Suscripciones
Plexo

Módulo de Suscripciones

Manual de integración v2.2

Handy

Secciones

1. Introducción..... 2
a) Descripción general del producto.....2
b) Principales conceptos de la plataforma..... 2
i) Cliente..... 2
ii) Tipos de suscripción.....2
iii) Suscriptores.....2
iv) Suscripción.....2
v) Issuer.....3
vi) Comercios..... 3
vii) Medios de pago..... 3
viii) Transacción.....3
ix) Suscriptor de cortesía.....3
x) RedirectUrl..... 3
c) Descripción de componentes.....3
i) Módulo de suscripciones (API).....3
ii) Backoffice de suscripciones..... 3
iii) Sitio del usuario final.....4
iv) Pasarela de pagos Plexo.....4
2. Integración..... 5
a) Configuración previa a la integración.....5
i) Gestión..... 5
ii) Backoffice..... 5
b) Formas de integrarse (descripción de flujo y conteo de métodos)..... 10
i) Sencilla..... 11
ii) Intermedia..... 11
iii) Avanzada..... 11
c) Documentación de API..... 11
i) Consideraciones generales..... 11
ii) Métodos..... 12
iii) Clases.....20
iv) Visa DeviceFingerprint.....25
3. Recursos adicionales..... 27
a) Swagger.....27
b) Tarjetas ficticias para pagos en ambiente de test..... 27
c) Códigos de comercio por sello en ambiente de test..... 27

1. Introducción

a) Descripción general del producto

El Módulo de Suscripciones de Plexo permite delegar la gestión de pagos recurrentes de tus usuarios. Cuenta con un backoffice al que podrás acceder para especificar el monto y la frecuencia con la que se harán los cobros de tus suscripciones, así como para consultar información de tus suscriptores. El módulo forma parte de una pasarela de pagos, mediante la cual se harán efectivos los pagos.
Para empezar a usar el módulo e integrarlo con tu ecommerce o sitio institucional solo tienes que realizar una sencilla integración de servicios y encargarte de la experiencia de usuario que quieras darle a tus suscriptores.

b) Principales conceptos de la plataforma

A lo largo de este documento se utilizarán estos términos:

i) Cliente

Es la empresa que se integra al el Módulo de Suscripciones con el objetivo de que sus usuarios se puedan suscribir y efectivamente pagar el servicio de una suscripción. Su creación en el sistema corre por cuenta de Plexo que le brindará al cliente toda la información necesaria para que pueda gestionarlo. El sistema, para poder identificar el cliente, utiliza dos atributos que Plexo proporcionará: ClientId y ClientSecret. El ClientId es fijo, pero el ClientSecret puede ser cambiado por el usuario en cualquier momento.

ii) Tipos de suscripción

El cliente puede ofrecer distintos tipos de suscripciones que podrán diferir en su frecuencia de pago, costo, nombre, moneda, etc. También se podrá especificar la cantidad de ciclos en los que la suscripción debe permanecer activa (por ejemplo “6 meses”). Para poder operar en el sistema es necesario que haya al menos uno creado.

iii) Suscriptores

Son los usuarios que efectivamente se suscriben a alguno de los tipos de suscripción configurados por el cliente. Se debe proveer un identificador único con el cual distinguirlos, que el sistema llama “external Id”.

iv) Suscripción

Un usuario se puede suscribir a uno de los tipos de suscripción ofrecidos por el cliente, generando una suscripción. Esta tendrá un período de validez, identificador, status, etc.

v) Issuer

Es un proveedor de medios de pago, como puede ser VISA, OCA, MasterCard, etc.

vi) Comercios

Es una relación entre el cliente y los diferentes *issuers* que soporta el módulo de suscripciones. Para poder operar con cualquiera de ellos el cliente debe crear un comercio en el sistema (uno por cada *issuer* que el cliente quiera ofrecer como opción de pago). Para crearlo es necesario que ingrese información (usualmente código de comercio y/o número de terminal) que le proveerá el propio *issuer*.

vii) Medios de pago

Para poder pagar una suscripción, un suscriptor debe tener registrado un medio de pago a su nombre en alguno de los issuers en los que el cliente tenga un comercio creado. El medio de pago puede ser cambiado.

viii) Transacción

Al tratarse de una suscripción a un servicio, el usuario suscriptor efectuará pagos mientras su suscripción sea válida, siendo el sistema quien gestiona estos pagos, llamándolos “transacciones”. El cliente podrá acceder a un historial de transacciones.

ix) Suscriptor de cortesía

Se trata de un suscriptor que no paga. Está pensado para servicios gratuitos o para suscribir al personal de la empresa. Estos suscriptores son dados de alta por el administrador del cliente en el backoffice.

x) RedirectUrl

Algunos datos por contener información sensible (por ejemplo, número de tarjeta de crédito) deben ser cargados en una página de Plexo. Luego de cargarse los datos correspondientes, la página de Plexo redirigirá a una Url indicada por el cliente como ***RedirectUrl***

c) Descripción de componentes

i) Módulo de suscripciones (API)

API que va a ofrecer los servicios para la integración del módulo de suscripciones y que va a contener toda la lógica del funcionamiento del mismo.

ii) Backoffice de suscripciones

Sitio web en el que el cliente se podrá loguear y para visualizar información y administrar el módulo de suscripciones:

- Configurar los tipos de suscripciones y sus características
- Consultar los suscriptores que tiene cada tipo de suscripción
- Configurar comercios
- Consultar estadísticas sobre la evolución de la cantidad de suscriptores a través del tiempo
- Consultar las transacciones correspondientes a cada tipo de suscripción y su estado
(Exitosa/Fallida)
- Consultar qué suscriptores de cortesía existen, eliminarlos y agregar otros.

iii) Sitio del usuario final

Sitio web del cliente desde el que se consumirá la API del módulo de suscripciones. Dicho sitio deberá poder gestionar sus propios usuarios teniendo una id única para cada uno, con la cual se va a identificar al usuario como suscriptor dentro del módulo (a lo que de ahora en adelante nos referiremos como ExternalId).

iv) Pasarela de pagos Plexo

Servicio que ofrece Plexo para facilitar y dar seguridad a la autorización de pagos electrónicos y la transferencia de datos hacia las instituciones adquirentes o redes de pagos.

