Tipo**	**Requerido**
DatosTarjeta	Objeto del tipo DatosTarjeta	DatosTarjeta	SI
AceptaHabeasData	Control de la aceptación de Habeas Data	bool	SI
AceptTerminosyCondiciones	Control de la aceptación de Términos y condiciones.	bool	SI
CantidadCuotas	Numero de cuotas en las que se va a pagar la compra.	int	SI
Hash	-	-	NO
IPCliente	IP del cliente que está realizando la compra	string	SI
MedioPagoId	Identificador del medio de pago.		

DatosTarjeta			
Campo	**Descripción**	**Tipo**	**Requerido**
AñoVencimiento	Año en que caduca la tarjeta	string	SI
MesVencimiento	Mes en que caduca la tarjeta	string	SI
CodigoTarjeta	CVV – Código de seguridad de la tarjeta	string	SI
DocumentoTitular	Número de documento del titular de la tarjeta	string	SI
Email	Email del usuario que realiza la compra. A este email se enviará la confirmación del pago.	string	SI
FechaNacimientoTitular	Fecha de nacimiento del titular de la tarjeta. En forma DDMMAAAA	string	SI
NumeroPuertaResumen	N.º del domicilio dónde recibe el resumen de la tarjeta el titular de la misma.	string	NO
NumeroTarjeta	Número de la tarjeta	string	SI
TipoDocumento	Tipo de Documento del Titular de la tarjeta (DNI)	string	SI
TitularTarjeta	Nombre y apellido del titular de la tarjeta tal cual sale en la tarjeta.	string	SI

Importante: El campo IP es un mecanismo de seguridad que sirve para verificar que el cliente que solicita el pago, sea el mismo que ejecuta el pago, por lo tanto, es responsabilidad del implementador realizar el cálculo de la misma en las dos etapas del pago (paso 2 y paso 4). En la **Referencia 3**, se indica un ejemplo de cómo obtener la dirección IP.

```
DatosTarjeta dt = new DatosTarjeta();
dt.setAnoVencimiento("20");
dt.setMesVencimiento("08");
dt.setCodigoTarjeta("123");
dt.setDocumentoTitular("22222222");
dt.setEmail("email@gmail.com");
dt.setFechaNacimientoTitular("12031993");
dt.setNumeroPuertaResumen("20");
dt.setNumeroTarjeta("4507990000004905");
dt.setTipoDocumento("DNI");
dt.setTitularTarjeta("JUAN PEREZ");

PaymentModel pm = new PaymentModel();

pm.setAceptaHabeasData(true);
pm.setAceptTerminosyCondiciones(true);
pm.setCantidadCuotas(1);
pm.setIpCliente(publicIp);
pm.setMedioPagoId(4);
pm.setDatosTarjeta(dt);
pm.setAceptaHabeasData(true);
pm.setAceptTerminosyCondiciones(true);
pm.setCantidadCuotas(1);
pm.setIpCliente(publicIp);
pm.setMedioPagoId(4);
pm.setDatosTarjeta(dt);

PPConnector.getPaymentToken(tokenModel, secretKey, new PPCallback() {
    @Override
    public void onFinished(Response resPT) {
           PPConnector.executePayment(pm, resPT.getData(), secretKey, new PPCallback() {
           @Override
           public void onFinished(Response resPayment) {
                 //mostrar resultado del pago
                 }
              });
        }
    });
```

Response

Código	Response	
200	code	200
	status	true
	message	Pago realizado correctamente.
	data	-
400	code	400
	status	false
	message	Referencia 2
	data	-

1.2.2 Flujo de una consulta

1. Inicializar el conector:

```
PPConnector.init(AmbienteHelper.Ambiente.SANDBOX, comercio, frase);
```

El mismo recibe como parámetro el ambiente en el que se está desarrollando la integración (Sandbox o Producción), el identificador del comercio y la frase provista por Macro Click de Pago. El conector permitirá el acceso a los métodos expuestos por la API Pública de Macro Click de Pago.

2. Obtener el listado de transacciones del comercio

El comercio podrá obtener un listado de las transacciones y el estado en el que se encuentran, invocando al método getTransactions () que recibe como parámetro el identificador de acceso.

```
PPConnector.getTransactions(new PPCallback() {
@Override
public void onFinished(Response res) {
   //transacciones
}
});
```

Response

Código	Response	
200	code	200
	status	true
	message	Transacciones listadas correctamente.
	data	[{"TransaccionId":7638,"Estado":"RECHAZADA", "Fecha":"2018-09-19T13:14:46.527278","Moneda":"ARS", "Monto":"2510,51","Productos":"Producto 1, Producto 2","Sucursal": null, "MedioPagoId":"1024","MedioPago":"Maestro", "TransaccionComercioId":"431787"}, {"TransaccionId":7639,"Estado": "REALIZADA","Fecha":"2018-09-19T13:17:41.417821", "Moneda":"ARS","Monto":"234,44","Productos": "test", "Sucursal": null,"MedioPagoId":"1023","MedioPago":"Visa Credito", "TransaccionComercioId":"34343443","Result":{"código":"00", "descripcion":"Pago Correcto"}}, {"TransaccionId":7642, "Estado":"EXPIRADA","Fecha":"2018-09-26T15:27:28.4327796","Moneda":"ARS", "Monto":"200,00", "Productos": aaa, bbb","Sucursal": null, "MedioPagoId": null, "MedioPago": null, "TransaccionComercioId":"613419","Result":{"código":"00", "descripcion":"Pago Correcto"}}]
400	code	400
	status	false
	message	Error al listar las transacciones.
	data	-

3. Obtener el detalle de una transacción en particular

El comercio podrá obtener el detalle de una transacción en particular invocando al método getTransactionByTxComercioId () que recibe como parámetro el identificador de acceso y el identificador de la transacción en el comercio.

```
PPConnector.getTransactionByTxComercioId("097d8d7b-967b-48af-9d99-8d21666244b448",
new PPCallback() {
    @Override
    public void onFinished(Response res) {
          //detalle de una transacción
    }
});
```

