

# GUÍA DE IMPLEMENTACIÓN

DATAWEB (BOTÓN DE PAGOS)

VERSIÓN 3.2.1  
2022

Fecha	Versión	Descripción	Autor
}	3.0	<p>Nueva versión que incluye:</p> <ol style="list-style-type: none"> <li>1. Nuevos parámetros de envío de información.</li> <li>2. Nuevas validaciones en cálculos de impuestos.</li> <li>3. Mejoras en texto de mensajería de respuesta de las transacciones</li> <li>4. Eliminación de campos SHOPPER_interes / SHOPPER_gracia</li> <li>5. Nuevo método de anulación</li> <li>6. Nuevo parámetro para tipos de crédito.</li> </ol>	Andrés Flores
22/07/2021	3.1	<ol style="list-style-type: none"> <li>1. Nueva mensajería 3DS / OTP como segundo factor de autenticación.</li> <li>2. Actualización de imágenes de ejemplos de código(Anulación / Tokenización / Verificador)</li> </ol>	Andrés Flores
11/10/2021	3.1.1	<p>Se agrega código nuevo de respuesta asociado a la cancelación de transacción de tipo OTP (800.100.152).</p> <p>Se cambió el Título de la sección 3.1.3 por "Obtener el estado de la transacción"</p>	Andrés Flores
11/04/2022	3.2	Cambio de ENDPOINT del Gateway de pagos tanto en producción como en ambiente de prueba por motivos de desactivación del anterior.	Andrés Flores
9/04/2024	3.2.1	Actualización de requerimientos técnicos	Andrés Flores
15/04/2024	3.2.2	Actualización de la descripción del tipo de diferido código 22	Andrés Flores

## Índice

1. Introducción.....	8
2. DATAWEB .....	8
2.1 Front-end.....	8
2.2 Back-end (Tecnologías) .....	8
2.3 Certificados.....	9
2.4 Sobre los ejemplos .....	9
2.5 Sobre los Plugins .....	9
2.6 Sobre las transacciones y monto .....	9
2.7 Esquema de Seguridad .....	9
2.8 Validación de conexión al Gateway .....	10
3. Fases de Integración .....	10
3.1 Fase uno (Prueba Inicial) .....	11
3.1.1 Identificador de Pago (CheckoutId) .....	11
3.1.2 Formulario de Pago.....	12
3.1.3 Obtener el estado de la transacción .....	14
3.2 Fase dos (Prueba Final) .....	16
3.2.1 Nuevos campos .....	16
3.2.1.1 Nombre .....	16
3.2.1.2 Segundo Nombre .....	16
3.2.1.3 Apellido .....	16
3.2.1.4 Dirección IP .....	16
3.2.1.5 Identificador del Cliente en el Comercio .....	17
3.2.1.6 Número de Transacción .....	17
3.2.1.7 Dirección de Correo Electrónico .....	17
3.2.1.8 Tipo de Documento de Identificación del Usuario .....	17
3.2.1.9 Documento de Identificación del Usuario .....	17
3.2.1.10 Datos del producto .....	18
3.2.1.10.1 Nombre .....	18
3.2.1.10.2 Descripción .....	18

3.2.1.10.3 Precio .....	18
3.2.1.10.4 Cantidad .....	18
3.2.1.11. Teléfono del cliente .....	18
3.2.1.12 Dirección de entrega.....	18
3.2.1.13 Dirección del Cliente .....	18
3.2.1.14 País de entrega .....	19
3.2.1.15 País del Cliente.....	19
3.2.1.16 Modo de Prueba.....	19
3.2.1.17 Parámetros Personalizados .....	19
3.2.1.17.1 Datos de Impuestos .....	19
3.2.1.17.2 Datos del comercio (MID / TID).....	20
3.2.1.17.3 Datos de identificación.....	20
3.2.1.17.4 Versión .....	21
3.2.1.18 Risk Parameters .....	21
4 Código de Mensajes.....	24
4.2 Códigos Generales.....	24
4.3 Códigos de Mensajes Detallados.....	30
4.3.1 Aprobación de la transacción (Banco adquirente).....	30
4.3.2 Rechazo de la Transacción (Respuestas del Banco).....	31
5 Opciones Avanzadas.....	33
5.1 Diferidos .....	33
5.2 Tipo de Crédito.....	36
5.3 Validación campos vacíos.....	38
5.6 Estilo del formulario .....	39
5.7 Optimización Móvil .....	39
5.7.1 Habilitar el diseño responsivo en el HTML .....	39
5.7.2 Mobile First (CSS) .....	39
5.7.3 Botones y Campos de texto (CSS).....	40
6 OneclickCheckout (Pago con un click) .....	41
6.2 Preparación del pago (primera compra).....	41
6.3 Eliminación del Token .....	42
7 Anulaciones .....	43

8	Otros métodos de integración.....	45
9	Personalización .....	45
10	Verificador de transacciones.....	46
9.1	Verificar por identificador de la transacción .....	46
9.2	Verificar por el número de transacción del comercio.....	47
11	Paso a Producción (Cronograma <i>estimado</i> ) .....	48
ANEXOS .....		50
Anexo A.1 Php .....		50
Anexo A.2 .....		51
Anexo B.1 Java.....		52
Anexo B.2 .....		53
Anexo C.1 C# .....		54
Anexo C.2 .....		55
Anexo D.1 Python.....		56
Anexo D.2 .....		57
Anexo E.1 Node.js .....		58
Anexo E.2.....		59
Anexo F.1 VB.net.....		61
Anexo F.2.....		62
Anexo G.....		63
Anexo H (Requerimientos Técnicos) .....		64
Anexo I.....		65