2. Integración

a) Configuración previa a la integración

Previo a la integración es necesario realizar una configuración inicial del cliente. Esta configuración se puede dividir en dos grandes categorías, de gestión y de backoffice, las cuales son explicadas a continuación.

i) Gestión

La gestión se refiere a los pasos operativos necesarios que se requieren para poder comenzar la integración.

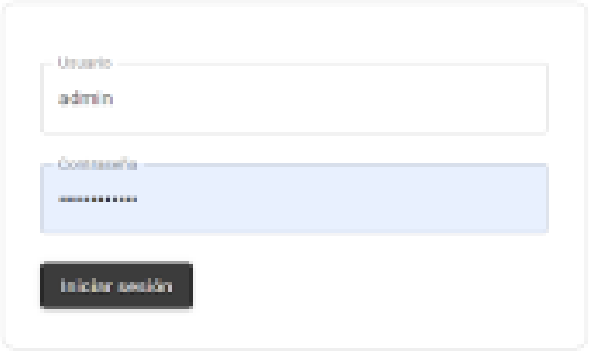
1) El cliente deberá brindar ciertos datos necesarios (nombre de comercio, razón social y rut), para poder comenzar con la configuración.

2) El equipo de suscripciones enviará al cliente las credenciales y la url para acceder a los servicios del módulo en el ambiente que corresponda (inicialmente será el de test).

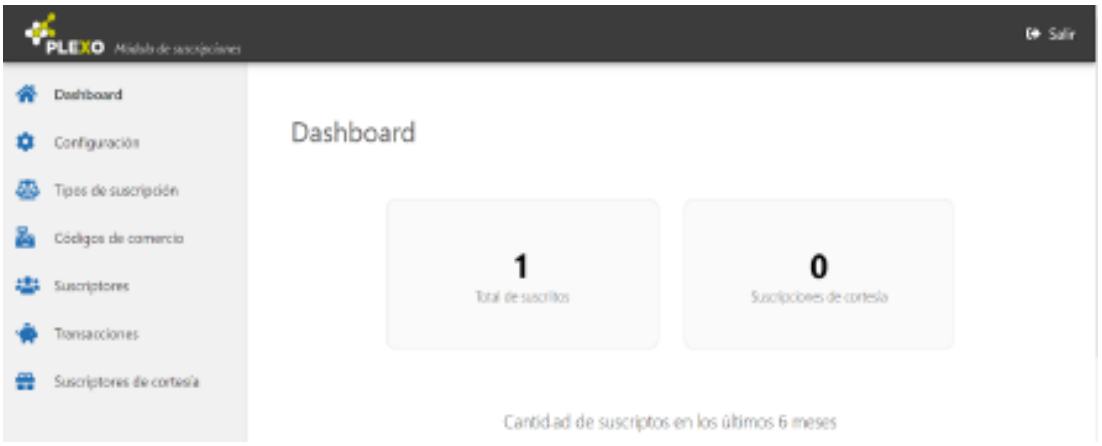
ii) Backoffice

Como se mencionó previamente, este es un sitio web proporcionado por Plexo con el objetivo de que el cliente pueda gestionar el módulo de forma fácil e intuitiva.

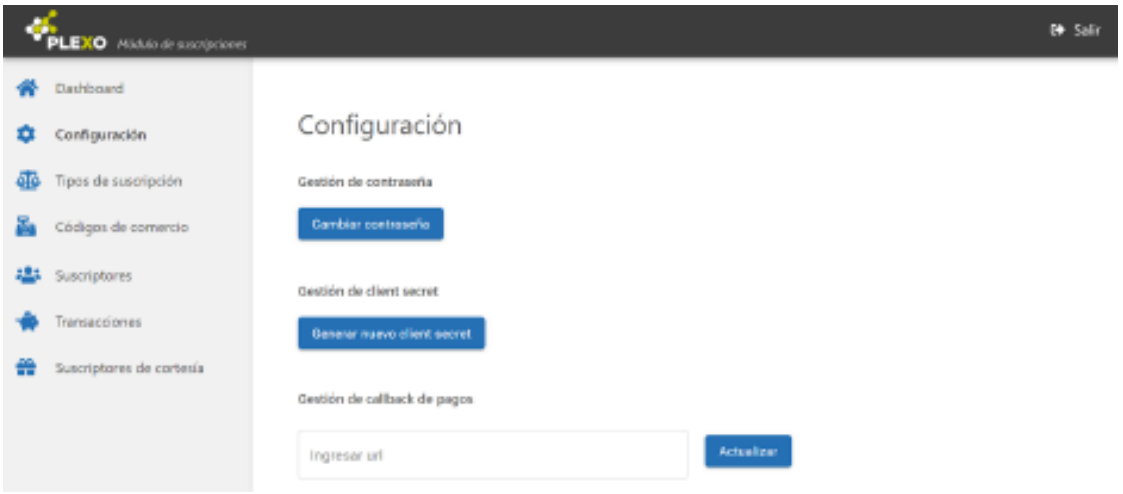
1) Al ingresar al sitio, lo primero que debe hacerse es iniciar sesión con el usuario administrador proporcionado por Plexo. Este usuario sólo permite gestionar un cliente.



2) Una vez iniciada la sesión, el usuario podrá ver el dashboard, con algunos datos importantes, como el total de usuarios suscriptos y su evolución a lo largo del tiempo.