Response

Código	Response	
200	code	200
	status	true
	message	Transaccion identificada como <063337> listada correctamente.
	data	{"TransaccionId":7759,"Estado":"REALIZADA","Fecha":"2018-10-25T10:59:08.7145689","Moneda":"ARS","Monto":"20,00","Productos":"Producto1, Producto2, ","Sucursal": null,"MedioPagoId":"1023","MedioPago":"Visa Credito","-TransaccionComercioId":"063337","Result":{"código":"00", "descripcion":"Pago Correcto"}}
400	code	400
	status	false
	message	Error al listar la transacción solicitada.
	data	-

1.2.3 Órdenes de pago

Dar de alta una caja

El comercio podrá dar de alta cajas para sus sucursales, invocando al método **PPConnector.caja(CajaModel cajaModel, String secretKey)** que recibe como parámetro un objeto del tipo CajaModel y la secretkey provista por Macro Click de Pago al registrar el comercio.

CajaModel			
Campo	**Descripción**	**Tipo**	**Requerido**
Nombre	Nombre de la caja	string	SI
Codigo	Identificador alfanumérico de la caja	string	SI
SucursalComercioId	Identificador de la sucursal en el que se creará la caja	int	SI
Fixed_amount	Enviar el valor true si la caja opera con órdenes de pago o false en caso contrario.	boolean	SI

```
CajaModel caja = new CajaModel();
caja.setNombre("Caja 2");
caja.setSucursalComercioId(15715);
caja.setCodigo("ABC123");
caja.setFixedAmount(true);
PPConnector.caja(caja, secretKey, new PPCallback() {
    @Override
    public void onFinished(Response res) {
          //resultado de la creación de la caja
    }
});
```

Código	Response	
201	code	201
	status	true
	message	[Nombre caja] creada correctamente.
	data	{ "CajaId":29, "GUID":"1f255edd-d354-490c-9a2b-5cf54b686bff", "Codigo":"ABC123", "Nombre":"Caja de ejemplo", "Fixed_Amount":true, "Habilitado":true, "TSCreate":"2019-09-18T22:36:40.2640222-03:00", "TSModificado":null, "TSEliminado":null, "SucursalComercioId":15715 }

Código	Response	
400	code	400
	status	false
	message	Ocurrió un error al crear la caja para la Sucursal.
	data	null
403	code	403
	status	false
	message	El servidor rechazó la conexión con el servicio.
	data	-

Generar una orden de pago

Si el comercio desea cobrar con QR, deberá generar una orden de pago para una caja. Para ello, deberá invocar al método **PPConnector.order(OrderModel order, String secretKey, String codigo, String ttlPreference)** y enviar como parámetro un objeto del tipo OrderModel, la secretkey provista por Macro Click de Pago al registrar el comercio, el identificador de la caja en la que se creará la orden (codigo) y opcionalmente, un tiempo de expiración para esa orden (ttlPreference).

OrdenModel			
Campo	**Descripción**	**Tipo**	**Requerido**
MontoTotal	Monto por pagar. Los últimos dos posiciones representan los decimales. Ej.: 12,00 → "1200"	string	SI
IdTransaccionInterno	Identificador alfanumérico de la orden	string	SI
Productos	Descripción de los productos a pagar	string	NO
UrlNotificacion	URL a la que se notifica, en caso de dejar vacío si notifica a la configurada al dar de alta el comercio en Macro Click de Pago	string	NO
Fixed_amount	Enviar el valor true si la caja opera con órdenes de pago o false en caso contrario.	boolean	SI

La orden expira a un tiempo definido por Macro Click de Pago al ser creada. Si se desea expirarla a un tiempo diferente, se deberá enviar al invocar el método de creación de orden, en el parámetro **ttlPreference** en segundos.

```
OrderModel order = new OrderModel ();
order.setMontoTotal("1284");
order.setFixedAmount(true);
order.setIdTransaccionInterno("ORDEN001");
order.setProductos("Producto 1, Producto 2");
PPConnector.order(order, secretKey,  "ABC123", null, new PPCallback() {
    @Override
    public void onFinished(Response res) {
          //resultado de la creación de la orden
    }
});
```

Código	Response	
201	code	201
	status	true
	message	[Nombre caja] creada correctamente.
	data	{ "CajaId":29, "GUID":"1f255edd-d354-490c-9a2b-5cf54b686bff", "Codigo":"ABC123", "Nombre":"Caja de ejemplo", "Fixed_Amount":true, "Habilitado":true, "TSCreate":"2019-09-18T22:36:40.2640222-03:00", "TSModificado":null, "TSEliminado":null, "SucursalComercioId":3016 }
500	code	500
	status	false
	message	Ocurrió un error interno, la orden de compra no se pudo crear.
	data	-
400	code	400
	status	false
	message	Error al crear la orden de pago.
	data	-

Consultar la orden vigente por caja

• Consultar por código de caja

El comercio tiene la posibilidad de consultar por la orden que se encuentre vigente en una caja. Para ello deberá invocar al método **PPConnector.getOrder(String codigo)** y enviar como parámetro el código que identifica a la caja.

```
PPConnector.getOrder("ABC123", new PPCallback() {
    @Override
    public void onFinished(Response res) {
          //orden disponible en la caja
    }
});
```

Código	Response	
201	code	201
	status	true
	message	Ultima orden pendiente de la caja con codigo [codigo de la caja] se obtuvo correctamente.
	data	{ "OrdenId":29, "IdTransaccionInterno":"ORDEN001", "UrlNotificacion": null, "MontoTotal": 12.40 , "Productos": "Producto1, Producto2", "CajaId": "29", "FechaCreacion":"2019-09-18T22:36:40.2640222-03:00", "FechaExpiracion":"2019-09-18T22:50:40.2640222-03:00", "NombreCaja": "ABC123", "Estado": "PENDIENTE" }
400	code	404
	status	false
	message	No se encontró ninguna orden vigente con el codigo ingresado.
	data	-

