SDK-PaGo

Manual de Integración SDK PaGo V2.5.2

V1

Contenido

Historial de revisiones....3
Servicios de IA PaGo....4
Descripción Detallada de Servicios de API....7
1. Validación....7
Petición....7
Respuesta....7
2. Registro....8
Petición....8
Respuesta....10
3. Pago (compra y reverso) – Venta No Presente....11
Petición Autorización....11
Petición reversa....12
Respuesta para autorización y reverso....13
4. Microdébito....14
Petición....14
Respuesta....15
5. Autenticación....15
Petición....15
Respuesta....16
6. Cierre Sesión....17
Petición....17
Respuesta....17
7. Listado Tarjetas....17
Petición....17
Respuesta....18
8. Agregar Tarjeta....19
Petición....19
Respuesta....20

9. Eliminar Tarjeta....21
Petición....21
Respuesta....21
Anexos....23
ANEXO 1: API RESTful de RedHat SSO....23
ANEXO 2: Códigos de Error - Vpayment....31
ANEXO 3: Términos y condiciones de PaGo....37

Historial de revisiones

Fecha	Versión	Descripción	Autor
11/11/2019	1.0	Generación del documento	CD
21/04/2020	2.0	Inclusión de especificación SDK	LG
30/03/2020	2.1	Actualización de URL de pruebas	CB
15/05/2020	2.2	Cambio de nombre del producto de IA PaGo a SDK PaGo.	CB
01/06/2020	2.3	Inclusión historial de revisiones	CB
12/06/2020	2.4	Inclusión de nuevos plugin para el SDK: Nativos, IONIC, Cordova.	CB
18/06/2020	2.5	Se ajusta el cuadro de los plugin. Se adiciona anexo 3: términos y condiciones de PaGo	CB
22/10/2020	2.5.1	Se actualizan los códigos de respuesta de cada unos de los servicios de IA PaGo. Se adiciona especificación para la integración de SDK WEB.	CB
30/10/2020	2.5.2	Se adiciona campo "accountype" en el servicio de listado de tarjeta. Se adiciona códigos HTTPStatus para cada servicio.	CB

CD: Camilo Díaz

CB: Cesar Benavides

LG: Linda Gonzalez

Servicios de IA PaGo

El IA PaGo es un método de integración al servicio de pagos PaGo, que le permite al establecimiento de comercio recibir de manera confiable los pagos de sus clientes. El servicio está habilitado para aplicaciones móviles y le permitirá mantener su look and feel.

Este producto comprende los siguientes servicios:

ID	Nombre	Descripción
1	Validación	Permite validar si un usuario está registrado en PaGo
2	Registro/SDK	Permite el registro de un usuario en PaGo/SDK
3	Pago	Permite realizar autorizaciones / reversos de pago de venta no presente.
4	Microdebito	Permite determinar si la tarjeta que se intenta registrar está activa y apta para recibir transacciones de venta no presente.
5	Autenticación	Permite la habilitación de los usuarios en la app del comercio
6	Cierre sesión	Permite al usuario eliminar la habilitación de su usuario en la app del comercio.
7	Listado Tarjetas	Permite listar todas las tarjetas registradas en PaGo de un usuario.
8	Agregar Tarjeta/SDK	Permite registrar una nueva tarjeta a los usuarios ya registrados en PaGo y autenticados en la app del comercio/SDK
9	Eliminar Tarjeta	Permite eliminar tarjetas a los usuarios ya registrados en PaGo y autenticados en la app del comercio.

Dichos servicios se encuentran detallados en el contrato swagger disponible en las URLs:

Pruebas:

https://testsrv.credibanco.com/PaGoIntegration/docs/#/

Producción:

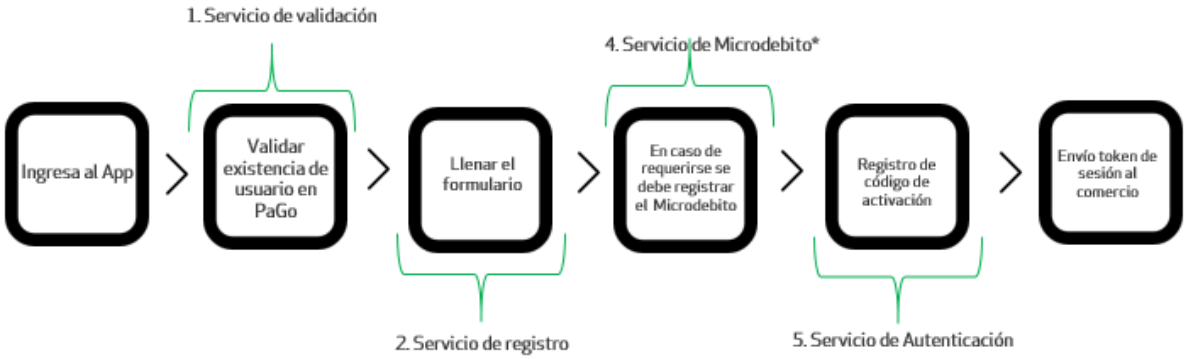
https://pago.credibanco.com/PaGoIntegration/ia/services/{urlServicio}

Donde {urlServicio} será el resto de la URL que corresponde a cada uno de los servicios que se muestran en esta tabla:

ID	Nombre	URL
1	Validación	/validateUserByDocument/{documentNumber}
2	Registro	/createUser
3	Pago	/authorize /reverse
4	Microdébito	/validateMicrodebit
5	Autenticación	/authenticateUser
6	Cierre sesión	/logout/{email}
7	Listado Tarjetas	/userCards
8	Agregar Tarjeta	/addCard
9	Eliminar Tarjeta	/deleteCard

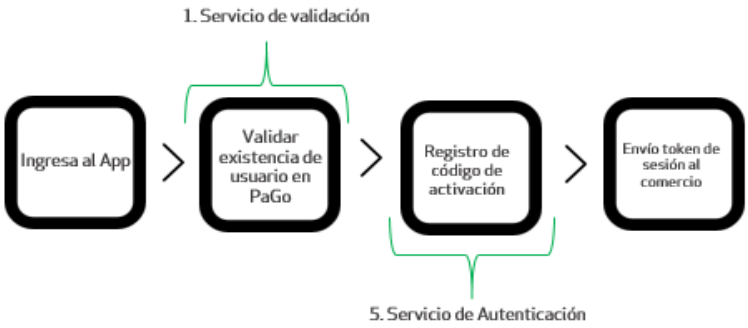
A continuación, se presenta la manera secuencial como se deben integrar los servicios para cada uno de proceso de maneja PaGo:

