



Manual de Integración API PHP Sistema PayCodex

SF- 20130102-01
Versión 2.0



Historial de Revisión del Documento

Fecha	Versión/Release	Descripción	Autor
02/07/2013	1.0	ComCodex API PHP	Roger Zavala
22/12/2016	2.0	ComCodex API PHP	Juan Carrillo

Tabla de Contenidos

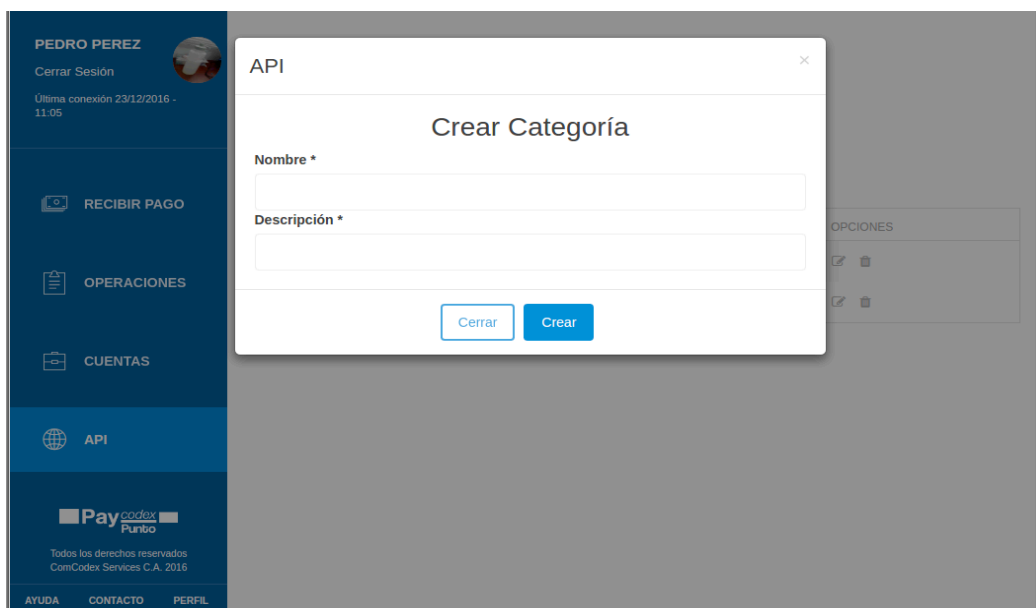
1. [Registrar dispositivo.](#)
 - 1.1. [Obtener Token y Llave de dispositivo.](#)
2. [Agregar la API a un proyecto PHP.](#)
3. [Establecer la configuración.](#)
4. [Objeto tipo CodexServiceCliente y conexión](#) al servicio PayCodex.
5. [Obtener el token de la sesión.](#)
6. [Excepciones.](#)
7. [Crear Transacción.](#)
8. [Obtener lista de Transacciones.](#)
9. [Consultar Transacciones.](#)
10. [Obtener imagen QR.](#)
11. [Consultar Una Sola Transacción.](#)
12. [Estados de Transacciones.](#)
13. [Revertir Transacciones.](#)
14. [Cierre de Operaciones.](#)

API PHP

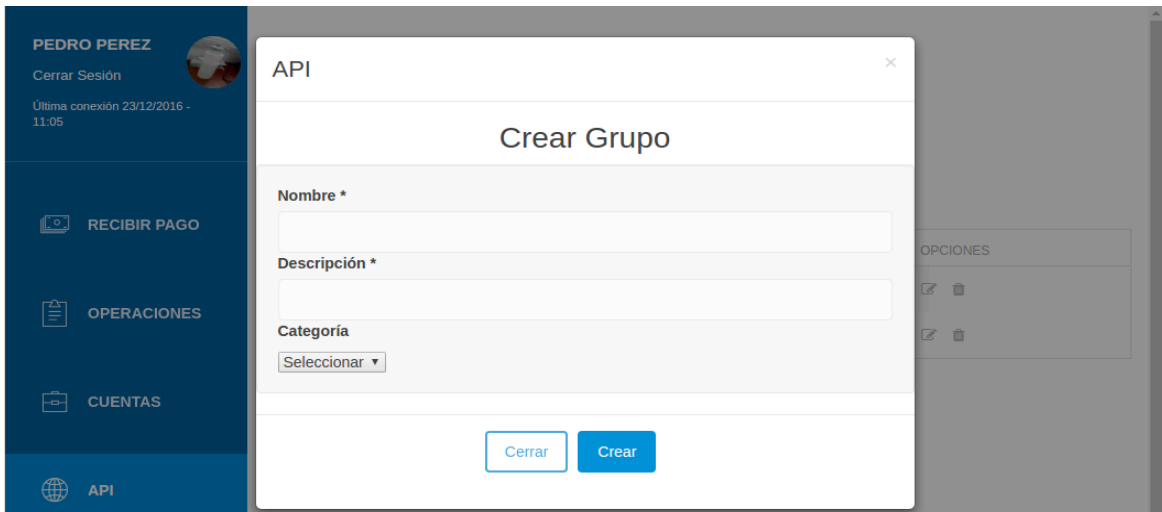
A continuación, se describirán los pasos para el uso de la API de PHP del servicio PayCodex.