ESTADO DE UNA ORDEN		
1	PENDIENTE	Orden pendiente de pago
2	APROBADA	Orden pagada con éxito
3	RECHAZADA	Orden con pago rechazado
4	EXPIRADA	Orden vencida o expirada

• Consultar por ID de caja

El comercio tiene la posibilidad de consultar por la orden que se encuentre vigente en una caja. Para ello deberá invocar al método **PPConnector.getOrderByCajaId(int cajaId)** y enviar como parámetro el identificador de la caja, devuelto al momento de crear la caja en el campo cajaId.

```
PPConnector.getOrderByCajaId(29, new PPCallback() {
    @Override
    public void onFinished(Response res) {
          //orden disponible en la caja
    }
});
```

Código	Response	
200	code	200
	status	true
	message	Ultima orden pendiente de la caja con ID [id de la caja] se obtuvo correctamente.
	data	{ "CajaId":29, "GUID":"1f255edd-d354-490c-9a2b-5cf54b686bff", "Codigo":"ABC123", "Nombre":"Caja de ejemplo", "Fixed_Amount":true, "Habilitado":true, "TSCreate":"2019-09-18T22:36:40.2640222-03:00", "TSModificado":null, "TSEliminado":null, "SucursalComercioId":3016 }
400	code	404
	status	false
	message	No se encontró ninguna orden vigente con el codigo ingresado.
	data	-

Eliminar una orden

El comercio podrá eliminar una orden de una caja, invocando al método **PPConnector.deleteOrder(String codigo)** y enviando como parámetro el identificador de la caja asociada a la orden.

```
PPConnector.deleteOrder("ABC123", new PPCallback() {
    @Override
    public void onFinished(Response res) {
       //resultado de la eliminación de la orden de la caja
    }
});
```

Response

Código	Response	
200	code	200
	status	true
	message	Orden eliminada correctamente.
	data	-
400	code	400
	status	false
	message	No se encontró una orden para eliminar en la caja con código [codigo_caja].
	data	-

Referencia 1: Errores de generación de token de pago

Mensaje	Campo involucrado
El monto del pago no puede ser nulo o vacío.	Monto
El identificador de comercio no puede ser nulo o vacío	Comercio
El identificador de la transacción en el comercio no puede ser nulo o vacío.	TransaccionComercioId
La dirección IP del comprador no puede ser nula o vacía.	IP
El monto del pago no puede ser negativo.	Monto
El identificador de comercio no es válido.	Comercio

Referencia 2: Errores de pago

Mensaje	Campo involucrado
Número de cuota no disponible para el medio de pago seleccionado.	CantidadCuotas
Número de cuotas inválido.	CantidadCuotas
No se han aceptado los términos y condiciones.	AceptTerminosyCondiciones
Formato en Email del comprador incorrecto.	Email
Formato de la fecha de nacimiento del titular incorrecto.	FechaNacimientoTitular
El token de pago ingresado ha expirado.	Token de Pago
No existe el token de pago ingresado.	Token de Pago
El medio de pago ingresado no está habilitado para el comercio.	MedioPagoId

Mensaje	Descripción
No se encontró la transacción correspondiente a ese token de pago.	El token de pago no corresponde a una transacción.
Error al realizar el pago.	Error genérico de un pago no exitoso.
Ocurrió un error al realizar el pago. Los datos ingresados son inválidos.	Error en los datos de la tarjeta ingresada.
El pago no se pudo realizar. Motivo: [MOTIVO]	Especificación del motivo por el cual no se realizó el pago con ese medio de pago.
El pago para esta transacción ya ha sido realizado.	Intento de pagar una transacción cuyo pago ya ha sido realizado.
La dirección IP que solicitó el pago no coincide con la dirección IP que está realizando el pago.	Conflicto con las direcciones IP que se enviaron en la petición del token de pago y en la realización del pago.

Referencia 3: Cálculo de dirección IP del comprador

• Net Framework

```
string ip = System.Web.HttpContext.Current.Request.ServerVariables["HTTP_X_FORWARDED_FOR"];
if (string.IsNullOrEmpty(ip))
{
    ip = System.Web.HttpContext.Current.Request.ServerVariables["REMOTE_ADDR"];
}
```

• Net Core

• Inyectar la dependencia IHttpContextAccessor en el Controller dónde se va a instanciar la librería.

```
public class HomeController: Controller
{
    public IHttpContextAccessor _accessor;
    public HomeController (IHttpContextAccessor httpAccessor)
       {
             _accessor = httpAccessor;
    }
}
```

• Obtener la dirección IP

```
var ip = _accessor.HttpContext.Connection.RemoteIpAddress.ToString();
```

• JAVA

• Obtener la dirección IP:

Normalmente, mediante el parámetro request al método se debería poder obtener con lo siguiente:

```
import javax.servlet.http.HttpServletRequest;

String ipAddress = request.getRemoteAddr();
```

Si por algún motive, el cliente se conecta mediante proxy o cloudflare, la otra solución es obtener la ip remota mediante el HTTP request header **X-Forwarded-For (XFF)** :

```
import javax.servlet.http.HttpServletRequest;

    //...

    private static String getClientIp(HttpServletRequest request) {

        String remoteAddr = " ";

        if (request != null) {
            remoteAddr = request.getHeader("X-FORWARDED-FOR");
            if (remoteAddr == null || " ".equals(remoteAddr)) {
                remoteAddr = request.getRemoteAddr();
            }
        }

        return remoteAddr;
    }
```