1. Proceso de Registro con SDK

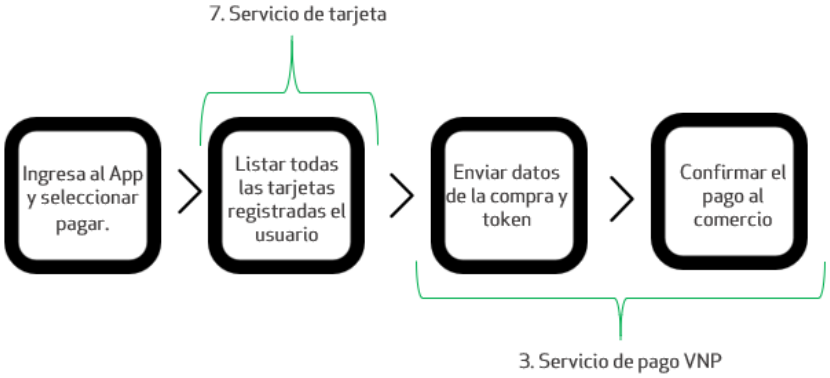


Nota: El servicio de microdebito se utilizará solo cuando el sistema lo requiera

2. Proceso de autenticación:



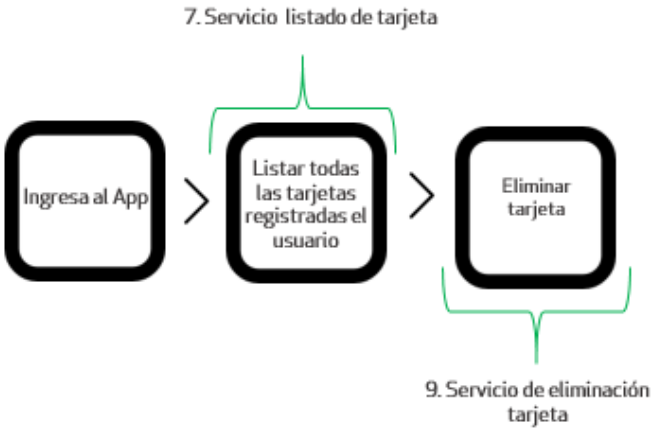
3. Proceso de pago



4. Proceso de adición de nueva tarjeta con SDK

Nota: El servicio de microdebito se utilizará solo cuando el sistema lo requiera

5. Proceso de eliminación de tarjetas



6. Proceso cierre de sesión:

Descripción Detallada de Servicios de API

A continuación, se detallarán cada uno de los servicios que componen a lA PaGo.

1. Validación

El servicio de validación proporcionará un mecanismo para que el comercio pueda comprobar el estado de registro en PaGo de algún usuario. En caso de que el usuario exista, a dicho usuario se le proporcionará una OTP que será necesaria para la autenticación. En caso contrario se mostrará el mensaje de error.

Petición

El método HTTP es GET

Respuesta

El objeto de respuesta será de tipo JSONy está definido de esta manera:

```
{
    "statusCode": codigoDeEstado,
    "message": Mensaje,
    "email": email
}
```

Donde el valor que tenga la propiedad statusCode será el código de la respuesta, message será el mensaje asociado a la respuesta sobre la petición enviada anteriormente, y email será el correo del usuario al que se ha enviado la OTP que se debe usar en el servicio de autenticación.

De esa manera, los códigos de respuesta y los mensajes que podrán obtenerse de acuerdo al estado del registro a consultar serán:

HTTPStatus	StatusCode	Message
200	0	El usuario no ha confirmado el correo. Favor validar el código de verificación enviado al correo.
200	1	El usuario no ha registrado tarjetas. Favor continuar con el proceso de registro.
200	2	El documento existe, se puede proceder con la autenticación.
404	3	No existe registro para ese documento.
500	4	Ocurrió un error al validar el documento del usuario.
500	4	Error al obtener valores desde el token de authorización

2. Registro

Este servicio permitirá crear un nuevo usuario en PaGo. Será necesario ingresar datos de usuario y de tarjeta. El consumo de este servicio se hará por SDK

Petición

Usando una petición de tipo POST con este objeto de petición. Si se va a utilizar Postman para pruebas, se usaría un objeto raw de tipo JSON, comprobar que en Headers exista el atributo Content-Type con el valor application/json:

Otros atributos necesarios en los Headers son:

Nombre header	Valor
Authorization	Bearer <JWT generado por RedHat SSO>
SDK	ID de SDK asignado a la aplicación

El objeto de petición es el siguiente:

```
{
	"userData": {
		"email" : "",
		"tyc" : "S",
		"firstName" : "",
		"lastName" : "",
		"userType" : "1",
		"documentNumber" : "1030536230",
		"cellphone" : ""
	}
```

```
      "cardData": {
            "cvv" : "",
            "cardNumber" : "",
            "alias" : "",
            "validDate" : "",
            "cardType" : ""
      }
}
```

Los datos de "carddata" relacionados en el objeto de petición será capturados a través del componente SDK. A continuación, se relaciona la especificación de este componente para cada plugin desarrollado:

Plugin	Manual	
Nativo - App	Android Native.pdf	IOS Native.pdf
Cordova - App	Plugin Cordova.pdf	
Ionic - App	Plugin Ionic.pdf	
React - App	Plugin React.pdf	
SDK- Web	sdk web.pdf	

La definición de cada uno de los campos es:

- email: el correo del usuario a registrar.
- tyc: La aceptación de los términos y condiciones, solo toma dos valores posibles, S para Sí, N para No. Los términos y condiciones se adjuntan en el anexo 2 del presente documento para que sean publicados en la aplicación que integra el presente servicios.
- firstName: El nombre del usuario.

- secondName: El apellido del usuario.
- userType: El tipo de documento de identificación del usuario, solo toma cuatro valores posibles, 1 para Cédula de Ciudadanía, 4 para Cédula de Extranjería, 5 para Pasaporte, 6 para Otro.
- documentNumber: El número del documento de identidad del usuario a registrar.
- cellphone: Número de celular del usuario a registrar.
- cvv: Número CCV2 de la tarjeta de crédito/débito del usuario a registrar.
- cardNumber: El número de tarjeta de crédito/débito del usuario a registrar.
- alias: El alias que se le dará a la tarjeta.
- validDate: Fecha de expiración de la tarjeta.
- cardType: Tipo de la tarjeta a registrar, solo toma dos posibles valores, cd para tarjeta de crédito, db para tarjeta débito.

Respuesta

HTTPStatus	StatusCode	Message
400	5	El usuario ya existe.
400	7	Correo electrónico con formato invalido.
400	8	Campos con valores vacíos.
400	9	El tipo de documento debe ser 1 (CC), 4 (CE), 5 (Pasaporte) o 6 (Otro)
400	10	El usuario no aceptó los Términos y condiciones.
400	11	El tipo de tarjeta no es válido.
400	12	La fecha de expiración de la tarjeta no es valida.
400	13	El número de celular no es válido.
500	14	Validaciones diarias excedidas. Favor validar nuevamente el día de mañana.
500	15	Error al realizar el microdébito/reverso"
500	16	Error desconocido al realizar la validación de titularidad de tarjeta.
500	17	Error al realizar microdébito/reverso.
200	18	Se requiere la validación de microdébito
200	19	Validación de tarjeta exitosa
500	20	No existe la identificación.
500	21	No coinciden los datos.
500	22	Válido con documento no vigente.
500	23	Tipo de Documento Errado.
500	24	El número de tarjeta no corresponde a la identificación dada.
500	25	La tarjeta existe pero no es válida para transar.

500	26	El número no es de una tarjeta de crédito.
500	27	El teléfono no coincide.
500	28	El email no coincide.
500	29	Error general.
500	30	Cliente con tarjeta activa en entidad.
500	31	Cliente sin tarjeta activa en entidad.
500	32	Error desconocido en validación de datacredito
500	33	Error al guardar el registro de usuario.
200	34	Registro de usuario exitoso.
500	36	Error al realizar la transacción

El objeto de respuesta será de tipo JSON y está definido de esta manera:

```
{
    "statusCode": codigoDeEstado,
    "message": Mensaje
}
```

Donde el valor que tenga la propiedad statusCode será el código de la respuesta, y message será el mensaje asociado a la respuesta sobre la petición enviada anteriormente.