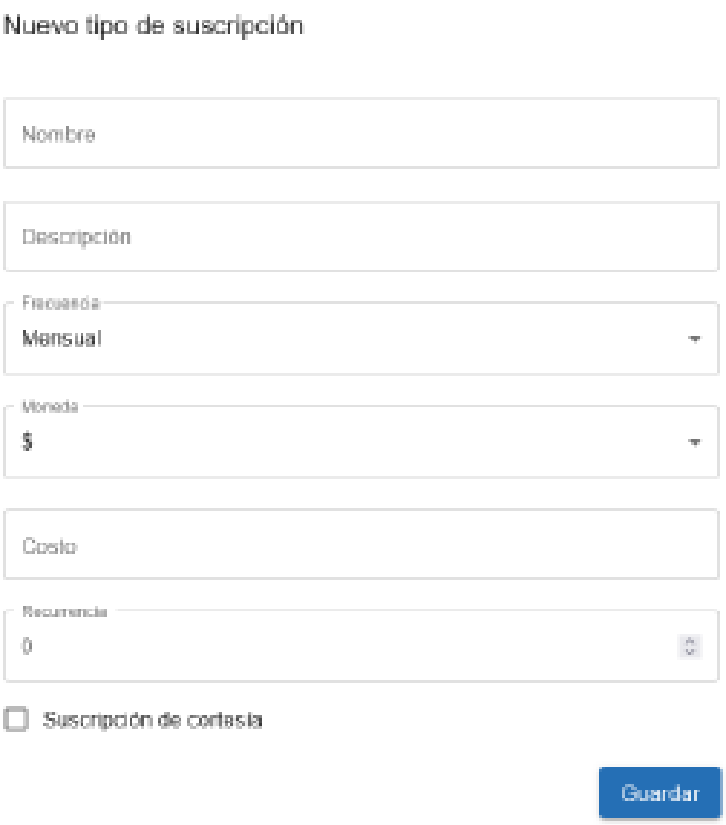


3) En la pantalla de configuración se pueden realizar funcionalidades básicas como cambiar la contraseña del usuario administrador, generar un nuevo client Secret y definir una URL de Callback para los pagos recurrentes (si se configuró esta URL, en caso de que falle un pago, el módulo le avisará al cliente que el pago falló a esa dirección).



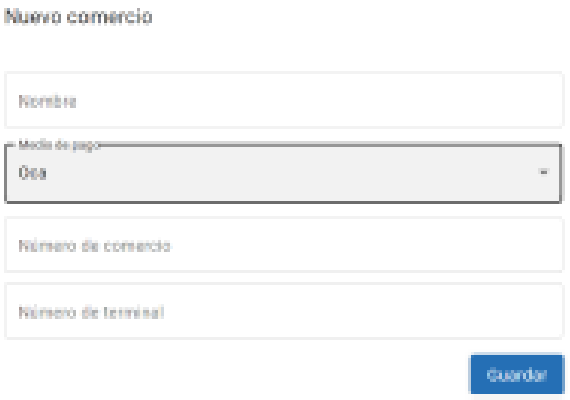
4) En la pestaña tipos de suscripción, el usuario podrá ver aquellas existentes, editarlas, eliminarlas o crear nuevas. Para editar una suscripción se hace click sobre ella, una vez dentro de ella se puede editar sus campos o eliminarla con el botón .

Para crear un nuevo tipo de suscripción, basta con hacer click en el botón “Nuevo tipo de suscripción”. Allí se completan los campos relevantes, se indica si es una suscripción de cortesía y al hacer click en “Guardar” quedará creado el tipo de suscripción.

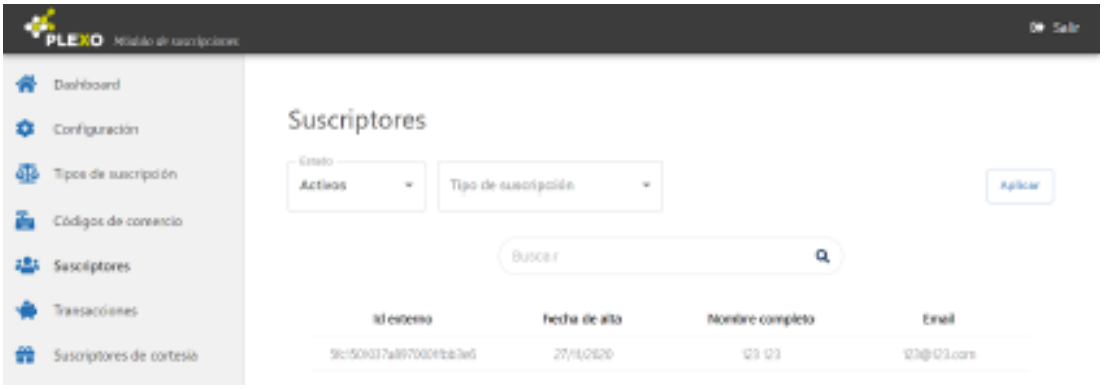


5) Como se mencionó, el usuario debe crear Comercios para poder recibir pagos. Esto se puede realizar desde la pestaña Comercios. De forma análoga, es posible eliminar los comercios aunque no es posible editarlos.