## Figuras

FIGURA 1. ESQUEMA DE SEGURIDAD	10
FIGURA 2. EJEMPLO EN PHP DEL PRIMER MÉTODO (PARA OBTENER EL ID)	12
FIGURA 3. RESPUESTA EN FORMATO JSON (ID)	12
FIGURA 4. SCRIPT DE FORMULARIO DE PAGO.	13
FIGURA 5. CÓDIGO HTML DE INVOCACIÓN DEL FORMULARIO DE PAGO	13
FIGURA 6. CÓDIGO EJEMPLO DEL FORMULARIO DE PAGO	13
FIGURA 7. FORMULARIO DE PAGO CON DATOS DE PRUEBAS PARA SU USO	14
FIGURA 8. PARÁMETRO RESOURCEPATH OBTENIDO DEL GATEWAY DE PAGOS	14
FIGURA 9. SEGUNDO MÉTODO UTILIZANDO EL PARÁMETRO RESOURCEPATH	15
FIGURA 10. PROCESAMIENTO DE LA TRANSACCIÓN – RESPUESTA FORMATO JSON FASE 1.	15
FIGURA 11 . CÓDIGO EJEMPLO DEL PRIMER MÉTODO QUE HACE LA PETICIÓN DEL ID (CHECKOUTID) FASE 2.	22
FIGURA 12. PARTE DE LA RESPUESTA EN FORMATO JSON EN LA FASE 2	23
FIGURA 13. JAVASCRIPT PARA DIFERIDOS (INSTALLMENTS).	33
FIGURA 14. EJEMPLO DE INTERFAZ PREVIA DE SELECCIÓN DE DIFERIDOS	35
FIGURA 15. CAMPO DE DIFERIDOS EN RESPUESTA DE LA TRANSACCIÓN	35
FIGURA 16. OPCIÓN DE DIFERIDOS	35
FIGURA 17. OPCIÓN TIPO DE CRÉDITO	36
FIGURA 18. CÓDIGO JAVASCRIPT DE LA IMPLEMENTACIÓN DE DIFERIDOS Y TIPOS DE CRÉDITO	37
FIGURA 19. FORMULARIO CON LAS OPCIONES DE DIFERIDOS Y TIPOS DE CRÉDITO	38
FIGURA 20. PARTE DE LA RESPUESTA DE UNA TRANSACCIÓN CON TIPO DE CRÉDITO.	38
FIGURA 21. CÓDIGO PARA VALIDACIÓN DEL CAMPO CARDHOLDER (NOMBRE DEL DUEÑO DE LA TARJETA).	39
FIGURA 22. ESTILO PLANO DEL FORMULARIO DE PAGO	39
FIGURA 23. FORMULARIO ADAPTADO PARA APLICACIONES MÓVILES	40
FIGURA 24. CÓDIGO PARA ACTIVAR LA CAPTURA (TOKENIZACIÓN) DE LOS DATOS DE LA TARJETA DEL CLIENTE.	41
FIGURA 25. FORMULARIO INCLUYENDO LA OPCIÓN DE CAPTURA DE DATOS.	41
FIGURA 26. FORMULARIO DE PAGO ONECLICKCHECKOUT (PAGO CON UN CLICK)	42
FIGURA 27. CAMBIO DE REGLA CSS PARA LOS DIFERIDOS Y TIPOS DE CRÉDITO ONECLICKCHECKOUT	42
FIGURA 28. FUNCIÓN DE ELIMINACIÓN DEL TOKEN.	43
FIGURA 29. JSON DE RESPUESTA AL EJECUTAR LA FUNCIÓN DE ELIMINACIÓN.	43
FIGURA 30. MENSAJE DE ERROR EN LA TRANSACCIÓN YA QUE NO EXISTE EL TOKEN.	43
FIGURA 31. PARTE DEL CONTENIDO DEL JSON EN UNA TRANSACCIÓN APROBADA	44
FIGURA 32. MÉTODO DE ANULACIÓN	44
FIGURA 33. RESPUESTA DE UNA ANULACIÓN EN FORMATO JSON	45
FIGURA 34. IMAGEN DE REFERENCIA EN LA PERSONALIZACIÓN DE REGLAS DE ESTILOS.	46
FIGURA 35. FUNCIÓN PARA VERIFICAR Y OBTENER SI UNA TRANSACCIÓN FUE PROCESADA (ACEPTADA/RECHAZADA)	46
FIGURA 36. MUESTRA PARCIAL DEL JSON DE RESPUESTA DONDE ENCONTRARÁN DATOS IMPORTANTES	47
FIGURA 37. EJEMPLO DE FUNCIONAMIENTO EN UN PORTAL WEB	47
FIGURA 38. FUNCIÓN PARA VERIFICAR Y OBTENER SI UNA TRANSACCIÓN FUE PROCESADA (ACEPTADA/RECHAZADA) MEDIANTE EL MERCHANTTRANSACTIONID	47

Tablas

TABLA 1. PARÁMETROS GENERALES ..... 11

TABLA 2. PARÁMETROS DE LA FASE 1 ..... 12

TABLA 3. EJEMPLO DE UNA COMPRA CON IMPUESTOS ..... 20

TABLA 4. DESCRIPCIÓN DE DATOS IMPORTANTES ..... 23

TABLA 5. CÓDIGOS GENERALES DE MENSAJES TRANSACCIONALES EN PRODUCCIÓN ..... 24

TABLA 6. CÓDIGO DE MENSAJES GENERALES EN PRUEBAS ..... 29

TABLA 7. CÓDIGOS DE RESPUESTAS EN LA CREACIÓN DEL FORMULARIO (1RA. FUNCIÓN)..... 29

TABLA 8. BANCO ADQUIRENTE ..... 30

TABLA 9. AZ TARJETAS DE CRÉDITO. .... 30

TABLA 10. CÓDIGO DE RESPUESTA EMITIDA POR EL BANCO ..... 31

TABLA 11. TIPOS DE CRÉDITO ..... 36

TABLA 12. PASOS PARA LA SALIDA A PRODUCCIÓN..... 48

## 1. Introducción

*Dataweb* les permite integrar su negocio al e-commerce mediante la implementación de un botón de pagos en su sitio web. Con este producto, podrá ofrecerles a sus clientes la mejor experiencia al momento de realizar sus pagos en compras en línea con todas las tarjetas bancarias.

Todo lo expuesto en el documento es lo que se debe integrar en sus portales web, los campos **son mandatorios**, el código fuente adjunto es **referencial** y sirve como guía, finalmente la implementación del botón es responsabilidad del comercio y depende de la experiencia de la persona que hayan designado como implementador.

Si existe algún problema con la obtención de un dato del documento entonces tienen que comunicarlo vía correo electrónico al personal técnico que le dará soporte en la integración, una vez puesto en producción cualquier consulta o soporte debe ser mediante la cuenta de correo electrónico de servicio **servddpago@datafast.com.ec**

Antes de pasar a producción, Datafast S.A. deberá certificar el desarrollo aplicado por el comercio, por favor revisar la *tabla 12* donde se muestran los tiempos estimados a este proceso.

## 2. DATAWEB

Es una solución de forma de pago compuesta por una API y un widget (Con certificación PCI DSS). Permite la integración en los diferentes portales e-commerce ya sean de desarrollo propio o por medio de CMS, de aquí en adelante se lo describirá como *botón de pagos*.

### 2.1 Front-end

Al ser una solución web, su estructura está definida por las tecnologías:

- HTML
- CSS
- JavaScript

Para la recepción de datos se utiliza el formato **JSON**

### 2.2 Back-end (Tecnologías)

El botón de pagos se adapta a las nuevas tecnológica y los lenguajes de programación soportados son:

- Python
- Java
- Php
- Groovy



- VB.Net
- Ruby
- NodeJs
- C#
- Scala

## 2.3 Certificados

El botón de pagos utiliza **certificado TLS 1.2** con encriptación SHA-2 (256 bits) por lo que su sitio web debe poseer el mismo tanto en ambientes de pruebas y producción.

Cualquier otro certificado inferior al mencionado en el párrafo anterior debe ser deshabilitado (TLS 1.1 / 1.0, SSL2, SSL3, etc.), caso contrario debe comunicarlo de inmediato. Para conocer todos los requerimientos técnicos revisar el [Anexo H](#).

## 2.4 Sobre los ejemplos

En este documento encontrarán ejemplos de código basado en el lenguaje **PHP** pero esto *no indica* que es la manera de hacerlo ya que cada desarrollador debe aplicar los estándares y tecnología de plataforma a implementar el botón de pagos.

## 2.5 Sobre los Plugins

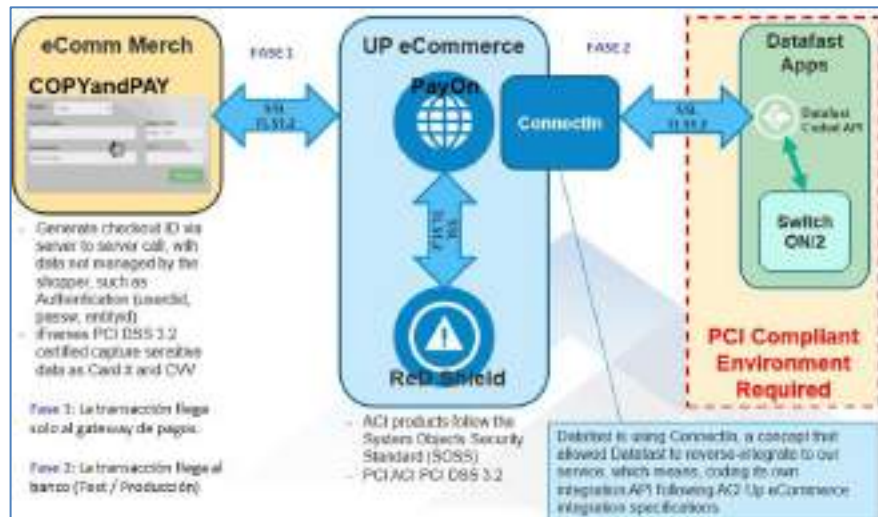
Si su portal web está construido sobre CMS tales como Wordpress + Woocommerce, Prestashop o Magento entonces puede solicitarlo al ejecutivo comercial de Datafast con el cual mantuvo contacto.