1. Registrar dispositivo:

El usuario debe tener una cuenta registrada en **PayCodex Punto**, también debe tener registrado al menos una **categoría** y un **grupo**, estos servirán para poder realizar agrupaciones de los dispositivos que sean registrados.

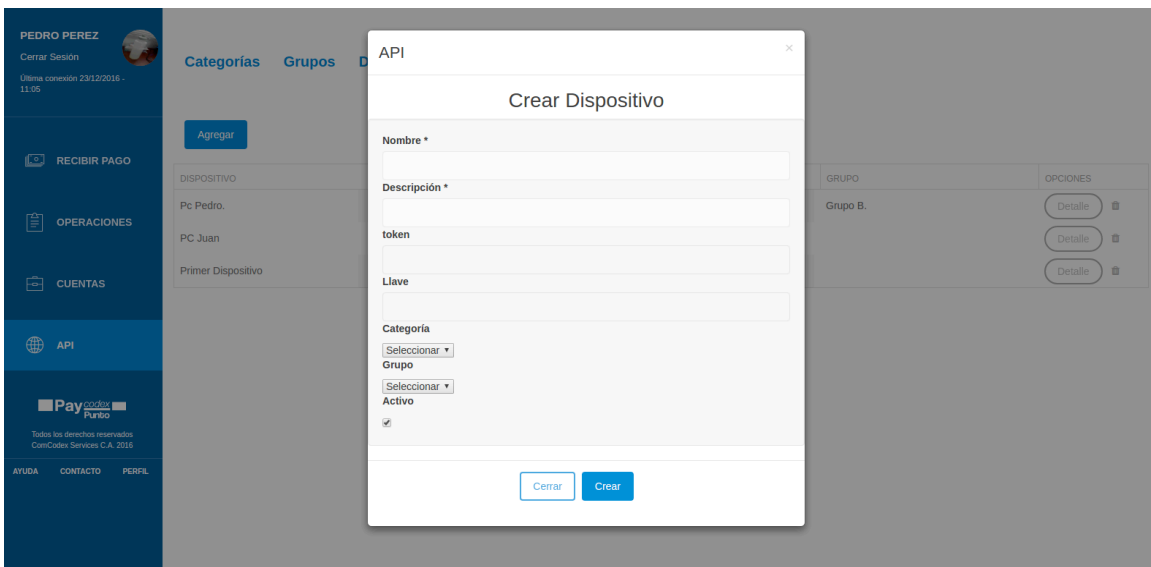


The screenshot displays the PayCodex Punto user interface. On the left is a dark blue sidebar with the user's name 'PEDRO PEREZ', a 'Cerrar Sesión' button, and the last connection time 'Última conexión 23/12/2016 - 11:05'. Below this are menu items: 'RECIBIR PAGO', 'OPERACIONES', 'CUENTAS', and 'API' (which is highlighted). At the bottom of the sidebar is the 'PayCodex Punto' logo and copyright information. The main area shows a modal window titled 'API' with a close button. Inside the modal is a form titled 'Crear Categoría' with two required fields: 'Nombre *' and 'Descripción *'. At the bottom of the modal are 'Cerrar' and 'Crear' buttons. In the background, a partially visible 'OPCIONES' menu is shown.