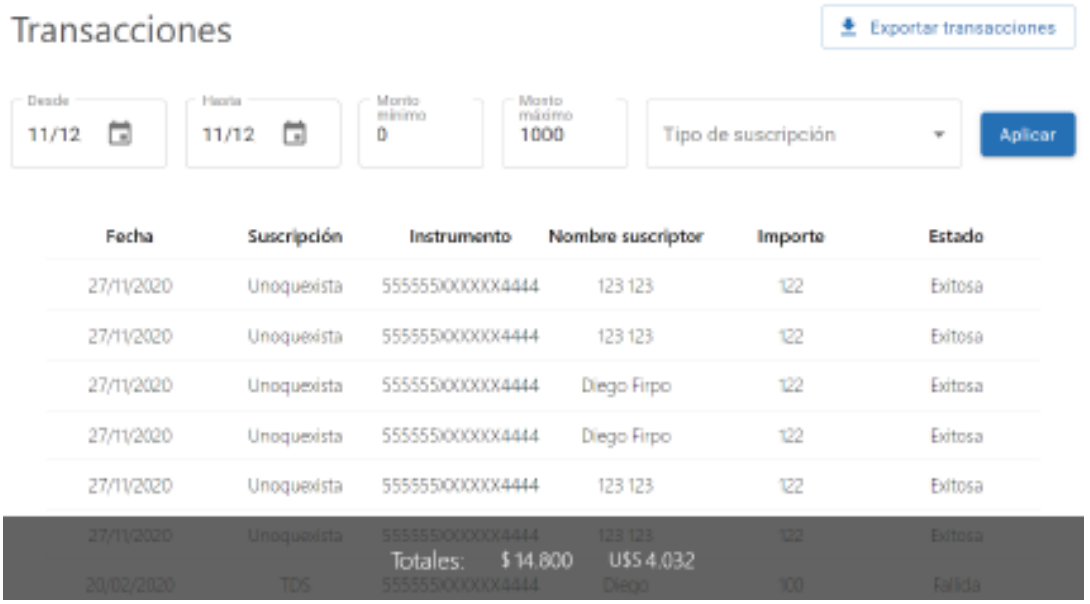
Además es posible agregar nuevos comercios haciendo click en “Agregar comercio”. Aquí se debe ingresar el nombre, el issuer, e información específica dependiente del issuer que el cliente debe obtener del issuer.



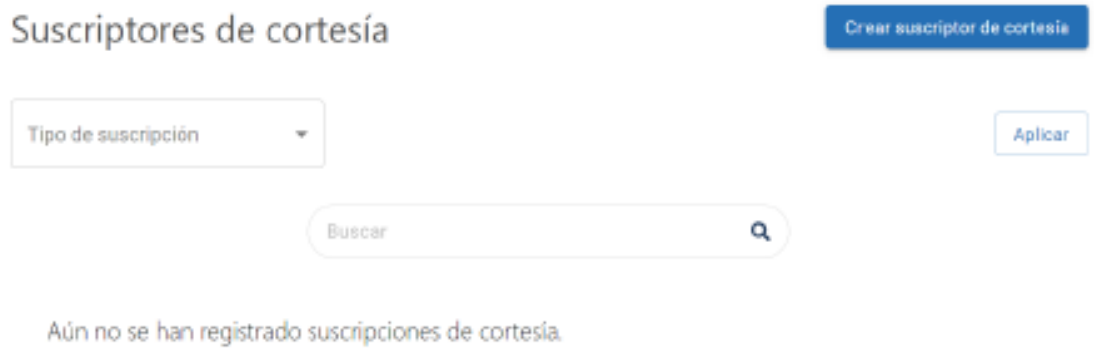
6) Una vez que se tengan todos los pasos anteriores configurados, será posible agregar suscriptores. No es posible agregar suscriptores desde el Backoffice (se explicará más adelante cómo hacerlo), aunque sí es posible hacer consultas y gestionarlos desde aquí en la pestaña “Suscriptores”, pudiendo filtrar según si están activos, por tipo de suscripción y ver datos importantes como su identificador, nombre, correo electrónico, y fecha en que se dio de alta en el módulo.



7) Además, es posible realizar consultas sobre las transacciones realizadas por los suscriptores, en donde se podrá filtrar según varios criterios, ver el total y además exportarlo a formato xlsx (Excel).



8) Por último, es posible obtener información de los suscriptores de cortesía e incluso crearlos desde el sitio.



b) Formas de integrarse (descripción de flujo y conteo de métodos)

Para gestionar el módulo, el cliente puede optar por tres posibilidades diferentes para integrarse. La forma sencilla, la intermedia y la avanzada. Cada una de estas explicaciones serán explicadas a continuación:

Nota: Las especificaciones operativas de cómo llevar a cabo estas tareas se encuentran, para el backoffice, en la sección 2-a-ii y para la API en la sección 2-c-ii.

i) Sencilla

En esta primera posibilidad consiste en configurar en el backoffice, suscribir los usuarios mediante API y luego simplemente utilizar el backoffice proporcionado por Plexo para consultar. Las funcionalidades que va a tener disponible en el backoffice van a ser:

- La visualización de usuarios suscriptores.
- La creación, actualización y eliminación de tipos de suscripciones.
- La creación, actualización y eliminación de los comercios.
- La visualización de de las suscripciones.

Las funcionalidades que va a tener disponible en la API van a ser

- La creación de usuarios suscriptores.
- La creación de la suscripción, en base a el usuarios suscriptores y a los tipos de suscripciones ya creados.

ii) Intermedia

La opción intermedia es para quienes quieran realizar desde un sistema propio las consultas sobre suscripciones y pagos, integrando para ello los servicios de la API del módulo de suscripciones.

iii) Avanzada

Esta última alternativa consiste en consumir exclusivamente la API (tanto para configurar como para suscribir usuarios y consultar información).

c) Documentación de API

i) Consideraciones generales

Headers

Toda solicitud que se haga a la API de suscripciones deberá incluir como headers tanto ClientId como ClientSecret (ambas proporcionadas previo a la integración al módulo de suscripciones). Tanto las solicitudes como las respuestas van a tener Content-type “application/json”.

Respuestas

En caso de credenciales de cliente inválidas (ClientId y ClientSecret), la API devolverá un error 404. En caso de error a la hora de realizar los procesos necesarios para gestionar la solicitud, la API va a devolver un error 400 con la siguiente estructura:

```
{
"code": 400,
"message": "Mensaje de descripción del error",
"subCode": 300
}
```

En caso de éxito, va a devolver una respuesta 200 Ok, junto al objeto de respuesta que sea preciso para cada caso.

ii) Métodos

Se detallarán en esta sección los métodos expuestos en la API del módulo de suscripciones.