De esa manera, los códigos de respuesta y los mensajes que podrán obtenerse de acuerdo al estado del registro a consultar serán:

3. Pago (compra y Anulación) – Venta No Presente

El servicio de pago comprende dos operaciones básicas: Autorización y Anulación. De esa forma se describirá cada una de esas dos operaciones así:

1. Se mostrará la URL,
2. Se mostrará el objeto de petición,
3. Se describirá cada uno de los campos que posee el objeto de petición,
4. Se mostrará el objeto de respuesta
5. Se describirá cada uno de los campos que posee el objeto de respuesta.

Petición Autorización

Usar una petición de tipo POST con este objeto de petición. En caso de utilizar Postman para pruebas, se puede usar un objeto raw de tipo JSON. Importante comprobar que en Headers exista el atributo Content-Type con el valor application/json:

```
{
      "cardToken":
"dabc01d13af61e44bb152b57b7bd214a0a45ec316e49f2a04cd308c4dee940f0",
      "dues": 1,
      "iva": 9400,
```

```
	"purchaseCode":"421252534",
	"totalAmount":94000,
	"terminalCode":"00035746",
	"ipAddress":"127.0.0.1",
	"commerceId":2066,
	"commerceCode":"015179914"
}
```

Se describirán cada uno de los campos de la petición presentada anteriormente mostrando el nombre del campo, su tipo, y su descripción. Los campos que tengan más campos dentro de sí mismos encerrados entre corchetes {} son también objetos y sus campos también se describirán de la misma manera en como se describen los demás campos.

- cardToken: El token que identifica a la tarjeta que se usará para este pago.
- dues: Cantidad de cuotas a diferir este pago.
- iva: Monto de IVA para este pago.
- purchaseCode: El número de orden de este pago.
- totalAmount: El valor total a pagar.
- ipAddress: La dirección IP desde donde se realiza la petición de pago.
- terminalCode: El código del terminal del comercio.
- commerceId: El ID del comercio.
- commerceCode: El código único del comercio.

Petición Anulación

El servicio la anulación debe efectuarse antes de las 11pm del día de realización de la compra, para que este quede aplicado el mismo día. En caso que la anulación se haga después de las 11pm del día de realización de la compra y esta sea autorizada, esta tendrá efecto al 5 día hábil de realizada, por los proceso de canje y compensación.

Usar una petición de tipo POST con este objeto de petición. En caso de utilizar Postman para pruebas, se debe usar un objeto raw de tipo JSON. Importante comprobar que en Headers exista el atributo Content-Type con el valor application/json:

```
{
	"purchaseCode": "421252534",
	"commerceId": 2066
}
```

Se describirán cada uno de los campos de la petición presentada anteriormente mostrando el nombre del campo, su tipo, y su descripción.

- purchaseCode: Código de la compra a realizar la anulación, campo de tipo caracter.
- commerceId: ID del comercio en donde se realizará el reverso de la compra, campo de tipo entero.

Respuesta para autorización y Anulación

El objeto de respuesta será de tipo JSON y está definido de esta manera:

```
{
    "statusCode": codigoDeEstado,
    "message": Mensaje,
    "responseEntity" : {
  "authorizationCode": "866262",
  "errorCode": "00",
  "additionalMessage": "",
  "authorizedAmount": "000000400000",
  "errorMessage": "Aprobada"
   }
}
```

Donde el valor que tenga la propiedad statusCode será el código de la respuesta, y message será el mensaje asociado a la respuesta sobre la petición enviada anteriormente.

El campo responseEntity es un objeto que contendrá el objeto de respuesta de vpayment. Dicho objeto está definido de esta manera:

HTTPStatus	StatusCode	Message
200	35	Transacción exitosa
500	36	Error al realizar la transacción
404	37	No se encontró el comercio, no se puede continuar con la transacción.
401	38	El login de aplicación no está asociado al comercio.

Para que una transacción se dé como aprobada, debe cumplir los siguientes tres criterios:

1. El objeto de respuesta el campo "error code" debe estar en 00.
2. En el campo "errorMessage" deber salir "Aprobada" o "Autorizada".
3. De existir un campo "authorizationCode" el cual es un campo alfanumérico.

4. Microdébito

El servicio de microdébito permitirá completar la validación de creación de un usuario o de adición de una nueva tarjeta en caso de que en el proceso de validación con Datacrédito se haya determinado que la tarjeta usada no es de propiedad del usuario, pero tanto los datos de tarjeta como los datos de usuario son correctos. Se preguntará al usuario por el valor que le fue debitado de la tarjeta con el fin de comprobar su identidad.

Petición

Usar una petición de tipo POST con este objeto de petición. En caso de usar Postman para pruebas, se debe usar un objeto raw de tipo JSON. Importante comprobar que en Headers exista el atributo Content-Type con el valor application/json:

```
{
  "value" : "",
  "token" : ""
}
```

Se describirán cada uno de los campos de la petición presentada anteriormente mostrando el nombre del campo, su tipo, y su descripción. Los campos que tengan más campos dentro de sí mismos encerrados entre corchetes {} son también objetos y sus campos también se describirán de la misma manera en cómo se describen los demás campos.

- value: Valor que fue débitado al usuario, campo de tipo string
- token: token de validación de microdébito producto del registro, campo de tipo string

Respuesta

Donde el valor que tenga la propiedad statusCode será el código de la respuesta, message será el mensaje asociado a la respuesta sobre la petición enviada anteriormente.

De esa manera, los códigos de respuesta y los mensajes que podrán obtenerse de acuerdo al resultado de la operación serán:

HTTPStatus	StatusCode	Message
500	4	Error al obtener valores desde el token de authorización
200	39	Se validó exitosamente el microdébito. Usuario registrado.
200		Se validó exitosamente el microdébito. Tarjeta registrada.
400	40	Límite de intentos diarios excedido.
400	41	Error al realizar la validación, estado de usuario no valido.
500		Error al realizar la validación.
500		Validación fallida.

5. Autenticación

El servicio de autenticación permitirá a un usuario iniciar sesión con el fin de poder hacer uso de los servicios que requieren una autenticación previa por parte del usuario, como lo son: Pagos, Listado de Tarjetas, Adición y Eliminación de Tarjetas.

Petición

Usar una petición de tipo POST con este objeto de petición. En caso de utilizar Postman para pruebas, se debe usar un objeto raw de tipo JSON. Importante comprobar que en Headers exista el atributo Content-Type con el valor application/json:

```
{
  "email" : "",
  "otp" : ""
}
```

Se describirán cada uno de los campos de la petición presentada anteriormente mostrando el nombre del campo, su tipo, y su descripción. Los campos que tengan más campos dentro de sí mismos encerrados entre corchetes {} son también objetos y sus campos también se describirán de la misma manera en cómo se describen los demás campos.

- email: Email del usuario a autenticar, campo de tipo string
- otp: OTP recibida por el usuario en el correo de inicio de sesión, campo de tipo string

Respuesta

El objeto de respuesta será de tipo JSON y está definido de esta manera:

```
{
    "statusCode": codigoDeEstado,
    "message": "Mensaje",
    "token": "0123456789abcdef"
}
```

Donde el valor que tenga la propiedad `statusCode` será el código de la respuesta, `message` será el mensaje asociado a la respuesta sobre la petición enviada anteriormente, y `token` será un número hexadecimal que representará la sesión iniciada como resultado de la autenticación, dicho token será necesario para el consumo de los demás servicios de IA PaGo.