The screenshot shows a user interface for Pedro Perez. On the left is a sidebar with options: RECIBIR PAGO, OPERACIONES, CUENTAS, and API. The main area displays a modal titled 'API' with the subtitle 'Crear Grupo'. The form includes fields for 'Nombre *', 'Descripción *', and a 'Categoría' dropdown menu. At the bottom are 'Cerrar' and 'Crear' buttons.

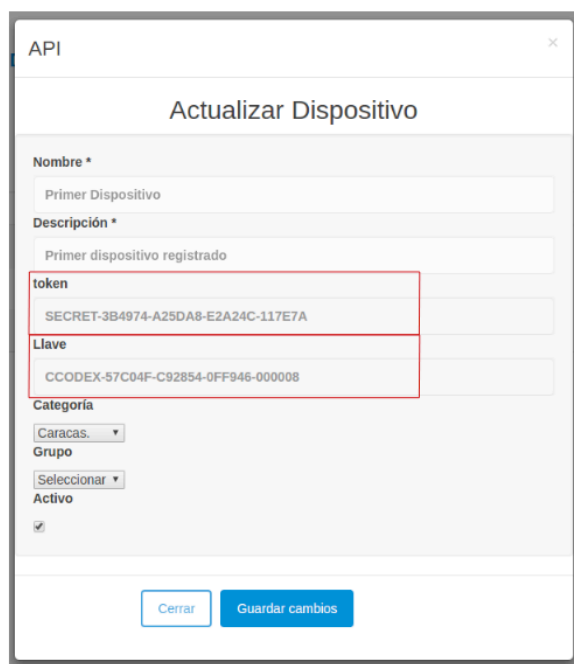
Un usuario puede registrar múltiples **dispositivos**, también puede seleccionar la categoría y/o el grupo al que pertenecerá el nuevo dispositivo. Si solo selecciona un grupo y dicho grupo esta asignado a una categoría entonces el nuevo dispositivo estará automáticamente asignado también a esa categoría.



The screenshot shows the same user interface as before, but with the 'Crear Dispositivo' modal open. The form includes fields for 'Nombre *', 'Descripción *', 'token', and 'Llave'. It also has dropdown menus for 'Categoría', 'Grupo', and 'Activo' (with a checked checkbox). 'Cerrar' and 'Crear' buttons are at the bottom. In the background, a table lists devices: 'Pc Pedro.', 'Pc Juan', and 'Primer Dispositivo'.

1.1. Obtener Token y Llave de dispositivo.

Cuando el usuario registre un dispositivo, le será asignado automáticamente un **token** y una **llave** secreta como se muestra en la siguiente imagen:



The screenshot shows a web form titled "Actualizar Dispositivo" (Update Device) within an "API" context. The form contains several input fields: "Nombre *" (Name) with the value "Primer Dispositivo", "Descripción *" (Description) with the value "Primer dispositivo registrado", "token" with the value "SECRET-3B4974-A25DA8-E2A24C-117E7A", and "Llave" (Key) with the value "CCODEX-57C04F-C92854-0FF946-000008". The "token" and "Llave" fields are highlighted with a red rectangular box. Below these fields are dropdown menus for "Categoría" (set to "Caracas.") and "Grupo" (set to "Seleccionar"), and a checked "Activo" checkbox. At the bottom of the form are two buttons: "Cerrar" (Close) and "Guardar cambios" (Save changes).



2. Agregar la API a un proyecto PHP.

Se debe incluir la clase de la API Codex.inc.php como se indica en el siguiente ejemplo:

```
include '../Codex.inc.php';
```

3. Establecer la configuración.

La conexión con el servicio PayCodex se realiza primero creando la instancia del objeto que tendrá los datos de la configuración del cliente.

```
$setting = new CodexSetting();
```

Luego se asigna los datos de la configuración, como se muestra en el siguiente ejemplo:

```
// Dirección URI del servicio
$setting->serviceUri = "https://api.paycodex.com";

// El puerto
$setting->servicePort = "";

// Clave secreta
$setting->secretPhrase = "SECRET-EB7C35-C247FE-0B1004-3A9851";

// Llave del cliente
$setting->clientKey = "CCODEX-5410B3-99F15C-C40000-000002";

// Dirección URI de las imágenes QR
```



```
$setting->qrServiceUri = "https://paycodex.com/image/qrImage/:token.png";  
  
// Ruta donde se almacenarán las imágenes QR  
$setting->pathImage = "/home/userName/Desktop/comcodex/qr/";
```