Flujo mínimo para suscribir a un usuario

Requiere que estén previamente configurados comercios y tipos de suscripción:

1- Registrar al usuario en la plataforma

POST: api/subscriptors

Body del request:

```
{
"externalId": id,
"email": email
"name": nombre,
"postalCode": código postal,
"fullName": nombre completo
}
```

El **externalId** es el Id del suscriptor dentro del sistema del cliente.
En caso de éxito, el cliente recibirá un código http 200 de respuesta, además de un objeto Suscriptor (revisar sección **iii - Clases** para ver estructuras de objetos de respuesta pertenecientes al dominio)

2- Obtener URL de página en la que el usuario ingresará los datos de su tarjeta

POST: api/instruments

Además del ClientId y el ClientSecret, se deberá incluir como headers el ExternalId y la RedirectUrl, que es la URL hacia la que la web de Plexo va a redirigir al usuario una vez hayaañadido su medio de pago.

Body del request:

```
{
"CallBackURL": url,
"PopulateData": booleano
}
```

Ambos campos pueden estar vacíos.

Si el medio de pago fue añadido con éxito, se notificará en **callbackUrl** incluyendo un set de información básica relacionada.
PopulateData viene por defecto en false. Este campo determina si se le brindará información sobre el usuario final a Plexo para que no tenga que cargar de nuevo datos (como su nombre o su email) a la hora de registrar un medio de pago.

La consulta devolverá una URL a la cual el cliente debe redireccionar para que el usuario pueda cargar los datos de su medio de pago que, como ya fue mencionado, será hecho en un sitio web de Plexo seguro para los datos sensibles del medio de pago del usuario.

A partir de ahora, los clientes que tengan instrumento de pago asociado, podrán suscribirse a tipos de suscripción regulares de los clientes.

3- Suscribir al usuario a un tipo de suscripción
POST: api/subscriptors/subscribe

Body del request:

```
{
"ExternalId": id del suscriptor,
"SubscriptionTypeId": Id del tipo de suscripción,
"LegalId": Rut, (opcional)
"CiberSourceFingerPrint": VISA Fingerprint
}
```

Para más información sobre el VISA Fingerprint, consulte la sección iv - **Visa DeviceFingerprint.**

En caso de éxito, el método devuelve un 200 y un objeto del tipo Subscription (revisar sección **iii - Clases**)

Extra - Comprobar la validez de una suscripción

GET: api/subscriptors/subscriptions/{subscriptionToken}
Retornará si está activada la suscripción así como hasta cuando la misma será válida (antes de intentar el pago recurrente de la misma)

Endpoints adicionales

Medio de pago

GET: api/instruments
Obtiene (si tiene) el medio de pago del usuario establecido por externalId. Además del ClientId y el ClientSecret, se deberá enviar como header el ExternalId.
En caso de éxito, el método devuelve un objeto del tipo Instrument (revisar sección **iii - Clases**)

DELETE: api/instruments/{instrumentId}
Elimina el instrumento de pago, quedando el usuario sin instrumentos de pago asociado. Además del ClientId y el ClientSecret, se deberá incluir como header el ExternalId y el InstrumentId.
Este endpoint devuelve solamente un 200 en caso de éxito.

Manejo de los suscriptores

GET: api/subscriptors/{externalId}
Se obtiene la información que hay en el módulo de suscripciones sobre ese usuario. Este endpoint devuelve un 200 con un objeto **Subscriptor** en caso de éxito.

PUT: api/subscriptors/{externalId}
Para modificar la información del usuario.

Body del request:

```
{
"Email":"string",
"Name":"string",
"PostalCode":"string",
"FullName":"string"
}
```

Retornará un 200 y un objeto Subscriptor en caso de éxito.

GET: api/subscriptors/subscriptions
Obtiene todas las suscripciones asociadas a ese usuario. Además del **ClientSecret** y del **ClientId**, en el header se deberá enviar también el **ExternalId** del subscriptor.
En caso de éxito, devolverá un 200 junto a una lista de objetos del tipo Subscription (Sección iii
- Clases).
Devolverá un 204 en caso de que no haya suscripciones asociadas a ese suscriptor.

PUT: api/subscriptors/subscriptions/{subscriptionToken}?{subscriptionTypeId}
En caso de que el usuario quiera, podrá cambiar el tipo de suscripción de su suscripción, tomando inmediata validez, pero cobrando el nuevo precio a partir del próximo pago recurrente.
En caso de éxito, devolverá un 200 y un objeto del tipo **Subscription**.

POST: api/subscriptors/force/{subscriptionToken}
En caso de que la suscripción haya vencido (por el fallo del instrumento de pago, por ejemplo) el usuario podrá "forzar" el pago de su suscripción si así lo desea. En el header, además del ClientId y el **ClientSecret**, se deberá enviar el **ExternalId**.
En caso de éxito, devolverá un 200 junto a un objeto del tipo **Subscription.**

POST: api/subscriptors/cancel/{subscriptionToken}
En todos los casos cancela la suscripción, no generándose un nuevo cobro recurrente al vencer el período de la frecuencia. En cuanto a cuándo se inactiva la suscripción y qué ocurre con el pago de la última transacción, la lógica depende de cuándo se hizo el pago y del siguiente parámetro en el payload. La lógica es la siguiente:

Request body:

```
{
  "CancelTransaction":"Boolean",
}
```

Si el pago (transacción) fue realizado hace más de 24 horas, se mantendrá la suscripción válida por lo que resta de la frecuencia del tipo de suscripción. Independientemente de lo que se pase en el parámetro, el último pago se mantiene y no es devuelto al TH.

Si el pago (transacción) fue realizado hace menos de 24 horas, dependiendo del parámetro (si viaja en True) se cancelará la última transacción (se devuelve el pago), y la suscripción pasa a estado inactivo inmediatamente. Si el parámetro viaja en False, el comportamiento es como en el caso anterior.

Una vez que está cancelada, si el cliente/suscriptor quiere volver a inscribirse, tendrá que crear una suscripción nueva. En el header, además del **ClientId** y el **ClientSecret**, se deberá enviar el **ExternalId**.