De esa manera, los códigos de respuesta y los mensajes que podrán obtenerse de acuerdo al resultado de la operación serán:

HTTPStatus	StatusCode	Message
500	4	Error al obtener valores desde el token de authorización
200	42	Autenticación correcta.
400	43	Error al validar el OTP
401		OTP no valido.
500		Error al validar el OTP
401	44	OTP no valido.
403	45	Número máximo de intentos excedido, favor generar nuevamente el OTP.
401	46	La fecha de vigencia del OTP se ha vencido, favor generar nuevamente el OTP.

6. Cierre Sesión

Este servicio permitirá a un usuario cerrar su sesión en el contexto de IA PaGo, automáticamente cada sesión será cerrada transcurridos 90 días después del inicio de sesión.

Petición

Usar una petición de tipo DELETE con esta cabecera de petición:

```
"sessionToken" : ""
```

En esa cabecera de petición se pondrá el token de sesión obtenido en el servicio de autenticación.

Respuesta

El objeto de respuesta será de tipo JSON y está definido de esta manera:

```
{
    "statusCode": codigoDeEstado,
    "message": "Mensaje",
}
```

Donde el valor que tenga la propiedad statusCode será el código de la respuesta, message será el mensaje asociado a la respuesta sobre la petición enviada anteriormente.

De esa manera, los códigos de respuesta y los mensajes que podrán obtenerse de acuerdo al resultado de la operación serán:

HTTPStatus	StatusCode	Message
200	47	Cierre de sesión correcto.
400	48	Error al cerrar sesión.
500		
401	49	Token de sesión no valido.

7. Listado Tarjetas

Mediante este servicio se mostrará el listado de las tarjetas que tiene asociadas el usuario a su cuenta en PaGo.

Petición

El método HTTP para este servicio es GET con esta cabecera de petición:

"sessionToken":""

En esa cabecera de petición se pondrá el token de sesión obtenido en el servicio de autenticación.

Respuesta

El objeto de respuesta será de tipo JSON y está definido de esta manera:

```
{
   "statusCode": codigoDeEstado,
   "message": Mensaje
   "responseEntity":[
         {
         "alias" : alias,
         "cardToken" : cardToken,
         "lastFour": lastFour,
         "franchise": franchise
         "accountType": Tipo de tarjeta
}
```

Donde las propiedades statusCode será el código de la respuesta, y message será el mensaje asociado a la respuesta sobre la petición enviada anteriormente.

Para el campo responseEntity se tendrá un arreglo de objetos de mínimo 1 objeto que tendrá la siguiente definición:

Campos objeto responseEntity	
Nombre	**Descripción**
alias	El alias definido por el usuario para la tarjeta

cardToken	El token que identifica a la tarjeta
lastFour	Los últimos 4 dígitos de la tarjeta
franchise	La franquicia de la tarjeta
accountType	Tipo de tarjeta (0=Crédito ; 1=Débito)

De esa manera, los códigos de respuesta y los mensajes que podrán obtenerse de acuerdo con el resultado de la operación serán:

HTTPStatus	StatusCode	Message
200	50	Consulta exitosa de listado de tarjetas.
500	51	Error al consultar el listado de tarjetas. El usuario no tiene tarjetas.
401	52	Token de sesión no valido.

8. Agregar Tarjeta

El servicio de adición de tarjeta permitirá adicionar una nueva tarjeta a un usuario, a dicha tarjeta se le realizará un microdébito y una validación usando Datacrédito. En algunos casos puede requerirse la validación del microdébito, para ello se usará el servicio de Microdébito.

El consumo de este servicio se hará por SDK. La especificación técnica del componente se encuentra relacionado la descripción del servicio de registro.

Petición

El objeto de petición será de tipo JSON y está definido de esta manera:

```
{
      "cvv" : "",
      "cardNumber" : "",
      "alias" : "",
      "validDate" : "",
      "cardType" : ""
}
```

Donde:

- cvv es el código de seguridad de la tarjeta a agregar, dato de tipo carácter.
- cardNumber es el número de tarjeta a agregar, dato de tipo carácter.
- alias es el nombre dado por el usuario a la tarjeta a agregar, dato de tipo carácter
- validDate es la fecha de expiración de la tarjeta a agregar, dato de tipo fecha con el formato: "mm/aaaa"
- cardType es el tipo de tarjeta a agregar, dato de tipo caracter,
- cardType acepta los valores cd para tarjeta de crédito y db para tarjeta débito.

Otros atributos necesarios en los Headers son:

Nombre header	Valor
Authorization	Bearer <JWT generado por RedHat SSO>
SDK	ID de SDK asignado a la aplicación
sessionToken	Token de sesión del usuario

Respuesta

El objeto de respuesta será de tipo JSON y está definido de esta manera:

```
{
    "statusCode": codigoDeEstado,
    "message": Mensaje,
    "token": token
}
```

Donde los campos statusCode será el código de la respuesta, message será el mensaje asociado a la respuesta sobre la petición enviada anteriormente, y token será el token de validación de microdébito en caso de ser necesario.

De esa manera, los códigos de respuesta y los mensajes que podrán obtenerse de acuerdo al resultado de la operación serán:

HTTPStatus	StatusCode	Message
500	4	Error al obtener valores desde el token de authorización
200	53	Se requiere la validación de microdébito.
200	54	Tarjeta almacenada exitosamente.
500	55	Error al almacenar la tarjeta.

401	56	Sesión de usuario invalida.
400	57	La tarjeta ingresada ya se encuentra registrada.
400	58	El alias actualmente ya se encuentra registrado.
500	24	El número de tarjeta no corresponde a la identificación dada.
500	25	La tarjeta existe, pero no es válida para transar.
500	26	El número no es de una tarjeta de crédito.
500	29	Error general.
500	30	Cliente con tarjeta activa en entidad.
500	31	Cliente sin tarjeta activa en entidad.
500	32	Error desconocido en validación de datacredito

9. Eliminar Tarjeta

El servicio de eliminación de tarjeta permitirá a un usuario eliminar alguna de las tarjetas que tiene registradas. No se podrán eliminar todas las tarjetas registradas, debe existir al menos una.

Petición

El método de petición HTTP para este servicio es DELETE

Respuesta

El objeto de respuesta será de tipo JSON y está definido de esta manera:

```
{
    "statusCode": codigoDeEstado,
    "message": Mensaje,
}
```

Donde los campos statusCode será el código de la respuesta, message será el mensaje asociado a la respuesta sobre la petición enviada anteriormente.

De esa manera, los códigos de respuesta y los mensajes que podrán obtenerse de acuerdo al resultado de la operación serán:

HTTPStatus	StatusCode	Message
200	59	Tarjeta Eliminada Exitosamente.
404	60	No se encontró la tarjeta.
500	61	Error al eliminar la tarjeta.
403	63	Como mínimo debe tener una tarjeta registrada. No es posible eliminar esta tarjeta.
401	62	Sesión no valida.

Anexos

ANEXO 1: API RESTful de RedHat SSO.

El componente RedHat SSO expone APIs RESTful con el fin de proporcionar mecanismos para el manejo de las sesiones de las aplicaciones que usarán IA PaGo. A continuación, se detallará la forma en cómo se debe usar los servicios de autenticación, regeneración de token y de cierre de sesión.