4. Objeto tipo CodexServiceCliente y conexión al servicio PayCodex.

Un objeto de tipo **CodexServiceCliente** es el encargado de realizar las operaciones con el servicio de PayCodex, durante la creación de este tipo de objeto se debe pasar como parámetro al objeto de configuración descrito anteriormente, como se indica en el siguiente ejemplo.

```
// $setting contiene los datos de la configuración.  
$client = new CodexServiceClient($setting);
```

Luego se ejecuta el método **connect()** del objeto **\$client** para establecer la conexión.

```
$client->connect();
```


4. Obtener el token de la sesión.

```
$token = $client->getSessionToken();
```

5. Excepciones y el evento de token de sesión invalido:

Las peticiones y consultas al servicio PayCodex, deben estar dentro del manejador de excepciones **try**, además se debe emplearse el **catch** e indicar que se desea capturar una excepción de tipo `CodexServiceClientException`, como se muestra en el siguiente ejemplo:

```
try {  
    // Código  
} catch (CodexServiceClientException $ex) {  
    // Código  
}
```

También, dentro del **catch**, se debe incluir un condicional **if**, para que se verifique si el código de la excepción indica que el token de la sesión ha expirado, en dicho caso, se vuelve ejecutar el método de conexión del objeto, como se muestra en el siguiente ejemplo:

```
try {  
    // Código  
} catch (CodexServiceClientException $ex) {  
    // Si la sesión ha expirado, se ejecuta otra vez la conexión y se continúa  
    // con la ejecución del código que produjo la excepción  
    if ($ex -> getCode() == CodexServiceClientException::ERROR_CODE_SESSION_TOKEN_INVALID) {
```

```
$client -> connect(); // Código  
}  
}
```

6. Crear Transacción:

Primero se crea la instancia de la clase CodexTransaction.

```
$newTransaction = new CodexTransaction();
```

Luego se asigna los datos de la transacción, empezando por la cantidad. Para asignar este parámetro, se debe crear un objeto con la clase CodexDecimal, para que el servicio PayCodex pueda identificar los decimales que estén presente en la cantidad de la transacción, como se muestra en ejemplo:

```
$newTransaction->amount = new CodexDecimal("100000", "32");
```

Luego se asignan los datos que faltan, el dispositivo, y el concepto de la transacción, como se muestra a continuación:

```
$newTransaction->device = "CAJA0100000 YYYY";  
$newTransaction->concept = "COMPRA FACTURA NUEVA YYY 00001";
```

Finalmente, se ejecuta el método **openTransaction**, el cual se encarga de crear la transacción con los datos que se asignaron al objeto **\$newTransaction**.

```
try {  
    // Se crea la nueva transacción  
    $newTransaction = $client->openTransaction( $newTransaction );  
} catch (CodexServiceClientException $ex) {  
    if( $ex->getCode() == CodexServiceClientException::ERROR_CODE_SESSION_TOKEN_INVALID ) {  
        $client->connect();  
        // Se crea la nueva transacción  
        $newTransaction = $client->openTransaction( $newTransaction );  
    }  
}
```

7. Obtener lista de transacciones:

Se instancia el objeto que tendrá los datos que servirán de base para obtener la lista de transacciones:

```
$query = new CodexTransactionQuery();
```

Se asigna las fechas de inicio y fin, para establecer el periodo de tiempo del cual se desea obtener la lista de transacciones.

```
// Fecha de inicio 23/09/2014 12:20:00
$beginDate = new DateTime();
// Establece el dia (23/09/2014)
$beginDate->setDate(2014, 9, 23);
// Establece la hora (12:20:00)
$beginDate->setTime(12, 20, 00);
// Se establece la fecha de inicio.
$query->beginDate = $beginDate;

// Fecha fin 23/09/2014 17:55:00
$endDate = new DateTime();
// Establece el dia (23/09/2014)
$endDate->setDate(2014, 9, 23);
// Establece la hora (17:55:00)
$endDate->setTime(17, 55, 00);
// Se establece la fecha fin.
$query->endDate = $endDate;
```

```
try {

    // Se obtiene la lista de transacciones
    $listTransactions = $client->listTransactions($query);
} catch (CodexServiceClientException $ex) {

    if( $ex->getCode() == CodexServiceClientException::ERROR_CODE_SESSION_TOKEN_INVALID ) {
        $client->connect();
        // Se obtiene la lista de transacciones
        $listTransactions = $client->listTransactions($query);
    }
}
```