En caso de éxito, retornará un 200.

GET: api/transactions/{subscriptionToken}
Se obtienen todas las transacciones de una suscripción dada. Va a tomar en la query string un objeto del tipo **TransactionQueryAttributes** (sección iii - Clases).
Si no hay transacciones asociadas a dicha suscripción, retornará un 204.

En caso de éxito va a retornar un 200 con un resultado paginado como en el siguiente ejemplo:

```
{
"results": [{Lista de objetos del tipo Transaction}],
"currentPage": "int",
"pageCount": "int",
"pageSize": "int",
"rowCount": "int",
"linkTemplate": "String",
"totals": "decimal",
"totalsDollar": "decimal",
"firstRowOnPage": "int",
"lastRowOnPage": "int"
}
```

Configuración del cliente por API (en lugar del backoffice)

Para todos estos endpoints se necesitará en el header de la request tanto **ClientId** como **ClientSecret**. Si el usuario no es el administrador del cliente, la API devolverá 401 - Unauthorized.

Comercios

GET: api/issuers

Se obtienen los issuers disponibles en la plataforma (VISA, OCA, Dinners, etc) con información relevante en caso de querer agregar el comercio.
En caso de éxito, el endpoint devuelve un 200 junto a una lista de objetos del tipo Issuer.

POST: api/commerces

Se agrega un comercio en plexo, utilizado para los posteriores pagos recurrentes.

Body del request:

```
{
"commerceName":"String",
"commerceNumber":"String",
"commerceTerminalNumber":"String"
"issuerId":"int"
}
```

En caso de éxito, este endpoint devuelve un 200 junto a un objeto del tipo Commerce(sección iii - Clases)

DELETE: api/commerces/{commerceId}
Se elimina el comercio.
Retorna un 200 en caso de éxito.

GET: api/commerces
Se obtienen todos los comercios habilitados por el cliente.

En caso de éxito retorna un 200 y una colección de objetos Commerce

Tipos de suscripción

POST: api/subscriptiontypes
Para agregar un tipo de suscripción en el cliente.

Body del request:

```
{
"Name":"String",
"Description":"String",
"Frequence":Frequence(enum, sección iii),
"SubscriptionCost":decimal,
"Currency":Currency(enum, sección iii),
"RegularType":boolean,
"Recurrence":int
}
```

En caso de éxito, retorna un 200 y un objeto del tipo SubscriptionType (sección iii - Clases)

PUT: api/subscriptiontypes/{subscriptionTypeId}
Modifica un tipo de suscripción.

Body del request:

```
{
"Name":"String",
"Description":"String",
"SubscriptionCost":decimal
}
```

En caso de éxito, retorna 200 y un objeto del tipo SubscriptionType

DELETE: api/subscriptiontypes/{subscriptionTypeId}
Para eliminar un tipo de suscripción dado.
En caso de éxito, retorna un 200.

GET: api/subscriptiontypes
Obtiene los tipos de suscripción activos en el cliente.
En caso de que no haya ningún **SubscriptionType** activo en el cliente dado, retorna 204.
En caso de éxito, retorna 200 y una colección de objetos **SubscriptionType**.

GET: api/subscriptiontypes/all
Obtiene todos los tipos de suscripción.
En caso de que no se haya ingresado aún ningún **SubscriptionType**, retorna 204.
En caso de éxito, retorna 200 y una colección de objetos **SubscriptionType**.

Account

PUT: api/account/callbacks
Se establece el callback que harán los pagos recurrentes al momento de pagar indicado si fue con éxito o no. No hay problema en dejarlo vacío. También sirve para modificarlo.

Body del request:

```
{
"RecurringPaymentCallback":"string"
}
```

En caso de éxito, retorna un 200 y un objeto del tipo AccountStrings, que es igual al objeto dela request

GET: api/account/callbacks
Se obtiene la url de ese mismo callback.
Retorna en caso de éxito un objeto **AccountStrings**.

"Suscriptores

GET: api/subscriptors
Se obtiene, paginado, los suscriptores de la plataforma. En la query string va un objeto del tipo SubscriptorsQueryAttribute para encargarse del paginado. En caso de éxito, retorna un objeto PagedResult con una colección de objetos del tipo **Subscriptor**.

```
{
"results": [{Lista de objetos del tipo Subscriptor}],
"currentPage": "int",
"pageCount": "int",
"pageSize": "int",
"rowCount": "int",
"linkTemplate": "String",
"totals": "decimal",
```

```
"totalsDollar": "decimal",
"firstRowOnPage": "int",
"lastRowOnPage": "int"
}
```

GET: api/subscriptors/export
Se obtienen todos los suscriptores de la plataforma en caso de que se quieran exportar.
Devuelve 200 en caso de éxito, junto a una colección de objetos del tipo Subscriptor

POST: api/subscriptors/courtesy
Agrega un suscriptor de cortesía a un tipo de suscripción anteriormente creado.
El tipo de suscripción debe ser de cortesía. En caso de éxito, devuelve un 200 y un objeto **AddCourtesySubscriptorResponse**, que contiene el id del suscriptor y un objeto del tipo **Subscription**, siendo esta la suscripción de cortesía elegida.

DELETE: api/subscriptors/courtesy
Elimina un suscriptor de cortesía.
Retorna un 200 si tiene éxito.

Transacciones

GET: api/transactions
Se obtienen todas las transacciones en el cliente paginadas. Toma por query string un objeto del tipo TransactionQueryAttributes (sección iii). En caso de éxito, devuelve un objeto del tipo PagedResult (sección iii) conteniendo objetos del tipo Transaction. En caso de que no haya transacciones que cumplan con las condiciones especificadas, devolverá un 204.

GET: api/transactions/export
Se obtienen todas las transacciones en el cliente sin paginación en caso de que se quieran exportar.
Retorna un 200 y una colección de objetos del tipo **Transaction** en caso de éxito.