## 2.6 Sobre las transacciones y monto

Cuando se está en la **fase dos** (Fase final) el monto de las transacciones no debe superar los **\$50** caso contrario se consumirá el cupo rápidamente de las tarjetas de prueba. Cuando se realice el paso a producción, el comercio deberá tener a la mano una tarjeta de crédito real y un set de datos ficticios de ser necesario, esto ya queda a criterio del comercio.

## 2.7 Esquema de Seguridad

Cada transacción pasa por un proceso de validación y verificación antes de ser procesada, el siguiente esquema muestra el flujo de este. Al salir la transacción del portal pasa por dos sistemas de validaciones para finalmente llegar a los servidores de Datafast.



Descripción de la imagen: El diagrama y el texto describen el proceso de integración de pagos para un comercio electrónico, detallando la seguridad y el flujo de transacciones:

Fase 1: Gateway de Pagos (eComm Merch & COPYandPAY):

La transacción inicial llega al gateway de pagos, donde se genera un ID de checkout a través de una llamada de servidor a servidor, manejando datos sensibles como la autenticación y utilizando iFrames certificados PCI DSS 3.2 para capturar información de la tarjeta.

Fase 2: Procesamiento y Conexión Bancaria (UP eCommerce & PayOn):

La transacción se dirige al banco (en entorno de prueba o producción), utilizando SSL TLS1.2 para la comunicación segura entre los componentes de PayOn y Datafast.

Cumplimiento PCI y Seguridad (ReD Shield & PCI Compliant Environment Required):

Los productos de ACI siguen el estándar de seguridad SOSS y PCI ACI PCI DSS 3.2, y se requiere un entorno compatible con PCI para la integración.

Integración Datafast (Connectin & Datafast Coded API):

Datafast utiliza Connectin para integrar su propio servicio, lo que implica codificar su propia API de integración siguiendo las especificaciones de integración de ACI Up eCommerce.

Figura 1. Esquema de seguridad

## 2.8 Validación de conexión al Gateway

Antes de empezar el desarrollo se debe comprobar que el servidor/terminal en donde se implementará el botón de pagos tenga conectividad a nuestro gateway de pagos.

Para realizar la comprobación debe ejecutarse en una terminal dentro del **servidor** este fragmento de código. [Anexo G](#)

```
C:\Windows\system32\cmd.exe
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\andre>curl https://eu-test.oppwa.com/v1/checkouts -d "entityId=8a829418533cf31d01533d06f2ee06fa" -d "amount=92.00" -d "currency=USD" -d "paymentType=DB" -H "Authorization: Bearer OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjIyUUVOWA=="

Descripción de la imagen= Consola: C:\Windows\system32\cmd.exe
(c) Microsoft Corporation. Todos los derechos reservados.
"entityId=8a829418533cf31d0153"
```

```
C:\Users\andre>curl https://eu-test.oppwa.com/v1/checkouts 3d06f2ee06fa" -d "amount=92.00" -d "currency USD" -d "payment Type=DB" -H "Authorization: Bearer OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjIyUUVOWA=="
```

Deberá obtener un resultado como se muestra a continuación.

```
{"result":{"code":"000.200.100","description":"successfully created checkout"},"buildNumber":"0e531a0327bd64cc117eabe6a098c851b0c69050@2022-04-11 07:05:27 +0000","timestamp":"2022-04-11 16:41:13+0000","ndc":"F37D8E9DDCED2FA25A1D7EC2EF7372E0.uat01-vm-tx04","id":"F37D8E9DDCED2FA25A1D7EC2EF7372E0.uat01-vm-tx04"}
```

Descripción: Respuesta enCMD:

```
{"result":{"code":"000.200.100","description":"successfully created checkout"},"buildNumber":"0e531a0327bd64cc117eabe6a098c851b0c69050@2022-04-11 07:05:27 +0000","timestamp":"2022-04-11 16:41:13+0000","ndc":"F37D8E9DDCED2FA25A1D7EC2EF7372E0.uat01-vm-tx04","id":"F37D8E9DDCED2FA25A1D7EC2EF7372E0.uat01-vm-tx04"}
```

Puede que se tenga que activar el módulo CURL para esta prueba.

### 3. Fases de Integración

Para la integración de la solución se ha dividido en **dos** etapas de pruebas antes del paso al ambiente de producción.

En ambos casos se debe incluir la siguiente línea de Javascript después del tag de cierre </body>

```
<script type="text/javascript" src="https://www.datafast.com.ec/js/dfAdditionalValidations1.js">
```

### 3.1 Fase uno (Prueba Inicial)

En esta fase el comercio o establecimiento a implementar el botón podrá tener una primera experiencia al interactuar con el Gateway de pagos.

Se debe tener parametrizado los campos que varían cuando se cambien de ambientes, es decir entre **prueba** y **producción**.

Tabla 1. Parámetros Generales

	Descripción
<b>url</b>	Indica el ambiente en el cual se está trabajando (Prueba / Producción)
<b>mid</b>	Indica el identificador del comercio
<b>tid</b>	Indica el identificador del terminal
<b>textMode</b>	Este parámetro en ambiente de producción no se incluye en el método, pero en pruebas si es necesaria (Fase 2) tal como se verá en los ejemplos.
<b>entityId</b>	Identificador de entidad
<b>Authorization</b>	Parámetro de cabecera para autenticación de tipo <i>Bearer</i>

En la versión anterior eran mandatorios los parámetros `userId` y `password`, estos fueron reemplazados por el *autorizador* (*Authorization*).

#### 3.1.1 Identificador de Pago (CheckoutId)

En esta primera instancia se envían los parámetros básicos hacia la plataforma de pago, para esta guía se utilizará el **lenguaje PHP** (Anexo [A.1](#)). La respuesta a una solicitud exitosa es una cadena JSON con un **id** que se requiere en el segundo paso para crear el formulario de pago. Si usted utiliza otro lenguaje de por ejemplo Java puede obtener el código en el Anexo [B.1](#), para C# el Anexo [C.1](#), para los otros lenguajes que soporta nuestro API ([2.2](#)) revisar los anexos.

Los parámetros en esta sección son los que se describen en la tabla 2.

Tabla 2. Parámetros de la fase 1

Parámetro	Descripción
<b>url</b>	<i>https://eu-test.oppwa.com</i>
<b>entityId</b>	<i>8a829418533cf31d01533d06f2ee06fa</i>
<b>amount</b>	Monto de la transacción, ejemplo: 5.00
<b>currency</b>	El valor es <b>USD</b>
<b>paymentType</b>	Para compras el valor es <b>DB</b>
<b>Authorization</b>	Bearer <i>OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUUVOWA==</i>

```
function request() {
    $url = "https://eu-test.oppwa.com/v1/checkouts";
    $data = "entityId=8a8294174b7ecb28014b9699220015ca" .
        "&amount=92.00" .
        "&currency=USD" .
        "&paymentType=DB";

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Authorization:Bearer OGE4Mjk0MTc0Yjd1Y2IyODAxNGI5Njk5MjIwMDE1Y2N8c3k2S0pzVDg='));
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $responseData = curl_exec($ch);
    if(curl_errno($ch)) {
        return curl_error($ch);
    }
    curl_close($ch);
    return $responseData;
}
```

Transcripción de la imagen:

```
function request() {
    $url "https://eu-test.oppwa.com/v1/checkouts";
    $data "entityId=8a8294174b7ecb28014b9699220015ca"
    "&amount=92.00".
    "&currency=USD"
    "&paymentType=DB";
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    "Authorization: Bearer
    OGE4Mjk0MTc0Yjd1Y2IyODAxNGI5Njk5MjIwMDE1Y2N8c3k2S0pzVDg='));
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in
    production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $responseData = curl_exec($ch);
    if(curl_errno($ch)) {
    }
    return curl_error($ch);
    }
    curl_close($ch);
    return $responseData;
}
```

Figura 2. Ejemplo en PHP del primer método (Para obtener el id)