- AUTENTICACIÓN:

La URL del servicio pruebas:

https://sso-sso-pruebas.apps-pruebas.credibanco.com/auth/realms/IAPaGo/protocol/openid-connect/token

La URL del servicio en producción:

https://sso-sso-credibanco.crediservices.credibanco.com/auth/realms/IAPaGo/protocol/openid-connect/token

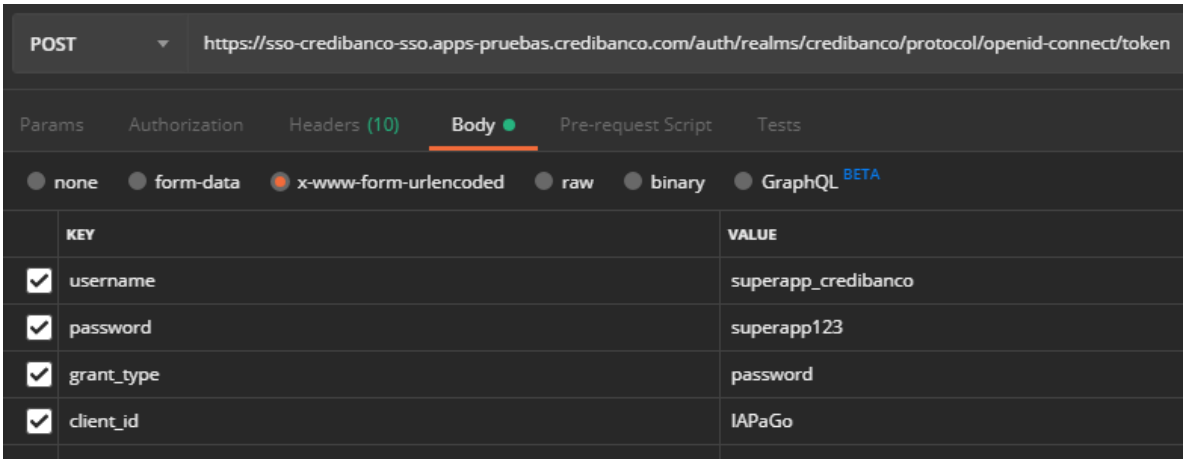
El método HTTP es: POST

El tipo de contenido es: application/x-www-form-urlencoded

El cuerpo de la petición es de tipo "clave-valor" debido al tipo de contenido. Las claves a ingresar y su descripción son:

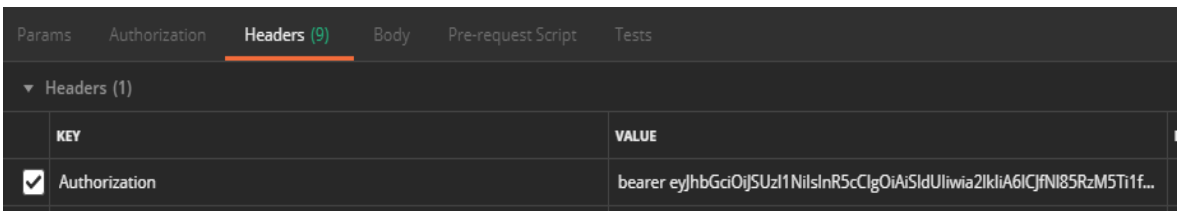
- username: El nombre de usuario.
- password: La contraseña asociada con ese usuario.
- grant_type: Para este servicio su valor será siempre: password
- client_id: El Client ID asignado para la aplicación.

Usando el software Postman para realizar la prueba, la petición quedaría conformada de esta manera:

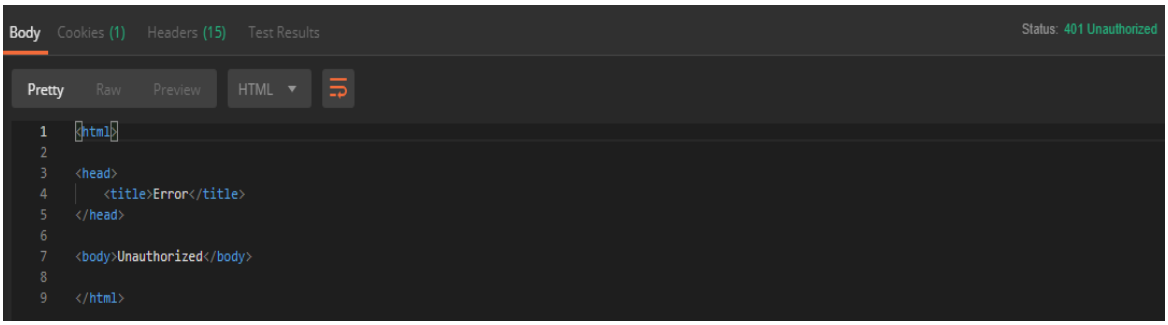


Al realizar la petición el servicio retornará un objeto de autorización que contiene el token de acceso (expira en 300 segundos, o 5 minutos, tal y como se ve en el campo expires_in) y el token de regeneración (expira en 1800 segundos, o 30 minutos, tal y como se ve en el campo refresh_expires_in). Por ejemplo,

El campo access_token es el que debe utilizarse en el encabezado Authorization de cada petición a los servicios de IA PaGo de la siguiente manera: bearer <access_token>, donde <access_token> es el valor que contenga dicho campo. Por ejemplo,

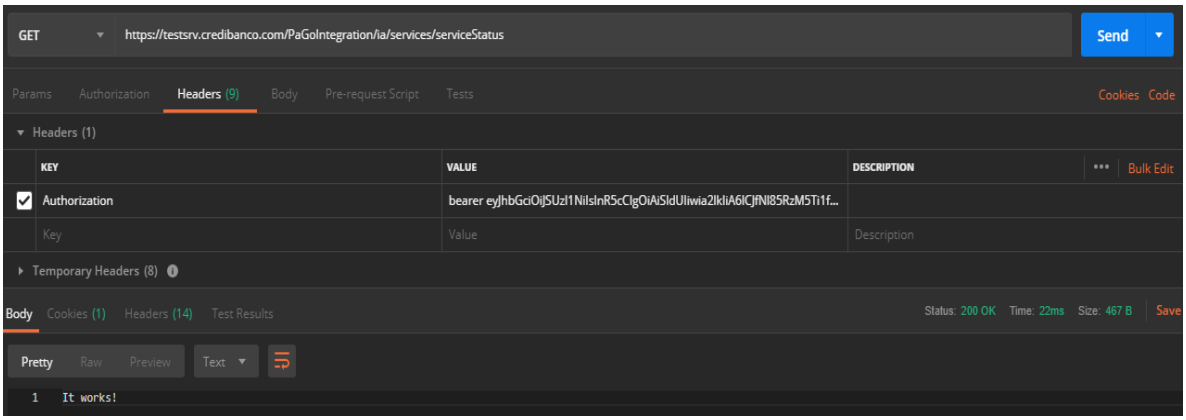


En caso de que el valor del encabezado Authorization sea incorrecto la respuesta que se obtendrá será de esta manera:



Se obtiene un código de respuesta 401 No Autorizado, y un HTML sencillo con la palabra Unauthorized.

Cuando se realiza la autenticación de forma correcta el consumo del servicio se realizará de manera normal. En el caso del servicio de comprobación de estado del servicio (https://testsrv.credibanco.com/PaGoIntegration/ia/services/serviceStatus) será de esta manera:



- REGENERACIÓN DE TOKEN DE ACCESO

El consumo de este servicio será necesario cuando el token de acceso haya expirado y no sea posible el consumo normal de los servicios.