Home

GET: api/home
Se obtiene la información correspondiente con el Dashboard del BO. En caso de éxito retornará un 200 y un objeto del tipo HomeDataInfo, con esta estructura:

```
{
"TotalSubscriptors": int,
```

```
"CourtesySubscriptors": int,
"HistoricTotalSubscriptors":[
            {
            "Fecha" :"Datetime" ,
            "Nro de suscriptores": "int"
            }
      ]
}
```

Siendo el campo **HistoricTotalSubscriptors** una lista de tuplas representando el número de suscriptores en cada fecha

iii) Clases

Subscriptor

```
{
      "Id": "tipo int,Id del suscriptor generado por el sistema",
      "externalId": "tipo string,Id del suscriptor generado por el cliente",
      "email": "tipo string,correo electrónico del suscriptor",
      "name": "tipo string, Nombre de pila del suscriptor",
      "postalCode": "tipo string, código postal del suscriptor",
      "fullName": "tipo string, nombre completo del suscriptor",
      "Instrument": "tipo Instrument, medio de pago del suscriptor",
      "Subscriptions": "lista de tipo Subscription, suscripciones de suscriptor",
      "Transactions": "lista de tipo Transaction, transacciones de suscriptor"
}
```

Subscription:

```
{
      "Id": "tipo string,Id de la suscriptor generado por el sistema",
      "Token": "tipo string, token generado por el sistema",
      "LegalId": "tipo string, opcional, RUT de suscriptor",
      "SubscriptionType": "tipo SubscriptionType, tipo de suscripción de la
      suscripción",
      "SubscriptionDate": "tipo DateTime, fecha en que se creó la suscripción",
      "SubscriptionStatus": "tipo TransactionState, estado de la suscripción",
      "SuscriptionLastPaymentDate": "tipo DateTime, fecha de último pago",
      "ValidUntil": "tipo DateTime, fecha hasta la cual la suscripción es válida"
}
```

SubscriptionType

```
{
	"Id": "tipo string,Id del tipo de suscripción",

	"Name": "tipo string, nombre del tipo de suscripción",
	"Description": "tipo string, descripción del tipo de suscripción",
	"Frequence": "tipo Frequence, frecuencia de pago",
	"SubscriptionBaseCost": "tipo decimal, costo sin impuestos",
	"TaxPercentage": "tipo string, tasa impositiva",
	"Currency": "tipo Currency,moneda a pagar",
	"RegularType": "tipo bool, false si es de cortesía.",
	"Recurrence": "cantidad de veces que se realizará el cobro del tipo de
suscripción,
	por defecto es cero, lo que significa que no tiene fecha de término",
}
```

Transaction

```
{
	"Id": "tipo int,Id de la transacción",
	"Subscription": "tipo Subscription, suscripción a la que pertenece",
	"Amount": "tipo decimal, costo de la suscripción",
	"CreatedAt": "tipo DateTime, fecha de creación de la suscripción",
	"TransactionState": "tipo TransactionState, estátus",
	"Currency": "tipo Currency, moneda de la transacción",
	"Instrument": "tipo Instrument, medio de pago con el que se pagó",
	"Subscriptor": "tipo Subscriptor, suscriptor al que pertenece",
}
```

TransactionState

```
{
	Enumerado: 0 si es OK, 1 esperando, 2 cancelada, 4 fallada
}
```

Currency

```
{
	Enumerado: 1 peso uruguayo, 2 dólar, 6 peso mexicano
}
```

Frequence

```
{
	Enumerado: Daily, Weekly, Monthly, Bimonthly, Quarterly, Biannual,Annual
}
```

Commerce

```
{
        "Id": "tipo int, identificador del comercio",
        "Name": "tipo string, nombre de comercio",
        "CommerceNumber": "tipo string, número de terminal proporcionado por issuer",
        "CommerceTerminalNumber": "tipo string, número de terminal proporcionado por issuer",
        "Issuer": "tipo issuer, issuer asociado al comercio"
}
```

Instrument

```
{
        "Expiration": "tipo DateTime, fecha de vencimiento del medio de pago",
        "InstrumentId": "tipo string, identificador del instrumento, generado por sistema",
        "IssuerImageUrl": "tipo string, Url con logo del issuer",
        "IssuerName": "tipo string, nombre del issuer",
        "Name": "tipo string, nombre del dueño del medio de pago",
}
```

TransactionQueryAttribute

```
{
        “From” : “tipo Datetime, límite inferior de rango de fechas”,
        “To” : “tipo Datetime, límite superior de rango de fechas”,
        “MinAmount” : “tipo decimal, importe mínimo de las transacciones”,
        “MaxAmount” : “tipo decimal, importe máximo de las transacciones”,
        “SubscriptionTypes” : “”,
        “Currency” : “tipo int (enumerado CurrencyEnum). Moneda de las transacciones”,
        “Page” : “Página que se solicita”,
        “Limit” : “Transacciones que se solicitan por página”,
}
```

Issuer

```
{
        “Id” : “tipo int, id del issuer”,
        “Name” : “tipo string, nombre del issuer”,
        “NormalizedName” : “tipo string, nombre del issuer normalizado”,
        “IssuerImageUrl” : “tipo string, url de imagen de logo del issuer”,
        “PlexoId” : “tipo int, id del issuer dentro de Plexo”,
        “Active” : “Booleano, marca si está activo o no”,
        “Fields” : “Lista de objetos del tipo FieldInfo”,
}
```

PagedResult

Para responses paginadas. Hereda de PagedResultBase. Se colocan acá todos los atributos tanto de la subclase como de la superclase por motivos prácticos.

```
{
	"CurrentPage" : "tipo int, página actual",
	"PageCount" : "tipo int, total de páginas",
	"PageSize" : "tipo int, registros por página",
	"RowCount" : "tipo int, total de items",
	"LinkTemplate" : "tipo int, en desuso",
	"Totals" : "tipo decimal, monto total en pesos",
	"TotalsDollar" : "tipo decimal, monto total en dolares",
	"FirstRowOnPage" : "tipo int, nro de registro general del primer registro de la
	página",
	"LastRowOnPage" : "tipo int, nro de registro general del último registro de la
	página",
	"Results" : "Lista de objetos del tipo especificado según el caso"
}
```