Se ejecuta el método y se obtiene la respuesta en formato JSON

```
{
  "result":{
    "code":"000.200.100",
    "description":"successfully created checkout"
  },
  "buildnumber":"0e531a0327bd64cc117eabe6a098c851b0c69050@2022-04-11 07:05:27 +0000",
  "timestamp":"2022-04-11 10:37:16+0000",
  "ndc":"54E3A6A1958447ADA8E9B14EE5EDC4E3.uat01-vm-tx03",
  "id":"54E3A6A1958447ADA8E9B14EE5EDC4E3.uat01-vm-tx03"
}
```

Transcripción:

```
{
  "result":{
    "code": "000.200.100",
    "description": "successfully created checkout"
  },
  "buildNumber": "0e531a0327bd64cc117eabe6a098c851b0c69050@2022-04-11 07:05:27 +0000",
  "timestamp": "2022-04-11 19:37:16+0000",
  "ndc": "54E3A6A1958447ADA8E9B14EE5EDC4E3.uat01-vm-tx03",
  "id": "54E3A6A1958447ADA8E9B14EE5EDC4E3.uat01-vm-tx03"
}
```

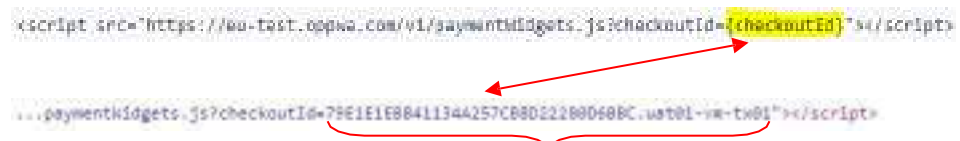
Figura 3. Respuesta en formato JSON (id)

### 3.1.2 Formulario de Pago

Para la generación del formulario de pago se debe utilizar el **id** (checkoutId) obtenido en el paso anterior. En la página donde

ubicarán el formulario de pago deben agregar el código Javascript que se muestra a continuación junto al checkoutId. Cabe mencionar que este checkoutId se obtiene cada vez que se ejecute el método y tiene una caducidad de 30 minutos.

```
<script src="https://eu-eu-test.oppwa.com/v1/paymentWidgets.js?checkoutId={checkoutId}"></script>
```



```
<script src="https://eu-test.oppwa.com/v1/paymentWidgets.js?checkoutId={checkoutId}"></script>  
...paymentwidgets.js?checkoutId=79E1E1E6841134A257CB8D22280068BC.uat01-vm-tx01"></script>
```

Transcripción de imagen:

```
<script src="https://eu-test.oppwa.com/v1/paymentWidgets.js?checkoutId={checkoutId}"></script>  
... paymentwidgets.js?checkoutId=79E1E1E6841134A257CB8D22280068BC.uat01-vm-tx01"></script>
```

Figura 4. Script de Formulario de Pago.

Una vez agregado el script se procede a invocar el formulario de pago insertando el código que muestra la figura 5, donde **shopperResultURL** es la dirección a donde se redirigirá la página una vez que haya presionado el botón “Pagar” ya que el Gateway le enviará un parámetro necesario para el siguiente paso, por otro lado, en la propiedad *data-brands* se puede configurar las marcas de las tarjetas que se utilizaran, siendo estas **VISA MASTER AMEX DINERS DISCOVER**, esto también puede ser parametrizado si así lo creen conveniente. Para tarjetas **ALIA** primero deberá solicitarlo al departamento comercial del banco emisor y luego Datafast le dará las instrucciones técnicas.

```
<form action="{shopperResultUrl}" class="paymentWidgets" data-brands="VISA MASTER AMEX DINERS DISCOVER"></form>
```

Figura 5. Código HTML de invocación del Formulario de Pago

Transcripción: <form action=" (shopperResultUrl)" class="paymentWidgets" data-brands-"VISA MASTER AMEX DINERS DISCOVER"></form>

A continuación, un ejemplo utilizando el lenguaje Php.

```
<script src="<?php echo $url ?>paymentWidgets.js?checkoutId=<?php echo $json['id'] ?>"></script>  
<form action="<?php echo $baseUrl ?>" class="paymentWidgets" data-brands="VISA MASTER DINERS DISCOVER AMEX">  
</form>
```

Transcripcion: <script src="<?php echo \$url  
>paymentWidgets.js?checkoutId=<?php echo \$json['id'] ?>"></script>

```
<form action="<?php echo $baseUrl ?>" class="paymentWidgets" data-  
brands "VISA MASTER DINERS DISCOVER AMEX">
```

```
</form>
```

Figura 6. Código ejemplo del Formulario de Pago

Una vez configurado, el formulario se construye y se comporta de la siguiente manera.



Descripción interfaz de Datafast tarjeta VISA

Figura 7. Formulario de Pago con datos de pruebas para su uso

Para esta fase puede utilizar los datos de tarjeta de crédito que se muestran en la figura 7. Por asuntos de revisión, en el campo *cardholder* se debe poner el nombre del comercio.

Una vez presionado el botón pagar, los datos vuelven a viajar al gateway y nos devuelve un parámetro a la página seleccionada en el parámetro ***shopperResultURL***.

En esta página usted podrá obtener el parámetro ***resourcePath*** cuyo contenido es: `/v1/checkouts/{id}/payment` tal como lo muestra la figura 8.

```
resourcePath=/v1/checkouts/79E1E1EBB41134A257CB8D22280D6B8C.uat01-vm-tx01/payment
```

Transcripción: `resourcePath=/v1/checkouts/79E1E1EBB41134A257CB8D22280D6B8C.uat01-vm-tx01/payment`

Figura 8. Parámetro *resourcePath* obtenido del Gateway de pagos

Con esto puede continuar con el paso 3 (Procesamiento de la compra y obtención de los datos de la transacción)

### 3.1.3 Obtener el estado de la transacción

Cuando reciben el parámetro *resourcePath* se procede a crear el método que se muestra en la figura 9 para obtener los datos de respuesta de la transacción (Anexo [A.2](#) / [B.2](#) / [C.2](#)). Cabe recalcar que el valor del parámetro obtenido se concatena a la Url de prueba/producción.

```

function request() {
    $url = "https://eu-test.oppwa.com/v1/checkouts/{id}/payment";
    $url .= "?entityId=8a8294174b7ecb28014b0690220015ca";

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Authorization:Bearer OGE4Mjk0MTc0Yjd1Y2IyODAXNGI5Njk5MjIwMDE1Y2N8c3k2S0p1VDg='));
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $responseData = curl_exec($ch);
    if(curl_errno($ch)) {
        return curl_error($ch);
    }
    curl_close($ch);
    return $responseData;
}

```

Transcripción:

```

function request() {
    $url = "https://eu-test.oppwa.com/v1/checkouts/{id}/payment";
    $url .= "?entityId=8a8294174b7ecb2801469699220015ca";
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        "Authorization: Bearer
        OGE4Mjk0MTc0Yjd1Y2IyODAXNGI5Njk5MjIwMDE1Y2N8c3k2S0p1VDg="));
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to
    true in production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $responseData = curl_exec($ch);
    if (curl_errno($ch)) {
        return curl_error($ch);
    }
    curl_close($ch);
    return $responseData;
}

```

Figura 9. Segundo método utilizando el parámetro resourcePath

Una vez ejecutado este método la plataforma envía la respuesta en formato de texto JSON hacia el cliente.

```
{
  "id": "8a82944a5ec84a95015ee46612620fbd",
  "paymentType": "DB",
  "paymentBrand": "VISA",
  "amount": "92.00",
  "currency": "USD",
  "descriptor": "7985.7566.9922 OPP_Channel",
  "result": {
    "code": "000.100.110",
    "description": "Request successfully processed in 'Merchant in Integrator Test Mode'"
  },
  "resultDetails": [
    {
      "clearingInstituteName": "Acceptance Test"
    }
  ],
  "card": {
    "bin": "420000",
    "last4Digits": "eeee",
    "holder": "Juan Ruiz",
    "expiryMonth": "01",
    "expiryYear": "2020"
  },
  "threeDSecure": {
    "eci": "e7"
  },
  "customParameters": {
    "CTPE_DESCRIPTOR_TEMPLATE": ""
  },
  "risk": {
    "score": "100"
  },
  "buildNumber": "0602a109f824c84b382349a08fb4a8cbaf1ba67b@2017-10-02 14:26:32 +0000",
  "timestamp": "2017-10-03 22:48:48+0000",
  "ndc": "86077C1622E0F52951849062816C6121.sbg-vm-tx02"
}
```