8. Consultar Transacciones:

Se obtiene la instancia de cada transacción ubicado en una lista de transacciones (que se obtiene de la forma descrita en el paso anterior), se emplea el bucle **foreach** para crear una instancia de cada transacción contenido en la lista. Con cada instancia de

transacción se acceder a sus respectivos datos, como se muestra en el siguiente ejemplo:

```
foreach( $listTransactions as $transaction ) {  
    // Cantidad  
    echo "Amount: " . $transaction->amount;  
    // Concepto  
    echo "concept: " . $transaction->concept;  
    // Dispositivo  
    echo "device: " . $transaction->device;  
    // id de la transacción  
    echo "id: " . $transaction->id;  
    // Se obtiene la fecha de creación de la transacción  
    if (!empty($transaction->openDate)) {  
        $openDate = new DateTime($transaction->openDate);  
        echo "openDate: " . ( $openDate->format('Y-m-d H:i:s'));  
    }  
    // Dato de tipo Boolean, que indica si el pago fue exitoso  
    echo "payed: " . $transaction->payed;  
    // Obtiene el nombre de la tarjeta  
    echo "payedCardHolder: " . $transaction->payedCardHolder;  
    // Obtiene el número de la tarjeta  
    echo "payedCardNumber: " . $transaction->payedCardNumber;  
    // Se obtiene la fecha cuando culminó la transacción  
    if (!empty($transaction->payedDate)) {  
        $payedDate = new DateTime($transaction->payedDate);  
        echo "payedDate: " . ( $payedDate->format('Y-m-d H:i:s'));  
    }  
    // Número de identidad del pagador  
    echo "payedIdentity: " . $transaction->payedIdentity;  
    // estado de la transacción  
    echo "status: " . $transaction->status;  
    // token de la transacción  
    echo "token: " . $transaction->token;  
}
```

9. Obtener imagen QR:

Primero se crea la instancia de la clase **CodexTransaction()**, luego al atributo **token** de la instancia creada se le asigna el token de la transacción a la cual pertenece el QR que se desea obtener, como se muestra en el ejemplo:

```
// Se crea la instancia
$transaction = new CodexTransaction();

// Se asigna el token de la transacción
$transaction->token = "e44c21fa42cad079629d6a595205974f";

try {

    // Se obtiene la imagen QR
    $client->getQrImage($transaction);
} catch (CodexServiceClientException $ex) {

    if( $ex->getCode() == CodexServiceClientException::ERROR_CODE_SESSION_TOKEN_INVALID ) {

        $client->connect();
        // Se obtiene la imagen QR
        $client->getQrImage($transaction);
    }
}
```

La imagen QR se almacenará en la ruta especificada en la configuración (objeto de la instancia **CodexSetting**). La configuración se explica con detalle en el primer paso explicado para el uso de la API PHP PayCodex.

10. Consultar Una Sola Transacción:

Se puede obtener los datos actualizados de una sola transacción pasando como parámetro el **token** de dicha transacción al método **retrieveTransaction** como se indica en el siguiente ejemplo:

```
$transaction = $client->retrieveTransaction( $transactionToken );
```

11. Estados de transacciones.

Los posibles estados de las transacciones son los siguientes:

- 0: En espera.
- 1: En proceso.
- 2: Pagada.
- 3: Rechazada.
- 4: Fallida.
- 5: Reversado.

12. Revertir Transacciones:

Una transacción pagada se puede revertir empleando el método **revertTransaction**, como se muestra en el siguiente ejemplo:

```
$transactionRevert = $client->revertTransaction($transactionToken);
```

Si la transacción fue revertida exitosamente tendrá el **status** en **5**.

13. Cierre de Operaciones:

El cierre de operaciones se realiza creando un objeto de consulta de tipo **CodexTransactionQuery**, luego a ese objeto se le asigna los parámetros de **fecha** de inicio y fin, como se muestra en el siguiente ejemplo:

```
$query = new CodexTransactionQuery();  
$query->beginDate = new DateTime();  
$query->beginDate->setDate(2016, 06, 08);  
$query->endDate = new DateTime();  
$query->endDate->setDate(2016, 07, 25);  
  
$closingReport = $client->closingReport($query);
```