FieldInfo

```
{
	"LabelName" : "tipo string, nombre del campo",
	"FieldType" : "tipo FieldType (enumerado)",
	"Required" : "tipo boolean, marca si el campo es requerido o no"
}
```

FieldType

```
{
        Expiration = 257,
        Name = 258,
        Address = 259,
        ZipCode = 260,
        Email = 261,
        Phone = 262,
        Cellphone = 263,
        AmountLimitExtension = 264,
        Birthdate = 265,
        InstrumentName = 266,
        Identification = 267,
        IdentificationType = 268,
        IdentificationTypeExtended = 269,
        AccountNumber = 270,
        FirstName = 271,
        LastName = 272,
        City = 273,
        Country = 513,
        ShippingAddress = 514,
        ShippingZipCode = 515,
        ShippingCity = 516,
        ShippingCountry = 517,
        PromotionalCode = 518,
        CommerceReferenceId = 519,
        TransactionDateTime = 520,
        DeferredMonths = 521,
        Plan = 522,
        RecurringPayment = 523,
        Provider = 1025,
        SistarBancPaymentMethod = 1281,
        RedPagosProductNumber = 1282,
        RedPagosUserEnabled = 1283,
        VisaNetUserId = 1284,
        CardType = 1285,
        CardIssuer = 1286,
        CybersourceDeviceFingerprint = 1287,
        ClientIP = 1288,
        IntegerId = 1289,
        RefundIntegerId = 1296,
        ProviderCommerceNumber = 2049,
        OcaTaxiCode = 2050,
        TerminalNumber = 2051,
        PosNumber = 2052,
        ProviderMerchantId = 2053,
        ProviderBranchNumber = 2054,
        CommerceReserveExpirationInSeconds = 2055,
        AuthServiceCommerceIndicator = 2305,
        AuthServiceXid = 2306,
        AuthServiceEciRaw = 2307,
        AuthServiceCavv = 2308,
        Pan = 33025,
        Token = 33026,
        UniqueId = 33027,
```

```
        Pin = 33153,
        CVC = 33154,
}
```

iv) Visa DeviceFingerprint

Introducción

Se trata de un mecanismo de seguridad implementado por Visa que Plexo debe aplicar.
En síntesis: se utiliza una librería para capturar datos del navegador del usuario, los que configuran una identidad (fingerprint) que luego es requerida para efectuar pagos. De esta manera los pagos siempre están respaldados por actividad del propio tarjetahabiente, eliminando un buen porcentaje de probabilidades de fraude.

Puesta en marcha

Antes de hacer una invocación a Plexo con el token del medio de pago, desde la Web o APP de e-commerce se debe obtener el DeviceFingerprint del dispositivo del usuario final (un script en la Web, o API desde una APP).
La URL es
https://h.onlinemetrix.net/fp/tags.js?org_id={IdEnvironment}&session_id={IdComercio}{IdInvocacion} donde:

- **IdEnvironment**: donde se debe utilizar 1snn5n9w para test y k8vif92e para producción.
- **IdComercio**: ID del comercio de VISA para el cual se va a hacer la transacción.
- **IdInvocacion**: ID que tiene que ser único por invocación. Se recomienda utilizar el mismo ID de la transacción de pago.

Agregar Script de invocación a la Web de e-commerce:

```
<head>
<script type="text/javascript"
src="https://h.online-metrix.net/fp/tags.js?org_id=k8vif92e&session_id=visanetuy_px_1234TRX2157"></script>
</head>

<body>
      <noscript>
            <iframe style="width: 100px; height: 100px; border: 0;
            position: absolute; top: -5000px;"
            src="https://h.online-metrix.net/fp/tags?org_id=k8vif92e&
            session_id=visanetuy_px_1234TRX2157"></iframe>
         </noscript>
</body>
```

En este ejemplo, se ejecuta este Script en el e-commerce del cliente antes de invocar a Plexo,para el comercio visanetuy_px_1234 y con un ID “TRX2157”.
Luego, en el momento de invocar a Plexo, se envía en el campo opcional

CybersourceDeviceFingerprint el valor TRX2157. En el caso de requerir implementar esto para una APP, ponerse en contacto con el grupo de soporte de Plexo para poder ampliar la información.

3. Recursos adicionales

a) Swagger

En el ambiente para pruebas se encuentra disponible un servicio Swagger que permite estudiar y conocer todas las características de los métodos de la API expuestos, así como interactuar con ellos realizando pruebas en el momento.

Link: https://suscripcionestest.handsoft.com.uy/swagger/index.html#/

b) Tarjetas ficticias para pagos en ambiente de test

En el ambiente de Test se pueden realizar pagos que permitan pruebas en todo el ciclo de integración con el producto. Estos pagos no necesariamente deben ser hechos con medios de pago reales, sino que cualquier tarjeta que cumpla con el formato de cada sello será aceptada. En este link se pueden encontrar varios ejemplos para algunas de las tarjetas más populares.
Link: https://www.freeformatter.com/credit-card-number-generator-validator.html

c) Códigos de comercio por sello en ambiente de test

Antes de poder recibir pagos mediante el módulo de suscripciones será necesario configurar Tipos de suscripción (tema abordado en el punto 2.ii) así como comercios, descritos en el punto 1.b.iv de esta documentación. Para crear comercios es necesario contar con códigos de comercio y, en los casos que corresponda, números de terminal. En el ambiente de Producción estos serán oportunamente provistos por el sello correspondiente, pero en el ambiente de Test se deberá utilizar estos valores:

Mastercard
Código de comercio - cualquiera

Visa
Código de comercio - plexo_uy

OCA
Código de comercio - 87989
terminal - 87989SDP