• PHP

• Descarga del paquete:

```
composer require sdkconnector/ppconectorsdk:dev-master
```

Instanciar el connector para poder utilizar todos los métodos

```
<?php
include_once dirname(__FILE__)."/../vendor/autoload.php";
use PPConnectorSDK\Models\AmbienteEnum;

$connector = new \PPConnectorSDK\Connector(AmbienteEnum::Sandbox);

?>
```

1. HealthCheck

```
<?php
$response = $connector->HealthCheck ();
var_dump($response);
?>
```

a. Response

Código	Response	
200	code	200
	status	true
	message	Servicio disponible.
	data	Versión 1.18.0
400	code	400
	status	false
	message	Servicio no disponible.
	data	-

2. Token de autenticación

AuthenticationModel			
Campo	Descripción	Tipo	Requerido
frase	Frase generada por Macro Click de Pago al registrar el comercio	string	SI
guid	Identificador del comercio generado por Macro Click de Pago al registrar el comercio	string	SI

```
$model = new StdClass();
$model->{'guid'} = "d1050b9c-4805-4ea3-8f66-6a601490010a";
$model->{'frase'} = "JyPQ8nFbazpTF1aCZLCVexboOmVxeiKYE8c+PScTsQg=";

$token = $connector->GetAuthenticationToken($model);
$content = json_decode($token, true);
$tokenVal = $content["data"];
var_dump($token);
```

a. Respuesta:

Código	Response	
200	code	200
	status	true
	message	Identificación del comercio correcta.
	data	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYmYiOjE1OTcyNDMxOTAsImV4cCI6MTU5NzI0Mzc5MCwiaXNzIjoiaHR0cDovL2FzanBhZ29zYXBwMDFwOjgwODUvIiwiYXVkIjoiaHR0cHM6Ly9hc2pwYWdvc2RtejAxcDo4MDgyLyIsIk5vbWJyZSI6IlNvbHVjaW9uZXMgQW5kaW5hcyIsIkNvbWVyY2lvSWQiOjMwMTZ9._P_s7H4p-qW708gvoJ8O3VxbiNe5RMfC3Hq8oKCbKTow","secretKey":"SolucionesAndinas_9b59b264-5168-42aa-801d-fd27ca454503
401	code	401
	status	false
	message	No se pudo identificar el comercio.
	data	-

1.1 Flujo de pago

1. Modelo para obtener el token de pago

Para poder realizar una transacción de pago se deberá obtener un identificador del pago. Para ello se debe invocar el método **GetPaymentToken ()**, que recibe como parámetro un modelo **TokenModel** junto con el token obtenido anteriormente y la **SecretKey** que provee **Macro Click de Pago** al momento de registrar el comercio.

TokenModel			
Campo	Descripción	Tipo	Requerido
Comercio	Identificador del comercio generado por **Macro Click de Pago** al registrar el comercio.	string	SI
SucursalComercio	Identificador de la sucursal otorgado por **Macro Click de Pago** al registrar el comercio.	string	NO
Productos	Array de productos que se van a pagar	String [ ]	NO
TotalOperacion	Monto de la operación. Los últimos dos caracteres representan los decimales. Ej.: $ 2.00 → 200	string	SI
TransaccionComercioId	Identificador ÚNICO de la transacción en el comercio.	string	SI
IP	Dirección IP del cliente que está realizando la compra. Se deberá calcular cada vez que se haga una transacción de pago.	string	SI

Importante: El campo IP es un mecanismo de seguridad que sirve para verificar que el cliente que solicita el pago, sea el mismo que ejecuta el pago, por lo tanto, es responsabilidad del implementador realizar el cálculo de la misma en las dos etapas del pago (paso 3 y paso 5). En la Referencia 3, se indica un ejemplo de cómo obtener la dirección IP.

```
use PPConnectorSDK\Models\TokenModel;

$secretKey = “SolucionesAndinas_9b59b264-5168-42aa-801d-fd27ca454503”;

$token_model = new TokenModel();
$token_model->Comercio = “d1050b9c-4805-4ea3-8f66-6a601490010a”;
$token_model->UrlDominio = “www.google.com”;
$token_model->Productos = array(“test”, “test2”);
$token_model->TotalOperacion = “5000”;
$token_model->SucursalComercio = “0”;
$token_model->TransaccionComercioId = “Test123Test”;
$token_model->Ip = “::1”;

$token_pago = $connector->GetPaymentToken($token_model, $tokenVal, $secretKey);
var_dump($token_pago);
```

Response

Código	Response	
200	code	200
	status	true
	message	Token de pago generado correctamente.
	data	bf7f438a-90ab-41dd-a3a1-004464881f1d
400	code	400
	status	false
	message	Referencia 1
	data	-
400	code	400
	status	false
	message	La dirección IP del comprador no puede ser nula o vacía
	data	-
500	code	500
	status	false
	message	Tiempo de espera del servidor excedido. SQL Timeout
	data	-

2. Obtención de medios de pago (OPCIONAL)

El comercio podrá obtener los medios de pago disponibles para operar en **Macro Click de Pago** y deberá pasar el código del mismo a la hora de ejecutar el pago. Para ello deberá realizar una llamada al método **GetPaymentMethods()** pasando como parámetro el identificador de acceso obtenido en el paso 2.

```
<?php
$mediosPago = $connector->GetPaymentMethods($tokenVal);
var_dump($mediosPago);
?>
```

Response

Código	Response	
200	code	200
	status	true
	message	Medios de pago listados correctamente.
	data	[{"MedioPagoId":1023,"Nombre":"Visa Credito"," Descripcion"}, {"MedioPagoId":1024,"Nombre":"Maestro","Descripcion":""}, {"MedioPagoId"1027, "Nombre":"Visa Débito","Descripcion":""}]
400	code	400
	status	false
	message	Error al listar los medios de pago.
	data	-

3. Ejecución de un pago

Para ejecutar el pago de una transacción el comercio deberá hacer uso de el token de pago generado en el paso 3 e invocar al método **ExecutePayment()** que recibe como parámetro un modelo del tipo **PaymentModel**, el identificador de acceso y la **SecretKey** provista por **Macro Click de Pago** al registrar el comercio.