La URL del servicio pruebas:

https://sso-sso-pruebas.apps-pruebas.credibanco.com/auth/realms/IAPaGo/protocol/openid-connect/token

La URL del servicio en producción:

https://sso-sso-credibanco.crediservices.credibanco.com/auth/realms/IAPaGo/protocol/openid-connect/token

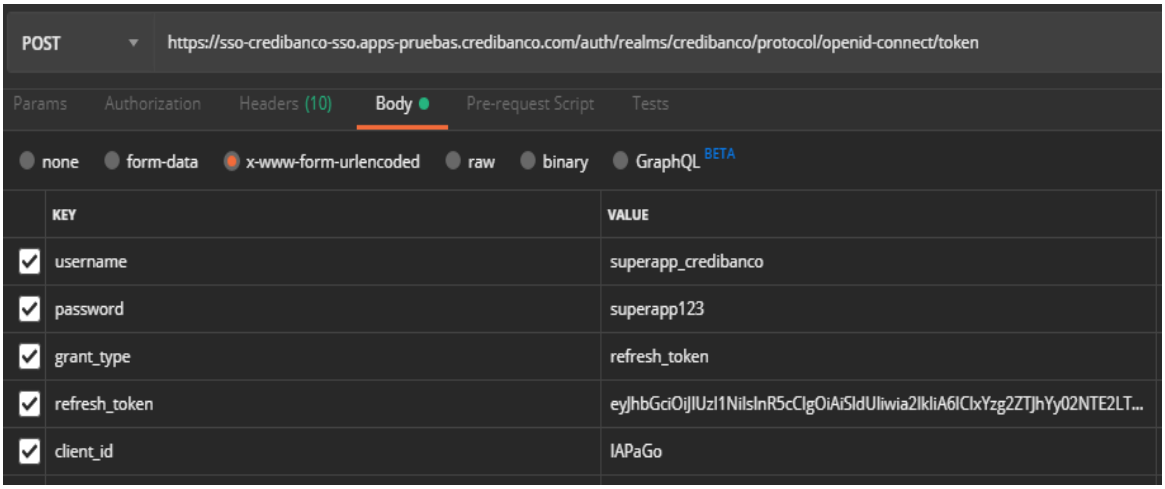
El método HTTP es: POST

El tipo de contenido es: application/x-www-form-urlencoded

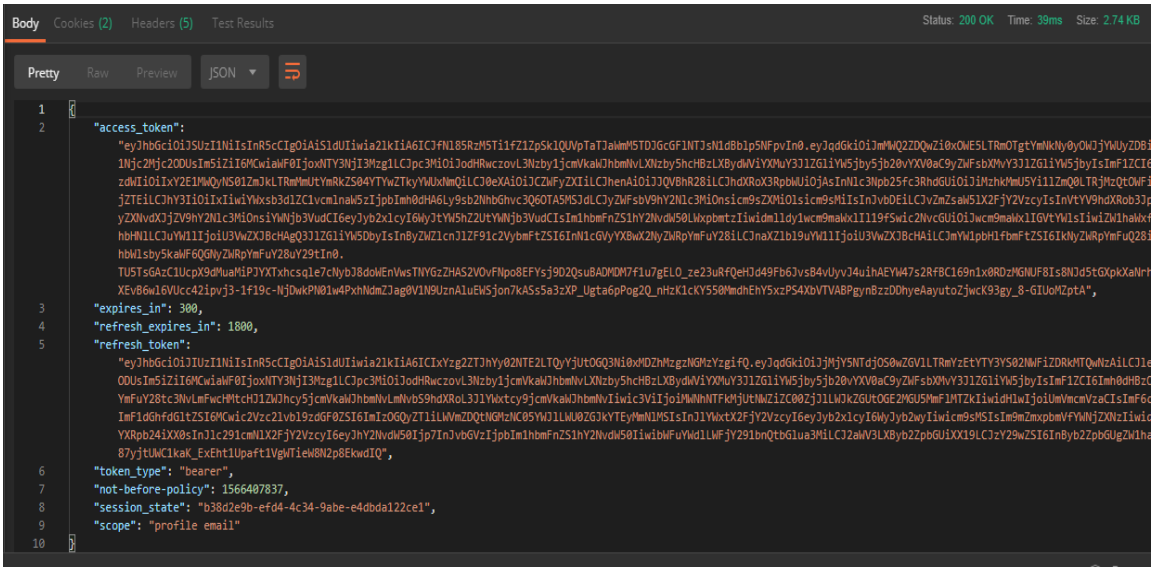
El cuerpo de la petición es de tipo "clave-valor" debido al tipo de contenido. Las claves a ingresar y su descripción son:

- username: El nombre de usuario.
- password: La contraseña asociada con ese usuario.
- grant_type: Para este servicio su valor será siempre: refresh_token
- client_id: El Client ID asignado para la aplicación.
- refresh_token: El "refresh_token" obtenido en el proceso de Autenticación.

Usando el software Postman para realizar la prueba, la petición quedaría conformada de esta manera:



Al realizar la petición el servicio retornará un objeto de autorización que contiene el token de acceso y el token de regeneración. Por ejemplo,



Como se puede observar, el Access_token es diferente en esta nueva petición, y nuevamente tendrá una vigencia de 300 segundos (5 minutos).

- CIERRE DE SESIÓN

Si por algún motivo fuera necesario cerrar la sesión de la aplicación en SSO dicho proceso podría realizarse consumiendo el servicio que expone la API de RedHat SSO.

La URL del servicio pruebas:

https://sso-sso-pruebas.apps-pruebas.credibanco.com/auth/realms/IAPaGo/protocol/openid-connect/logout

La URL del servicio en producción:

https://sso-sso-credibanco.crediservices.credibanco.com/auth/realms/IAPaGo/protocol/openid-connect/logout

El método HTTP es: POST

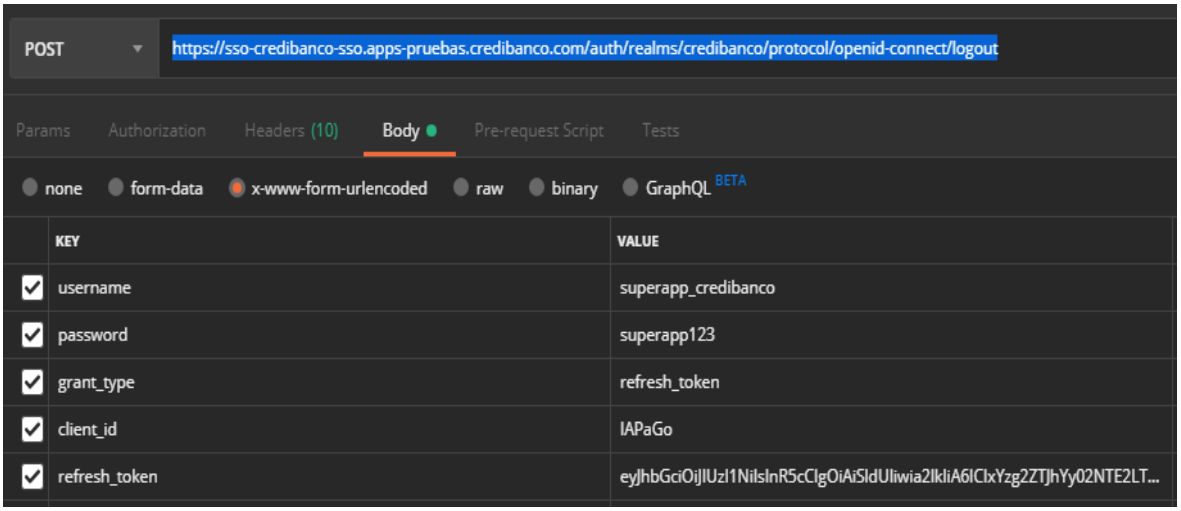
El tipo de contenido es: application/x-www-form-urlencoded