Transcripción:

```
{
  "id": "8a82944a5ec84a95015ee46612620fbd",
  "paymentType": "DB",
  "paymentBrand": "VISA",
  "amount": "92.00",
  "currency": "USD",
  "descriptor": "7985.7566.9922 OPP_Channel",
  "result": {
    "code": "000.100.110",
    "description": "Request successfully processed in 'Merchant in Integrator Test Mode'"
  },
  "resultDetails": {
    "clearingInstituteName": "Acceptance Test" },
  "card": {
    "bin": "420000",
    "last4Digits": "eeee",
    "holder": "Juan Ruiz",
    "expiryMonth": "01",
    "expiryYear": "2020"
  },
  "threeDSecure": {
    "eci": "e7"
  },
  "customParameters": {
    "CTPE_DESCRIPTOR_TEMPLATE": ""
  },
  "risk": {
    "score": "100"
  },
  "buildNumber": "0602a109f824c84b382349a08fb4a8cbaf1ba67b@2017-10-02 14:26:32 +0000",
  "timestamp": "2017-10-03 22:48:48+0000",
  "ndc": "86077C1622E0F52951849062816C6121.sbg-vm-tx02"
}
```

Figura 10. Procesamiento de la transacción – Respuesta formato JSON fase 1.

Si por algún motivo no se pueda obtener el estado de la transacción de manera sincrónica entonces puede usar el [Verificador de Transacciones](#).

## Fase dos (Prueba Final)

Una vez integrado el botón en su fase inicial de pruebas procederemos a realizar algunos cambios en el método inicial para poder así realizar transacciones de pruebas hacia el switch transaccional de Datafast en ambiente de desarrollo. En la fase uno las transacciones no llegan a Datafast.

Los parámetros de autenticación en esta fase son proporcionados por **Datafast** previa solicitud vía correo electrónico, adicional se otorga una tarjeta de crédito de prueba.

- entityId
- Authorization (Bearer)

### 3.1.4 Nuevos campos

Se procederá describir campo por campo lo que se necesita agregar como parámetros en el primer método del API. Estos campos son obligatorios y no se puede poner valores por defecto como “SN”, “NA”, “0999999999”, etc. Si tiene problemas para obtener desde su sistema algún campo por favor indicarlo mediante correo electrónico al departamento de riesgos mediante [servbdpago@datafast.com.ec](mailto:servbdpago@datafast.com.ec).

#### 3.1.4.1 Nombre

Este campo se refiere al nombre del usuario que está realizando la transacción, el mismo será asignado a un campo disponible de la plataforma.

**customer.givenName** | AlfNum {3,48}

#### 3.1.4.2 Segundo Nombre

Este campo se refiere al segundo nombre del usuario que está realizando la transacción, el mismo será asignado a un campo disponible de la plataforma.

**customer.middleName** | AlfNum {2,50}

#### 3.1.4.3 Apellido

Este campo se refiere al apellido del usuario que está realizando la transacción, el mismo será asignado a un campo disponible de la plataforma.

**customer.surname** | AlfNum {3,48}

#### 3.1.4.4 Dirección IP

Este campo se refiere a la dirección ip del equipo desde donde el usuario está realizando la transacción, el mismo

será asignado a un campo disponible de la plataforma, no debe enviar la IP de su servidor.

**customer.ip** | AlfNum 255{1,255}

#### ***3.1.4.5 Identificador del Cliente en el Comercio***

Este campo se refiere a la identificación (id único) del usuario registrado en el sistema, el mismo será asignado a un campo disponible de la plataforma.

**customer.merchantCustomerId** | AlfNum 16{1,16}

#### ***3.1.4.6 Número de Transacción***

Este campo se refiere a la identificación (id único) de la transacción (trx) registrada en el sistema, el mismo será asignado a un campo disponible de la plataforma, cabe mencionar que este identificador debe ser único por transacción así sea rechazada en su primer intento.

**merchantTransactionId** | AlfNum 255{8,255}

#### ***3.1.4.7 Dirección de Correo Electrónico***

Este campo se refiere a la dirección de correo electrónico asociada a un usuario del sistema, el mismo será asignado a un campo disponible de la plataforma.

**customer.email** | AlfNum 128{6,128}

#### ***3.1.4.8 Tipo de Documento de Identificación del Usuario***

Este campo se refiere al tipo de documento de identidad del usuario en sistema (DNI, Cédula, pasaporte, etc.), es una palabra reservada del gateway.

**customer.identificationDocType** en este caso el valor siempre será **IDCARD**

#### ***3.1.4.9 Documento de Identificación del Usuario***

Este campo se refiere al documento de identidad del usuario en sistema (DNI, Cédula, pasaporte, etc.), el mismo será asignado a un campo disponible de la plataforma. La longitud del campo es siempre de 10 caracteres y debe coexistir con el anterior campo caso contrario le emitirá error. En el caso de usar RUC se debe cortar y solo enviar la longitud indicada, internamente en su sistema puede guardar el dato completo, si el dato es menor a la longitud

requerida entonces se debe completar con ceros a la izquierda.

**customer.identificationDocId** | AlfNum 10{1,10}

#### ***3.1.4.10 Datos del producto***

En esta sección se describirá los campos necesarios para almacenar la información de los ítems(productos o servicios) involucrados en la compra, cabe aclarar que los parámetros a utilizar estarán agrupados en arreglos de datos simples.

##### ***3.1.4.10.1 Nombre***

Nombre del producto, no se acepta el &

**cart.items[0].name** | AlfNum 255{1,255}

##### ***3.1.4.10.2 Descripción***

Descripción del producto.

**cart.items[0].description** | AlfNum 255{1,255}

##### ***3.1.4.10.3 Precio***

Precio del producto

**cart.items[0].price** | Num 13 {1,10}.{2}

##### ***3.1.4.10.4 Cantidad***

Cantidad del producto

**cart.items[0]. quantity** | Num 5 {1,12}.{0,3}

#### ***3.2.1.11. Teléfono del cliente***

Número de teléfono, de formato alfanumérico y longitud máxima de 25 caracteres.

**customer.phone** | AlfNum 25 {7,25}

#### ***3.2.1.12 Dirección de entrega***

Dirección de entrega de la compra del cliente, de formato alfanumérico y longitud máxima de 100 caracteres.

**shipping.street1** | AlfNum 100 {1,100}

#### ***3.2.1.13 Dirección del Cliente***

Dirección del cliente que realice la compra (generalmente el de la factura), de formato alfanumérico y longitud máxima de 100 caracteres.z

**billing.street1** | AlfNum 100{1,100}

#### **3.2.1.14 País de entrega**

País de entrega de compra que realiza el cliente, de formato alfabético y longitud de 2 caracteres en mayúscula. Ejm (EC, CL, US, etc. Formato ISO 3166-1)

**shipping.country** | Alf 2{2}

#### **3.2.1.15 País del Cliente**

País de compra que realiza el cliente (generalmente de la factura), de formato alfabético y longitud de 2 caracteres en mayúscula. Ejm (EC, CL,US, etc. Formato ISO 3166-1)

**billing.country** | Alf 2{2}

#### **3.2.1.16 Modo de Prueba**

Le indica a la plataforma en qué modo se están haciendo las transacciones.

**testMode**

Toma el valor de EXTERNAL. En producción este parámetro tiene que ser eliminado completamente.

#### **3.2.1.17 Parámetros Personalizados**

Los parámetros personalizados sirven para el envío de datos adicionales importantes en la transacción

##### **3.2.1.17.1 Datos de Impuestos**

Toda transacción debe tener obligatoriamente información relacionada con los impuestos, si acaso no utiliza alguno de ellos igual deben enviarlo pero con valores en cero. El formato para estos campos es #####.## (2 decimales, sin separador de miles)

###### **6.2.1.17.1.1 % Base 0 (variable)**

Este campo es asignado para aquellos valores que no gravan impuestos.