PaymentModel			
Campo	Descripción	Tipo	Requerido
DatosTarjeta	Objeto del tipo DatosTarjeta	DatosTarjeta	SI
AceptaHabeasData	Control de la aceptación de Habeas Data	bool	SI
AceptTerminosyCondiciones	Control de la aceptación de Términos y condiciones.	bool	SI
CantidadCuotas	Numero de cuotas en las que se va a pagar la compra.	int	SI
Hash	-	-	NO
IPCliente	IP del cliente que está realizando la compra	string	SI
MedioPagoId	Identificador del medio de pago.		

DatosTarjeta			
Campo	Descripción	Tipo	Requerido
AñoVencimiento	Año en que caduca la tarjeta	string	SI
MesVencimiento	Mes en que caduca la tarjeta	string	SI
CodigoTarjeta	CVV – Código de seguridad de la tarjeta	string	SI
DocumentoTitular	Número de documento del titular de la tarjeta	string	SI
Email	Email del usuario que realiza la compra. A este email se enviará la confirmación del pago.	string	SI
FechaNacimientoTitular	Fecha de nacimiento del titular de la tarjeta. En forma DDMMAAAA	string	SI
NumeroPuertaResumen	N.º del domicilio dónde recibe el resumen de la tarjeta el titular de la misma.	string	NO
NumeroTarjeta	Número de la tarjeta	string	SI
TipoDocumento	Tipo de Documento del Titular de la tarjeta (DNI)	string	SI
TitularTarjeta	Nombre y apellido del titular de la tarjeta tal cual sale en la tarjeta.	string	SI

Importante: El campo IP es un mecanismo de seguridad que sirve para verificar que el cliente que solicita el pago, sea el mismo que ejecuta el pago, por lo tanto, es responsabilidad del implementador realizar el cálculo de la misma en las dos etapas del pago (paso 3 y paso 5). En la Referencia 3, se indica un ejemplo de cómo obtener la dirección IP.

```
use PPConnectorSDK\Models\DatosTarjeta;
use PPConnectorSDK\Models\PaymentModel;

$secretKey = “SolucionesAndinas_9b59b264-5168-42aa-801d-fd27ca454503”;

$tokenTransaccion = json_decode($token_pago, true);

$datosTarjeta = new DatosTarjeta();
$datosTarjeta->AñoVencimiento = “37”;
$datosTarjeta->MesVencimiento = “12”;
$datosTarjeta->CodigoTarjeta = “337”;
$datosTarjeta->DocumentoTitular = “34004923”;
$datosTarjeta->Email = “test@test.com”;
$datosTarjeta->FechaNacimientoTitular = “21021995”;
$datosTarjeta->NumeroPuertaResumen = “20”;
$datosTarjeta->NumeroTarjeta = “778899000000519004”;
$datosTarjeta->TipoDocumento = “DNI”;
$datosTarjeta->TitularTarjeta = “APELLIDO493/NOMBRE493”;

$payment = new PaymentModel();
$payment->DatosTarjeta = $datosTarjeta;
$payment->AceptaHabeasData = “True”;
$payment->AceptTerminosyCondiciones = “True”;
$payment->CantidadCuotas = “1”;
$payment->IPCliente = “::1”;
$payment->MedioPagoId = “7”;

$payment = $connector->ExecutePayment($payment, $tokenTransaccion[“data”], $tokenVal,
$secretKey);


var_dump($payment);
```

Response

Código	Response	
200	code	200
	status	true
	message	Pago realizado correctamente.
	data	{\"TransaccionComercioId\":\"028028\",\"TransaccionPlataformaId\":\"24336\",\"Tipo\": \"PAGO\",\"Monto\":\"20,00\",\"Estado\":\"REALIZADA\",\"Detalle\":\"Pago realizado exitosamente.\",\"MetodoPago\":\"2\",\"MedioPago\":\"8\",\"EstadoId\":\"3\", \"Cuotas\":\"1\",\"InformacionPagador\":{\"Email\":\"test@gmail.com\", \"Nombre\":\"test\",\"NumeroDocumento\":\"25123456\",\"Telefono\":null,\"TipoDocumento\": \"DNI\"},\"Resultado\":{\"CodigoInterno\":\"00\",\"Descripcion\":\"Pago correcto\"}}
400	code	400
	status	false
	message	Referencia 2
	data	{\"TransaccionComercioId\":\"058704\",\"TransaccionPlataformaId\":\"24337\",\"Tipo\": \"PAGO\",\"Monto\":\"20,00\",\"Estado\":\"RECHAZADA\",\"Detalle\":\"El pago fue rechazado.\",\"MetodoPago\":\"2\",\"MedioPago\":\"8\",\"EstadoId\":\"4\",\"Cuotas\":\"1\", \"InformacionPagador\":{\"Email\":\"test@gmail.com\",\"Nombre\":\"Test\",\"NumeroDocumento\":\"25123456\",\"Telefono\":null,\"TipoDocumento\":\"DNI\"}, \"Resultado\":{\"CodigoInterno\":\"54\",\"Descripcion\":\"Tarjeta expirada\"}}
500	code	500
	status	false
	message	Tiempo de espera del servidor excedido. SQL Timeout
	data	-

1.2 Flujo de una consulta

1. Obtener el listado de transacciones del comercio

El comercio podrá obtener un listado de las transacciones y el estado en el que se encuentran, invocando al método GetTransactions () que recibe como parámetro el identificador de acceso.
Filtros:

- TransaccionId – string
- Fecha desde – string (dd/MM/yyyy)
- Fecha Hasta – string (dd/MM/yyyy)
- EstadoTransaccion – string
- TransaccionComercioId – string
- NumeroSucursal – string
- Caja – string (código)
- Pagina – int
- Cantidad – int

Importante: los filtros por Fecha y TransaccionComercioId son obligatorios.

```
<?php
$transaccionId = “”;
$fechaDesde = “01/07/2020”;
$fechaHasta = “”;
$estadoTransaccion = “”;
$transaccionComercioId = “697648”;
$numeroSucursal = “”;
$caja = “”;
$pagina = “”;
$cantidad = “”;
$listTransacciones = $connector->GetTransactions($tokenVal, $transaccionId, $fechaDesde,
$fechaHasta, $estadoTransaccion, $transaccionComercioId, $numeroSucursal, $caja, $pagina,
$cantidad);
var_dump($listTransacciones);
?>
```

Response

Código	Response	
200	code	200
	status	true
	message	Transacciones listadas correctamente.
	data	“{\”Transacciones\”: [{“TransaccionId”:7638,”Estado”:”RECHAZADA”,”Fecha”:”2018-09-19T13:14:46.527278”,”Moneda”:”ARS”,”Monto”:”2510,51”,”Productos”:”Producto 1, Producto 2”,”Sucursal”: null,”MedioPagoId”:”1024”,”MedioPago”:”Maestro”, “TransaccionComercioId”:”431787”}, {“TransaccionId”:7639,”Estado”: “REALIZADA”,”Fecha”:”2018-09-19T13:17:41.417821”,”Moneda”:”ARS”,”Monto”:”234,44”,”Productos”: “test”, “Sucursal”: null,”MedioPagoId”:”1023”,”MedioPago”:”Visa Credito”,”TransaccionComercioId”:”34343443”,”Result”:{“Codigo”:”00”,”Descripcion”:”Pago correcto”}, {“TransaccionId”:7642, “Estado”:”EXPIRADA”,”Fecha”:”2018-09-26T15:27:28.4327796”,”Moneda”:”ARS”,”Monto”:”200,00”, “Productos”: aaa, bbb”,”Sucursal”: null, “MedioPagoId”: null, “MedioPago”: null, “TransaccionComercioId”:”613419”,”Result”:{“Codigo”:”00”,”Descripcion”:”Pago correcto”}], \”Pagina\”:3,\”Cantidad\”:2,\”TotalRows\”:1068}”
400	code	400
	status	false
	message	Error al listar las transacciones.
	data	-
400	code	400
	status	false
	message	“Debe ingresar al menos el dato TransaccionComercioId o FechaDesde/Hasta”
	data	-

Estados Transacción		
Campo	Valor	Descripción
1	CREADA	Transacción creada (no procesada)
2	EN_PAGO	Transacción en pago
3	REALIZADA	Transacción procesada con éxito
4	RECHAZADA	Transacción rechazada
5	ERROR_VALIDACION_HASH_TOKEN	Ocurrió un error en la validación de la firma enviada al crear la transacción
6	ERROR_VALIDACION_HASH_PAGO	Ocurrió un error en la validación de la firma al intentar pagar la transacción
7	EXPIRADA	Transacción expirada
8	CANCELADA	Transacción cancelada por el usuario
9	DEVUELTA	Transacción devuelta
10	PENDIENTE	Debin creado correctamente, esperando aprobación.
11	VENCIDA	Transacción vencida porque el DEBIN expiró.

2. Obtener el detalle de una transacción en particular

El comercio podrá obtener el detalle de una transacción en particular invocando al método **GetTransactionByTxComercioId ()** que recibe como parámetro el identificador de acceso y el identificador de la transacción en el comercio.

```
<?php
$transaccionComercioId = “697648”;
$transaccion = $connector->GetTransactionByTxComercioId($tokenVal, $transaccionComercioId);
var_dump($transaccion);
?>
```

Response

Código	Response	
200	code	200
	status	true
	message	Transaccion identificada como <697648> listada correctamente.
	data	{"TransaccionId":7759,"Estado":"REALIZADA","Fecha":"2018-10-25T10:59:08.7145689","Moneda":"ARS","Monto":"20,00","Productos":"Producto1, Producto2, ","Sucursal": null,"MedioPagoId":"1023","MedioPago":"Visa Credito","-TransaccionComercioId":"697648","Result":{"Codigo":"00","Descripcion":"Pago correcto"}
400	code	400
	status	false
	message	Error al listar la transacción solicitada.
	data	-

1.3. Órdenes de pago

1. Dar de alta una caja

El comercio podrá dar de alta cajas para sus sucursales, invocando al método **Caja()** que recibe como parámetro el identificador de acceso, un objeto del tipo **CajaModel** y la **secretkey** provista por **Macro Click de Pago** al registrar el comercio.

CajaModel			
Campo	Descripción	Tipo	Requerido
Nombre	Nombre de la caja	string	SI
Codigo	Identificador alfanumérico de la caja	string	SI
SucursalComercioId	Identificador de la sucursal en el que se creará la caja	int	SI
Fixed_amount	Enviar el valor true si la caja opera con ordenes de pago o false en caso contrario.	Bool	SI

```
<?php
use PPConnectorSDK\Models\CajaModel;
$secretKey = "SolucionesAndinas_9b59b264-5168-42aa-801d-fd27ca454503";

$caja = new CajaModel();
$caja->Nombre = "Test";
$caja->Codigo = "123456TEST";
$caja->SucursalComercioId = "3014";
$caja->Fixed_Amount = "False";
$responseCaja = $connector->Caja($caja, $tokenVal, $secretKey);
var_dump($responseCaja);
?>
```

Response

Código	Response	
201	code	201
	status	true
	message	[Nombre caja] creada correctamente.