El cuerpo de la petición es de tipo "clave-valor" debido al tipo de contenido. Las claves a ingresar y su descripción son:

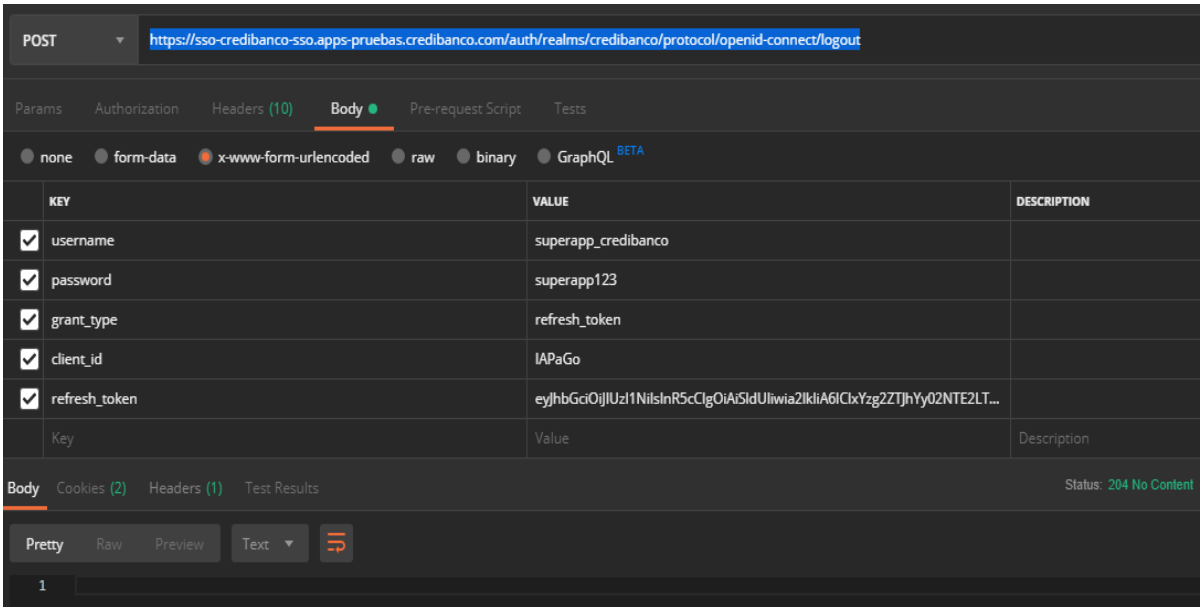
- username: El nombre de usuario.
- password: La contraseña asociada con ese usuario.

- grant_type: Para este servicio su valor será siempre: refresh_token
- client_id: El Client ID asignado para la aplicación.
- refresh_token: El "refresh_token" obtenido en el proceso de Autenticación.

Usando el software Postman para realizar la prueba, la petición quedaría conformada de esta manera:



La respuesta exitosa debe tener el código 204 Sin Contenido y no tener contenido en el cuerpo de la respuesta.



De esa manera, quedará cerrada la sesión y tanto el Access_token como el Refresh_token quedarán invalidados y todas las peticiones en donde se usen darán errores de no autorización.

ACLARACIONES IMPORTANTES

- El token de acceso es el que debe utilizarse como credencial de autorización al momento de consumir cada uno de los servicios que compone el IA PaGo.
- El nombre de la cabecera HTTP que se debe usar para la autorización es: Authorization
- El esquema en donde estará el token de acceso se llamará: bearer, dando cumplimiento a lo estipulado por el RFC 7519.
- El valor que debe tener la cabecera Authorization debe ser de la forma: bearer <access_token>, donde access_token es el valor del campo access_token (sin los paréntesis <>) que está en el objeto de respuesta del consumo de los servicios de autenticación y regeneración de token.
- La vigencia del token de acceso está dada por el campo expires_in que está en el objeto de respuesta del consumo de los servicios de autenticación y regeneración de token. Dicho campo es de tipo entero y su unidad de medida es en segundos.
- La vigencia del token de regeneración está dada por el campo refresh_expires_in que está en el objeto de respuesta del consumo de los servicios de autenticación y regeneración de token. Dicho campo es de tipo entero y su unidad de medida es en segundos.
- En el entorno de pruebas del SSO está configurado el tiempo de vigencia del token de acceso en 300 segundos, que es igual a 5 minutos.
- En el entorno de pruebas del SSO está configurado el tiempo de vigencia del token de acceso en 1800 segundos, que es igual a 30 minutos.
- Si bien en este ejemplo se usa Postman y los valores de los tokens de acceso y regeneración son conocidos, en la práctica dichos valores no deben ser divulgados.
- En la práctica los únicos valores que deben ser conocidos para un ser humano debe ser el nombre de usuario y la contraseña de acceso a SSO.

No es necesario almacenar el valor del token de regeneración en ningún momento. De ser necesario un nuevo token de regeneración, usar nuevamente el servicio de Autenticación.

ANEXO 2: Códigos de Error - Vpayment

Se listan los códigos de error correspondientes a errorCode y errorMessage del servicio de Pago tradicional.

ErrorCode	Error Message
00	Aprobada
01	Negada, comuniquese con su entidad
02	Negada, comuniquese con su entidad
03	Negada, comercio inválido
04	Negada, retener tarjeta
05	Negada, puede ser tarjeta bloqueada o timeout
06	Negada, no se pudo procesar la transacción
07	Negada, retener tarjeta
08	Aprobada, solicitar más información
09	Negada, transacción duplicada
11	Aprobada, vip
12	Negada, transacción inválida
13	Negada, monto inválido
14	Negada, estado de la tarjeta inválido
15	Negada, la institución no está en el IDF
16	Negada, Numero cuotas invalidas
30	Negada, error en edición de mensaje
31	Negada, el emisor no es soportado por el Sistema
33	Negada, tarjeta vencida con orden de retención
34	Negada, retener/capturar
35	Negada, retener/capturar
36	Negada, retener tarjeta
37	Negada, tarjeta bloqueada retener/capturar
38	Negada, número de intentos del PIN excedidos
39	Negada, puede ser tarjeta bloqueada o timeout
41	Negada, tarjeta robada o extraviada
43	Negada, estado en archivo de tarjetahabientes CAF
51	Fondos insuficientes
54	Negada, tarjeta vencida
55	Negada, PIN inválido
56	Negada, no se encontro CAF
57	Negada, transacción no permitida a esta tarjeta
58	Negada, transacción Inválida
61	Negada, excede el monto máximo
62	Negada, tarjeta restringida