**customParameters[SHOPPER\_VAL\_BASE0]**

###### **6.2.1.17.1.2 % Base imponible (variable)**

Este campo es asignado para aquellos valores que si gravan impuestos, en este caso el % del IVA.

**customParameters[SHOPPER\_VAL\_BASEIMP]**

#### 6.2.1.17.1.3 Valor de impuesto (variable)

Este campo es el valor calculado a la base imponible, en este caso el valor del IVA.

**customParameters[SHOPPER\_VAL\_IVA]**

#### 6.2.1.17.1.4 Ejemplo de compra

Para una mejor explicación se detalla el siguiente ejemplo.

Una compra por un total de **\$3.12** que se desglosa de la siguiente manera:

**Tabla 3. Ejemplo de una compra con impuestos**

a. % Base imponible	\$1
b. Valor de impuesto	\$0.12
c. Total (a+b)	\$1.12
d. % Base 0	\$2
Total (c+d)	\$3.12 (Monto)

Los campos quedarían así:

**customParameters[SHOPPER\_VAL\_BASE0]=2.00**

**customParameters[SHOPPER\_VAL\_BASEIMP]=1.00**

**customParameters[SHOPPER\_VAL\_IVA]=0.12**

#### 3.2.1.17.2 Datos del comercio (MID / TID)

En esta sección se describen el identificador del comercio (MID) y el identificador del terminal (TID). Para pruebas (fase 2) se tendrán los siguientes valores 1000000406 y PD100406 respectivamente. En producción, Datafast debe proporcionarles otros.

**customParameters[SHOPPER\_MID]= 1000000406**

**customParameters[SHOPPER\_TID]= PD100406**

#### 3.2.1.17.3 Datos de identificación

En esta sección se describen datos necesarios para validación de identificación, estos valores son fijos.

##### 3.2.1.17.3.1 Identificador de seguridad

**customParameters[SHOPPER\_ECI]= 0103910**



#### 3.2.1.17.3.2 Identificador del proveedor

**customParameters[SHOPPER\_PSERV]= 17913101**

#### 3.2.1.17.4 Versión

Para obtener las nuevas características es importante enviar la versión.

**customParameters[SHOPPER\_VERSIONDF]=2**

#### **3.2.1.18 Risk Parameters**

Este campo sirve para identificar al comercio en la transacción, solo se debe poner el nombre de este, también es obligatorio para procesos internos de prevención de fraude.

**risk.parameters[USER\_DATA2] | AlfNum (30)**

**risk.parameters[USER\_DATA2]=SuComercio**

A continuación, un ejemplo de concatenación de campos en el lenguaje PHP, a la función de la figura 2 se le agregan todos estos.

```

$url = "https://eu-test.oppwa.com/v1/checkouts";
$data = "entityId=".$entity.

    "&amount=".$total.
    "&currency=USD".
    "&paymentType=DB".
    "&customer.givenName=".$primer_nombre.
    "&customer.middleName=".$segundo_nombre.
    "&customer.surname=".$apellido.
    "&customer.ip=".$ip_address.
    "&customer.merchantCustomerId=000000000001".
    "&merchantTransactionId=transaction_".$trx.
    "&customer.email=".$email.
    "&customer.identificationDocType=IDCARD".
    "&customer.identificationDocId=".$cedula.
    "&customer.phone=".$telefono.
    "&billing.street1=".$direccion_cliente.
    "&billing.country=".$pais_cliente.
    "&shipping.street1=".$direccion_entrega.
    "&shipping.country=".$pais_entrega.
    "&customParameters[SHOPPER_ECI]=0103910".
    "&customParameters[SHOPPER_PSERV]=17913101".
    "&customParameters[SHOPPER_VAL_BASE0]=$valueTotalBase0.
    "&customParameters[SHOPPER_VAL_BASEIMP]=$valueTotalIva.
    "&customParameters[SHOPPER_VAL_IVA]=$valueIva.
    "&customParameters[SHOPPER_MID]=$merchantId.
    "&customParameters[SHOPPER_TID]=$terminalId.
    "&risk.parameters[USER_DATA2]=DATAFAST".
    "&customParameters[SHOPPER_VERSIONDF]=2".
    "&testMode=EXTERNAL";

foreach ($items["cart"] as $c) {
    $data.= "&cart.items[". $i. "].name=".$c["product_name"];
    $data.= "&cart.items[". $i. "].description=".$c["Description: ".$c["product_name"];
    $data.= "&cart.items[". $i. "].price=".$c["product_price"];
    $data.= "&cart.items[". $i. "].quantity=".$c["q"];
    $i++;
}

```

Transcripción:

```

Url = "https://eu-test.oppwa.com/v1/checkouts";
Sdata "entityId=".$entity.
"&amount=".$total.
"&currency USD".
"&payment Type=DB".
"&customer.givenName=".$primer_nombre.
"&customer.middleName=".$segundo_nombre.
"&customer.surname=".$apellido.
"&customer.ip=".$ip_address.
"&customer.merchantCustomerId=000000000001".
"&merchantTransactionId=transaction_".$trx.
"&customer.email=".$email.
"&customer.identificationDocType IDCARD".
"&customer.identificationDocId=".$cedula.
"&customer.phone=".$telefono.
"&billing.street1=".$direccion_cliente.
"&billing.country=".$pais_cliente.
"&shipping.street1=".$direccion_entrega.
"&shipping.country=".$pais_entrega.
"&customParameters [SHOPPER_ECI]=0103910".
"&customParameters [SHOPPER_PSERV]-17913101".
"&customParameters [SHOPPER_VAL_BASED]=$valueTotalBase0.
"&customParameters [SHOPPER_VAL_BASEIMP]=$valueTotaliva.
"&customParameters [SHOPPER_VAL_IVA]=$valueIva.

```

```

"&customParameters [SHOPPER_MID]=".$merchant Id.
"&customParameters [SHOPPER_TID]=".$terminalId.
"&risk.parameters [USER_DATA2]-DATAFAST".
"&customParameters [SHOPPER_VERSIONDF]=2".
"&testMode-EXTERNAL";
foreach ($items["cart"] as $c) (
Sdata. "&cart.items[". $i. "].name=".$c["product_name"];
Sdata. "&cart.items[". $i. "].descriptions", "Descripcion: ".$c["product name"];
$sd, "&cart.items[". $i. "].price=".$c["product_price"];
Sdata. "&cart.items[". $i. "].quantity=".$c["q"];
$i++;
}

```

**Figura 11 . Código parcial de ejemplo del primer método que hace la petición del id (CheckoutId) fase 2.**

Una vez ejecutado el método se obtendrá la respuesta en formato JSON, la cual se muestra en la figura 12. Como podrá observar en esta respuesta se incluye información de la transacción que puede ser almacenadas en su base de datos para un mejor control, se recomienda que se guarde esta información ya sea en una transacción exitosa como en una fallida, adicional se recomienda que cada transacción cuando se genere sea guardada con un estado (procesada/no procesada) para que esta información pueda usarse en el método de verificador de transacciones que se verá más adelante.

```

Array
(
    [id] => 8ac7a4a072e61a030172e8629eaf2af9
    [paymentType] => DB
    [paymentBrand] => VISA
    [amount] => 1.12
    [currency] => USD
    [descriptor] => 1459.4271.7847 Low Risk Sector
    [merchantTransactionId] => transaction_644
    [result] => Array
        (
            [code] => 000.100.112
            [description] => Request successfully processed in 'Merchant in Connector Test Mode'
        )

    [resultDetails] => Array
        (
            [RequestId] => 358552281200
            [RiskFraudStatusCode] => ACCEPT
            [AuthCode] => 158725
            [ConnectorTxID1] => 8ac7a4a072e61a030172e8629eaf2af9
            [ReferenceNbr] => 200624_000001
            [BatchNo] => 200624
            [EXTERNAL_SYSTEM_LINK] => https://csi-stage.redworldwide.com/index.red#transactiondet
            [TotalAmount] => 1.14
            [OrderId] => 145942717847
            [Response] => 00
            [Interest] => 0.02
            [RiskStatusCode] => PENDING
            [ExtendedDescription] => Transaction succeeded (Transacción Aprobada)
            [clearingInstituteName] => DataFast
            [RiskResponseCode] => 0100
            [AcquirerCode] => 04
            [CardType] => VS
            [AcquirerResponse] => 00_04VS
            [action] => created
            [RiskOrderId] => 000537010000000520200624181117550
            [ReferenceNo] => 000001
        )

    [card] => Array
        (
            [bin] => 454063
            [binCountry] => EC
            [last4Digits] => 0000
            [holder] => test
            [expiryMonth] => 04
            [expiryYear] => 2022
        )
)