	data	{
		"CajaId":12951,
		"GUID":"564cc130-6716-4b7f-840a-e03e7a0f51e7",
		"Codigo":"123456TEST",
		"Nombre":"Test",
		"Fixed_Amount":false,
		"Habilitado":true,
		"TSCreate":"2020-08-12T15:17:40.2640222-03:00",
		"TSModificado":null,
		"TSEliminado":null,
		"SucursalComercioId":3014
		}

400	code	400
	status	false
	message	Ocurrió un error al crear la caja para la Sucursal.
	data	null
403	code	403
	status	false
	message	El servidor rechazó la conexión con el servicio.
	data	-

2. Obtener Detalle de una caja

a. Por código de caja

El comercio podrá consultar el detalle de una caja creada con anterioridad invocando al método **GetDetailCajaByCodigo( )** que recibe como parámetros el identificador de acceso y el código de la caja que se quiere consultar.

```
<?php
$cajaDetalle = $connector->GetDetailCajaByCodigo($tokenVal, "123456TEST");
var_dump($cajaDetalle);

?>
```

Response

Código	Response	
200	code	200
	status	true
	message	Caja con codigo <codigo_caja> obtenida correctamente.
	data	{ "CajaId":12951, "GUID":"564cc130-6716-4b7f-840a-e03e7a0f51e7", "Codigo":"123456TEST", "Nombre":"Test", "Fixed_Amount":False, "SucursalComercioId":2014 }
400	code	404
	status	false
	message	No se encontró caja con código <codigo_caja>.
	data	-

b. Por identificador de caja

El comercio podrá consultar el detalle de una caja creada con anterioridad invocando al método **GetDetailCajaById( )** que recibe como parámetros el identificador de acceso y el identificar de la caja en Macro Click de Pago, que se quiere consultar.

```
<?php
$cajaDetalle = $connector->GetDetailCajaById($tokenVal, "12951");
var_dump($cajaDetalle);
?>
```

Código	Response	
200	code	200
	status	true
	message	Caja con ID <identificador_caja> obtenida correctamente.
	data	{ "CajaId":12951, "GUID":"564cc130-6716-4b7f-840a-e03e7a0f51e7", "Codigo":"123456TEST", "Nombre":"Test", "Fixed_Amount":false, "SucursalComercioId":2014 }
400	code	404
	status	false
	message	No se encontró caja con ID <identificador_caja>.
	data	-

3. Generar una orden de pago

Si el comercio desea cobrar con QR, deberá generar una orden de pago para una caja. Para ello, deberá invocar al método **Order()** y enviar como parámetro un objeto del tipo **OrderModel**, el identificador de acceso, la **secretkey** provista por **Macro Click de Pago** al registrar el comercio, el identificador de la caja en la que se creará la orden y opcionalmente, un tiempo de expiración para esa orden.

OrdenModel			
Campo	Descripción	Tipo	Requerido
MontoTotal	Monto por pagar. Los últimos dos posiciones representan los decimales. Ej.: 12,00 → "1200"	string	SI
IdTransaccionInterno	Identificador alfanumérico de la orden	string	SI
Productos	Descripción de los productos a pagar	int	SI
UrlNotificacion	URL a la que se notifica, en caso de dejar vacío si notifica a la configurada al dar de alta el comercio en **Macro Click de Pago**	Bool	SI

La orden expira a un tiempo definido por **Macro Click de Pago** al ser creada. Si se desea expirarla a un tiempo diferente, se deberá enviar al invocar el método de creación de orden, en el parámetro **ttlPreference** en segundos.

Creación de orden **sin establecer ttlPreference**, expirando según tiempo máximo default de **Macro Click de Pago**:

```
<?php
use PPConnectorSDK\Models\OrderModel;

$order = new OrderModel();
$order->MontoTotal = “1000”;
$order->IdTransaccionInterno = “ORDEN001”;
$order->Productos = “Producto1,Producto2”;

$responseOrder = $connector->Order($order, “123456TEST”, $tokenVal, $secretKey);
var_dump($responseOrder);
?>
```

Creación de orden **estableciendo ttlPreference**:

```
<?php
use PPConnectorSDK\Models\OrderModel;

$order = new OrderModel();
$order->MontoTotal = “1000”;
$order->IdTransaccionInterno = “ORDEN001”;
$order->Productos = “Producto1,Producto2”;

$responseOrder = $connector->Order($order, “123456TEST”, $tokenVal, $secretKey, 120);
var_dump($responseOrder);
?>
```

El **ttlPreference** debe ser un valor entero expresado en segundos. En este ejemplo, 120 segundos de expiración para la orden creada

Código	Response	
200	code	201
	status	true
	message	La orden de pago se creó correctamente.

	data	{
		“ordenId”: 235,
		“IdTransaccionInterno”: “ORDEN001”,
		“UrlNotificacion”:http://www.google.com,
		“MontoTotal”:”10.00,
		“Productos”: Producto1,Producto2,
		“CajaId”:12951,
		“FechaCreacion”:2020-08-13T08:54:09,
		“FechaExpiracion”:2020-08-13T08:54:09,
		“Nombre”:”Test”,
		“Estado”:PENDIENTE,
		“QrCaja”:00020102082511661743090005129515015001130710126220520497005802AR5918Soluciones Andinas6010SAN RAFAEL-610456005302326229031 1nicolas.hhm0710000000000801300 09com.gmail8110000623232363043 4EA
		}

400	code	400
	status	false
	message	No se encontró una caja con el código [codigo_caja].
	data	-
400	code	400
	status	false
	message	La URL de notificación no tiene un formato válido.
	data	-
400	code	400
	status	false
	message	IdTransaccionInterno no puede ser nulo.
	data	-
400	code	400
	status	false
	message	TtlPreference no tiene un formato válido.
	data	-
400	code	400
	status	false
	message	Ocurrió un error al crear la orden de pago no se pudo crear.
	data	-