ErrorCode	Error Message
65	Negada, límite de usos por período excedido
68	Negada, TIMEOUT
70	Negada, tarjeta vencida
71	Negada, El tipo de cuenta no corresponde
75	Negada, número de intentos de PIN excedidos
76	Aprobada, (Privado)
77	Aprobada, pendiente identificación firma del comp
78	Aprobada a ciegas
79	Aprobada, transacción administrativa
80	Aprobada por boletín de seguridad
81	Aprobada por el establecimiento
82	Negada, no hay módulo de seguridad
83	Negada, no hay cuenta para la tarjeta
84	Negada, no existe el archivo de saldos PBF
85	Negada, error en actualización de archivo saldo
86	Negada, tipo de autorización errado
87	Negada, track 2 errado
88	Negada, error en log de transacciones PTLF
89	Negada, inválida la ruta de servicio
90	Negada, no es posible autorizar
91	Negada, no es posible autorizar
92	Negada, puede ser tarjeta bloqueada o timeout
93	Negada, no es posible autorizar
94	Negada, transacción duplicada
96	Negada, no se pudo procesar la transacción
97	Negada, Número de documento inválido
98	Negada, CVV2 invalido
99	Error de comunicaciones
A1	Transacción autorizada, sujeta a evaluación **(Si tiene modulo antifraude)**
A3	La transacción fue anulada automáticamente por CyberSource **(Si tiene modulo antifraude)** Ver Transacciones con Módulo de Seguridad
B1	Numero de Factura invalida
B2	Factura vencida
B3	Factura pagada
F1	Filtro por Comercio
F2	FILTRO POR AGENCIA
F3	Bin no permitido para esta aerolínea
N0	Negada, no es posible autorizar
N1	Negada, longitud del número de PAN inválida
N2	Negada, se llenó el archivo de preautorizaciones
N3	Negada, límite de retiros en línea excedido

ErrorCode	Error Message
N4	Negada, límite retiros fuera de línea excedido
N5	Negada, límite de crédito por retiro excedido
N6	Negada, límite de retiros de crédito excedido
N7	Negada, customer selected negative file reason
N8	Negada, excede límite de piso
N9	Negada, maximum number of refund credit
O0	Negada, referral file full
O1	Negada, NEG file problem
O2	Negada, advances less than minimum
O3	Negada, delinquent
O4	Negada, over limit table
O5	Negada, PIN required
O6	Negada, mod 10 check
O7	Negada, force post
O8	Negada, bad PBF
O9	Negada, NEG file problem
P0	Negada, CAF problem
P1	Negada, over daily limit
P3	Negada, advance less than minimum
P4	Negada, number times used
P5	Negada, delinquent
P6	Negada, over limit table
P7	Negada, advance less than minimum
P8	Negada, administrative card needed
P9	Negada, enter lesser amount
Q0	Negada, invalid transaction date
Q1	Negada, Fecha de vencimiento invalida
Q2	Negada, invalid transaction code
Q3	Negada, valor del avance menor que el mínimo
Q4	Negada, excedido el número de usos por período
Q5	Negada, delinquent
Q6	Negada, tabla de límites excedida
Q7	Negada, el valor excede al máximo
Q8	Negada, no se encuentra la tarjeta administrativa
Q9	Negada, tarjeta administrativa no está permitida
R0	Negada, transacción admin aprobada en ven
R1	Negada, transacción admin aprobada fuera
R2	Negada, transacción administrativa aprobada
R3	Negada, la transacción Chargeback es aprobada
R4	Negada, devolución/drchivo de usuario actualizado

ErrorCode	Error Message
R5	Negada, devolución/número de prefijo incorrecto
R6	Negada, devolución código de rspta incorrecto
R7	Negada, transacción administrativa no soportada
R8	Negada, la tarjeta está en archivo de negativos
S4	Negada, PTLF full
S5	Negada, devolución aprobada, archivo de cliente n
S6	Negada, devolución aprobada, archivo de cliente n
S7	Negada, devolución aceptada, destino incorrecto
S8	Negada, ADMIN file problem
S9	Negada, unable to validate PIN, security module is
T1	Negada, tarjeta de crédito inválida
T2	Negada, fecha de transacción inválida
T3	Negada, card not supported
T4	Negada, amount over maximum
T5	Negada, CAF status = 0 or 9
T6	Negada, Bad UAF
T7	Negada, límite diario excedido en el Cash back
T8	Negada, el enlace esta caido
TO	Negada, time out
1001	Invalid Data sent by Commerce
1002	Invalid Currency for Commerce
1003	Invalid Commerce Code
1004	Duplicated order number
1101	Communication Problem
1102	Processing Problem
1103	Communication Problem
1104	Invalida Data
1107	Invalid VCI
1114	Card Number does not belong to selected brand
2100	Implementor Class not existent
2101	Default Plan Quotas inexistent
2102	Existing BIN for the acquirer but does not have Plan Quotas assigned
2200	Acquirer must need more data sent in the Plug-In
2201	Plan Quotas not sent by Commerce
2202	Duplicated order number
2308	Card Brand is not correct
2309	Card Number is not correct
2310	Card Expiry Date is not correct
2311	Card Security Code is not correct
2312	Card Number is not present and is required

ErrorCode	Error Message
2313	Commerce not well configured
2314	Acquirer not well configured
2315	Will not go to authorization due to Pre-Authorization Rules
2316	Transaction has been authorized but Post-Authentication Rules refused it
2317	Transaction has been authorized but has been reversed by Post-Authentication Rules
2303	PurchaseCode Field must not be greater than 12 characters
2305	Commerce is not active
2306	Acquirer is not active
2307	Transaction has been processed
2318	PurchaseOperationNumber is greater than 8 characters (Maximum for Associated Commerce)
2319	PurchaseOperationNumber is greater than 12 characters
2400	Transaction is rejected and will not be authorized due to pre-authorization rules
2401	Pre Authentication rules not approved
2402	Error processing authorization
2403	Maximum Monthly accumulated amount has been reached
2404	Maximum Daily accumulated amount has been reached
2405	Maximum Daily order number has been reached
2406	Maximum Monthly order number has been reached
2410	Mall cannot start transactions
2411	Shipper is needed but not found
2412	Shipping amount is not valid
2413	Purchase amount is not valid
2414	Associated Commerce amount is not valid
2415	Purchase Amount and Associated Commerce amount are not valid
2417	Iva amount is not valid
2418	Dev Iva amount is not valid
2500	Invalid amount format
2501	Purchase amount is less than iva
2502	Maximum iva exceeded
2503	Tip amount is not valid
2504	Maximum tip exceeded
2505	Airport tax is not valid
2506	Maximum airport tax exceeded
2507	Purchase amount is less than iva + dev iva
3001	Rejected by Ecommerce rule
3002	Rejected by Ecommerce rule Error\t
3003	Rejected by Ecommerce rule Comm. Error
5000	Certificate does not belong to commerceId sent
2320	Commerce does not belong to Acquirer Sent

ErrorCode	Error Message
2304	Commerce does not exist
2203	Currency does not exists
2203	Currency does not exists
4006	Financial associated with the card not registered in the system
4007	Airline is not registered in the system
CB01	Lo sentimos, la transacción fue bloqueada por listas.
CB02	Lo sentimos, la transacción no puede ser procesada por parámetros de seguridad.
CB03	Lo sentimos, la transacción fue bloqueada por reintentos denegados.
CB04	Lo sentimos, la transacción no puede ser procesada Filtro por Agencia.
CB05	Lo sentimos, BIN no permitido para esta aerolínea.

ANEXO 3: Términos y condiciones de PaGo

Se debe integrar la siguiente declaración en la página o APP al momento de solicitar el registro del usuario: "declaro que he leído, entendido y aceptado los términos y condiciones de CredibanCo S.A y su política de tratamiento de datos".

A continuación, se relaciona los términos y condiciones de PaGo para que sean publicado en la aplicación:

Términos y
Condiciones para el se