```

Transcripción:

Array

(

[id] => 8ac7a4a072e618030172e8629eaf2af9

[payment Type] => 06

[paymentBrand] => VISA

[amount] => 1.12

[currency]> USD

```
[descriptor] > 1459.4271.7847 Low Risk Sector

[merchantTransactionId] => transaction_644

[result] => Array

(

[code]> 000.100.112

) [description] => Request successfully processed in 'Merchant in Connector Test Mode

[resultDetails]->> Array

(

[RequestId] => 350552201200

[RiskFraudStatusCode]> ACCEPT

[AuthCode] => 158725

[ConnectorTx101] > Bac7a4a072e61a03017268629eaf2af9

[ReferenceNbr]> 200624 000081

[Batchio] => 200624

[EXTERNAL SYSTEM LINK] => https://csi-stage.redworldwide.com/index.red#transactiondet

[TotalAmount] => 1.14

[OrderId] => 145942717847

[Response] > 00

[Interest]> 0.02

(RiskStatusCode] => PENDING

[ExtendedDescription] -> Transaction succeeded (Transacción Aprobada)

[clearingInstituteName] -> Datafast

(RiskResponseCode]> 0120

[AcquirerCode] => 04

[CardType] => VG

[AcquirerResponse] > 00 04VG

[action] => created

[RiskOrderId]> 000537010000R0520200624151117550

[Referencelio] -> 000051

)

[card] => Array

(

(bin) > 454063
```

```

[binCountry] > EC
[lest4Digits]> 0000
[holder] Test
[expiryMonth] => 04
[expiryYear] => 2022
)

```

Figura 12. Parte de la respuesta en formato JSON en la fase 2 Tabla 4. Descripción de datos

importantes

Campo	Descripción
<b>Id</b>	Identificador de la transacción ya sea aprobada o rechazada
<b>resultDetails.Authcode</b>	Campo código de autorización de la transacción aprobada. Ej. 058142.
<b>resultDetails.BatchNo</b>	Número de lote
<b>resultDetails.TotalAmount</b>	Si es una transacción diferida con intereses entonces el valor será el totalizado del monto más el
	valor de los intereses generados por el banco en la transacción, sino el valor será igual al monto normal.
<b>resultDetails.Interest</b>	El valor individual del interés generado por el banco en la transacción
<b>resultDetails.Response</b>	Campo que indica si la transacción fue aprobada o rechazada, esto se detalla en la sección de mensajes.
<b>resultDetails.AcquirerCode</b>	Código asociado a la entidad financiera que aprobó/rechazó la transacción.
<b>resultDetails.ReferenceNo</b>	Número de referencia
<b>resultDetails.CardType</b>	Nomenclatura del Banco dueño de la tarjeta utilizada en la transacción

#### 4 Código de Mensajes.

El listado a continuación muestra los códigos de respuestas emitidos por el switch transaccional. La descripción de los códigos debe ser personalizados para una mejor comprensión por parte del cliente y evitar bloqueos, es decir usted puede cambiar la descripción que le llega como respuesta en el JSON para mostrarlo de mejor manera al usuario. De manera general los códigos que

empiezan por **800** son debido a un rechazo por la entidad bancaria.

## 4.2 Códigos Generales

El botón de pagos maneja dos categorías en su mensajería, una general y otra detallada para de esta manera tener claro el motivo de algún rechazo de la transacción. En la siguiente tabla podrá observar los diferentes tipos de mensajes recibidos en el campo result { code:\_codigo\_,descripción: \_abc\_} del JSON de respuesta, estos les permite tener una idea general del rechazo sin embargo para una mayor certeza revisen el apartado [4.2.2](#)

Tabla 5. Códigos Generales de Mensajes transaccionales en Producción

Código	Descripción Web	Detalle
000.000.000	Transaction succeeded	Approved (Producción). Mensaje para todo proceso exitoso.
000.200.100	successfully created checkout	Creación del id (checkoutId) de manera exitosa.
600.200.201	Channel/Merchant not configured for this payment method	Si este mensaje es emitido para todas las transacciones quiere decir que algún parámetro del MID o TID está mal configurado. Si es para unas cuantas entonces
		puede ser que no tenga códigos asignados de los bancos
800.110.100	Duplicate transaction	Mensaje cuando existe una mala interacción de la interfaz provocando el envío del identificador de la transacción más de una vez. El técnico del comercio debe solucionarlo.
200.100.101	invalid Request Message. No valid XML. XML must be url-encoded! maybe it contains a not encoded ampersand or something similar.	Mensaje generalmente emitido cuando se pone mal el dato de la cédula de identidad, teléfono entre otro. Si todo está bien, entonces se debe revisar el campo <i>Response</i> , si tiene el valor de 05 quiere decir que la respuesta del banco presentó una demora, esto puede causar confusión al cliente debido a que puede haberle llegado un mensaje de consumo sin embargo la respuesta no llegó a Datafast y por ende no le llega al portal web del comercio.

<b>200.100.103</b>	Invalid Request Message. The request contains structural errors	Mensaje invalido. Cuando hay algún dato errado enviado por ejemplo una cédula así 1234567890
<b>800.100.171</b>	Transaction declined (pick up card)	Pick up: Retenga y llame
<b>700.300.700</b>	referenced tx can not be reversed (reversal not possible anymore)	Reverso declinado
<b>800.100.174</b>	transaction declined (invalid amount)	Monto inválido
<b>800.100.151</b>	transaction declined (invalid card)	Tarjeta de crédito inválida
<b>800.100.402</b>	cc/bank account holder not valid	No such issuer. Omisión de nombre de tarjetahabiente.
<b>800.100.190</b>	transaction declined (invalid configuration data)	Transacción declinada por el banco, puede ser por estos motivos dependiendo del campo Response: Enviaron un tipo de crédito que no está autorizado por el banco al comercio. No está autorizado a transaccionar con dicha tarjeta. Transacción rechazada por el banco que no pertenece a nuestra red.
<b>800.100.152</b>	Authorization Error (95) transaction declined by authorization system	La transacción fue rechazada por el banco, se desconoce el motivo. En el caso de tarjetas Diners/Discover/Visa MC del Banco Pichincha se debe revisar el campo ExtendedDescription debido a que puede ser una transacción cancelada por el usuario.
<b>800.100.197</b>	transaction declined (registration cancelled externally)	Customer cancellation
<b>800.100.176</b>	transaction declined (account temporarily not available. Please try again later)	Transacción rechazada por el banco, intentar de nuevo. Esto ocurre generalmente con las tarjetas de débito.
<b>100.400.311</b>	transaction declined (format error)	Format Error
<b>100.100.100</b>	request contains no creditcard, bank account number or bank name	No se envió datos de la tarjeta, generalmente este un problema técnico de la página web del comercio, no del botón.
<b>800.100.165</b>	transaction declined (card lost)	Tarjeta reportada como pérdida
<b>800.100.159</b>	transaction declined (stolen card)	Tarjeta reportada como sustraída