4. Consultar por una orden

El comercio tiene la posibilidad de consultar el detalle de una orden que haya creado con anterioridad. Para ello deberá invocar al método **GetOrderById()** y enviar como parámetro el identificador de acceso **(access_token.data)** y el identificador de la orden en **Macro Click de Pago**. Este identificador se obtiene al crear una orden, en el campo **OrdenId**.

```
<?php
$orden = $connector->GetOrderById($tokenVal, "235");
var_dump($orden);
?>
```

Código	Response	
200	code	200
	status	true
	message	Orden de pago con ID <235> obtenida correctamente.

	data	{
		"OrdenId":235,
		"IdTransaccionInterno":"ORDEN001",
		"UrlNotificacion":http://www.google.com,
		"MontoTotal":10.00,
		"Productos":"Producto1, Producto2",
		"CajaId":12951,
		"FechaCreacion":"2020-08-13T12:59:02.6081937",
		"FechaExpiracion":"2020-08-13T12:59:02.6041596",
		"NombreCaja":"Test",
		"Estado":"PENDIENTE"
		}

400	code	404
	status	false
	message	No se encontró una orden con ID <235>.
	data	-

5. Consultar orden por código de caja

El comercio tiene la posibilidad de consultar por la orden que se encuentre vigente en una caja. Para ello deberá invocar al método **GetOrder()** y enviar como parámetro el identificador de acceso **(access_token.data)** y el código que identifica a la caja.

```
<?php
$orden = $connector->GetOrder($tokenVal, "123456TEST");
var_dump($orden);
?>
```

Response

Código	Response	
200	code	200
	status	true
	message	La orden de pago correspondiente a la Caja con Código <123456TEST> se obtuvo correctamente.

	data	{
		"OrdenId":237,
		"IdTransaccionInterno":"ORDEN001",
		"UrlNotificacion": http://www.google.com,
		"MontoTotal": 10.00 ,
		"Productos": "Producto1, Producto2",
		"CajaId": "235",
		"FechaCreacion":"2020-08-13T22:36:40.2640222-03:00",
		"FechaExpiracion":"2020-08-13T22:50:40.2640222-03:00",
		"NombreCaja": "Test",
		"Estado": "PENDIENTE"
		}

400	code	404
	status	false
	message	No se encontró ninguna orden de pago vigente con el codigo ingresado.
	data	-

Estado de una Orden		
1	PENDIENTE	Orden pendiente de pago
2	APROBADA	Orden pagada con éxito
3	RECHAZADA	Orden con pago rechazado
4	EXPIRADA	Orden vencida o expirada

6. Consultar orden por ID de caja

El comercio tiene la posibilidad de consultar por la orden que se encuentre vigente en una caja. Para ello deberá invocar al método **GetOrderByCajaId()** y enviar como parámetro el identificador de acceso **(access_token.data)** y el identificador de la caja, devuelto al momento de crear la caja en el campo **CajaId.**

```
<?php
$orden = $connector->GetOrderByCajaId($tokenVal, “12951”);
var_dump($orden);
?>
```

Response

Código	Response	
200	code	200
	status	true
	message	Ultima orden de pago pendiente de la caja con ID [id de la caja] se obtuvo correctamente.
	data	{ “OrdenId”: 240, “IdTransaccionInterno”: ORDEN001, “UrlNotificacion”: http://www.google.com, “MontoTotal”: 10.00, “Productos”: Producto1, Producto2, “CajaId”:12951, “FechaCreacion”: 2020-08-14T11:08:26, “FechaExpiracion”: 2020-08-14T11:10:26, “NombreCaja”:”Test”, “Estado”: Pendiente }
400	code	404
	status	false
	message	No se encontró ninguna orden de pago reciente de la Caja con ID <12951>.
	data	-

7. Eliminar una Orden

El comercio podrá eliminar una orden de una caja, invocando al método **DeleteOrder()** y enviando como parámetros el identificador de acceso, junto con el identificador de la caja.

```
<?php
$orden = $connector->DeleteOrder($tokenVal, “123456TEST”);
var_dump($orden);
?>
```

Código	Response	
200	code	200
	status	true
	message	Orden eliminada correctamente.
	data	-
400	code	404
	status	false
	message	No se encontró una orden para eliminar en la caja con código [codigo_caja]
	data	-

8. Dar de baja una caja

a. Por código de caja

El comercio podrá eliminar una caja, invocando al método **DeleteCajaByCodigo()** que recibe como parámetro el identificador de acceso y código asignado para identificar a la caja

```
<?php
$caja = $connector-> DeleteCajaByCodigo ($tokenVal, “123456TEST”);
var_dump($orden);
?>
```

Código	Response	
200	code	200
	status	true
	message	Orden eliminada correctamente.
	data	-
400	code	400
	status	false
	message	Ocurrió un error al eliminar la caja.
	data	null
403	code	403
	status	false
	message	El servidor rechazó la conexión con el servicio.
	data	-
404	code	404
	status	false
	message	No se encontró ninguna Caja con el código <código_caja>.
	data	-

b. Por identificación de caja

El comercio podrá eliminar una caja invocando al método **DeleteCajaById()** que recibe como parámetro el identificador de acceso y el identificador interno de la caja en **Macro Click de Pago**. Esta información se devuelve al consultar por una caja en el campo **CajaId.**

```
<?php
$caja = $connector-> DeleteCajaById($tokenVal, “12951”);
var_dump($orden);
?>
```

Código	Response	
200	code	200
	status	true
	message	Caja con ID <id_caja> se eliminó correctamente.
	data	-
400	code	400
	status	false
	message	Ocurrió un error al eliminar la caja.
	data	null
403	code	403
	status	false
	message	El servidor rechazó la conexión con el servicio.
	data	-
404	code	404
	status	false
	message	No se encontró ninguna Caja con el ID <id_caja>.
	data	-