<b>800.100.155</b>	transaction declined (amount exceeds credit)	Fondos insuficientes
<b>100.100.400</b>	request contains no cc/bank account holder	Se envía vacío el cardholder, esto lo controla la página web del comercio.
<b>100.100.303</b>	Card Expired	Tarjeta expirada
<b>800.100.170</b>	transaction declined (transaction not permitted)	Puede darse por el uso de una tarjeta de débito que no puede realizar compras por internet.
<b>100.550.310</b>	amount exceeds limit for the registered account	Monto excede el cupo
<b>800.100.168</b>	transaction declined (restricted card)	Tarjeta restringida, cliente debe llamar al banco.
<b>800.100.179</b>	transaction declined (exceeds withdrawal count limit)	Hay tarjetas que tiene un máximo de transacciones permitidas.
<b>100.100.402</b>	cc/bank account holder not valid	Cardholder inválido, puede ser que tenga caracteres especiales, la validación es del gateway, no del banco.
<b>600.200.100</b>	invalid Payment Method	Modalidad invalida
<b>700.100.200</b>	non matching reference amount	Verifique interés
<b>700.400.200</b>	cannot refund (refund volume exceeded or tx reversed or invalid workflow?)	Se intentó anular una transacción que ya había sido anulada previamente.
<b>800.100.157</b>	transaction declined (wrong expiry date)	Vigencia de la tarjeta errada
<b>800.100.501</b>	Card holder has advised his bank to stop all recurring payments for this merchant	Establecimiento cancelado
<b>800.100.100</b>	transaction declined for unknown reason	Puede ser por dos motivos: La tarjeta tiene problemas por lo que el cliente debe llamar al Banco. Transacción con una tarjeta de débito que no tiene autorizado compras por internet.
<b>800.100.171</b>	transaction declined (pick up card)	Llame al centro de Autorización
<b>800.300.500</b>	transaction temporary blacklisted (too many tries invalid CVV)	La tarjeta se encuentra en lista negra debido a múltiples intentos de transacciones con CVV errado, se libera después de 48 horas.

<b>800.300.501</b>	transaction temporary blacklisted (too many tries invalid expire date)	La tarjeta se encuentra en lista negra debido a múltiples intentos de transacciones con fecha de caducidad errada, se libera después de 48 horas.
<b>900.100.201</b>	error on the external gateway (bank)	Se ha perdido la conexión con el Gateway de Pagos, inténtelo después de unos minutos, si el problema persiste por un lapso estimado entonces debe contactarse con <a href="mailto:servbdpago@datafast.com.ec">servbdpago@datafast.com.ec</a>
<b>900.100.100</b>	unexpected communication error with connector/acquirer	Se ha perdido la conexión con el banco, intentar luego de unos minutos.
<b>900.100.200</b>	error response from connector/acquirer	Ha ocurrido un incidente con respecto al tiempo de respuestas entregadas por el banco
<b>900.100.300</b>	no response from connector/acquirer [uncertain result]	Existió una desconexión en el instante de la transacción, si el problema persiste entonces comunicarse rápidamente con Datafast.
<b>800.100.176</b>	transaction declined (account temporarily not available. Please try again later)	Try Request Again. Transacción rechazada por el banco, intente luego o llame al banco
<b>800.100.199</b>	transaction declined (invalid tax number)	Mensaje emitido cuando se ha calculado mal los impuestos, la base 0% + la base 12% + valor del IVA debe ser igual al monto que se envía.
<b>600.200.201</b>	Channel/Merchant not configured for this payment method	Terminal inválido o diferido no permitido.
<b>600.200.500</b>	Invalid payment data. You are not configured for this currency or sub type (country or brand)	Mensaje emitido cuando no se ha asignado una marca de tarjeta para transaccionar.
<b>800.100.151</b>	transaction declined (invalid card)	Tarjeta inválida
<b>800.100.156</b>	transaction declined (format error)	Depende del valor del código en el campo <i>Response</i> , revisar el apartado 4.3.2
<b>800.100.162</b>	Transaction declined (limit exceeded)	La tarjeta tiene un límite en el monto en compras, aún teniendo cupo.

<b>100.100.101</b>	invalid creditcard, bank account number or bank name /	Tarjeta de crédito inválida
<b>100.700.801</b>	identity contains no or invalid identification value	Se está enviando un campo mal, ya sea en el nombre o en su valor, el técnico debe revisar internamente.
<b>100.400.147</b>	Payment void and transaction denied by ReD Shield	La transacción infringió una regla antifraude. Estas reglas fueron definidas por el comercio al momento de firmar. Para detalles de que regla se activó deben enviar un email servicio al cliente servbdpago@datafast.com.ec
<b>100.400.149</b>	Payment void and data error by ReD Shield	La transacción tiene datos que infringe una regla antifraude. Para detalles de que regla se activó deben enviar un email servicio al cliente servbdpago@datafast.com.ec
<b>100.400.325</b>	External risk system not available	Si este mensaje se presenta en muchas transacciones entonces el comercio debe comunicarse con Datafast de manera inmediata.
<b>SEGUNDO FACTOR DE AUTENTICACIÓN (3DS – OTP)</b>		
<b>100.380.401</b>	User Authentication Failed. (3DSecure Error)	El código ingresado por el tarjetahabiente es incorrecto.
<b>100.380.501</b>	User Authentication Error (risk management transaction timeout)	El tarjetahabiente no digitó el código enviado por el banco dentro del tiempo establecido.
<b>100.396.103</b>	Previously pending transaction timed out (Error in asynchronous workflow)	El tarjetahabiente no digitó el código enviado por el banco dentro del tiempo establecido para tarjetas DINERS/DISCOVER/VISA-MC Banco Pichincha)
<b>100.390.108</b>	Transaction rejected because merchant not participating in 3DSecure program (3DSecure Error)	Error de configuración en el canal. Contactarse inmediatamente con servbdpago@datafast.com.ec
<b>100.390.112</b>	Technical Error in 3D system (3DSecure Error)	Error de configuración en el canal. Contactarse inmediatamente con servbdpago@datafast.com.ec

Tabla 6. Código de Mensajes Generales en Pruebas

Código	Descripción Web	Detalle
<b>000.100.112</b>	Request successfully processed in 'Merchant in Connector Test Mode'	Approved (Pruebas), este mensaje se recibe en la fase 2 cuando se agrega el campo testMode=External
<b>000.100.110</b>	Request successfully processed in 'Merchant in Integrator Test Mode'	Approved (Pruebas), este mensaje se recibe en la fase 1

Tabla 7. Códigos de respuestas en la creación del formulario (1ra. Función).

Código	Descripción Web	Detalle
<b>000.200.100</b>	successfully created checkout	Petición correcta, el checkoutId puede extraerse del JSON de respuesta.
<b>200.300.404</b>	invalid or missing parameter	Campo inválido: puede ser por nombre, longitud o formato.

### 4.3 Códigos de Mensajes Detallados

En cada transacción ya sea aprobada o rechazada se debe tener en cuenta estos dos campos para obtener información más detallada:

1. Response
2. AcquirerCode

#### 4.3.1 Aprobación de la transacción (Banco adquirente)

Para conocer el banco **adquirente** que autorizó la transacción primero se debe verificar que el campo *Response* tenga el valor de **00**, luego se verifica el valor del campo *AcquirerCode* en base a la siguiente tabla. Cuando la transacción es rechazada este campo es igual a null.

Tabla 8. Banco Adquirente

##	Descripción
01	Pacificard
02	Diners
03	Pichincha
04	Banco Guayaquil
06	Solidario
07	Medianet
08	Banco del Austro
09	Coop. 29 de Octubre

Así mismo en este escenario se puede conocer con que tarjeta se realizó la transacción, el valor del campo *CardType* tiene los siguientes valores:

Tabla 9. AZ Tarjetas de crédito.

AZ	Descripción
DC	Diners
PM	Mastercard Pacifico
MI	Mastercard Internacional
PV	Visa Pacífico
VI	Visa Internacional
VP	Visa Pichincha

<b>BG</b>	American Express
<b>VG</b>	Visa Banco Guayaquil
<b>MG</b>	Mastercard Banco Guayaquil
<b>DG</b>	Débito Banco Guayaquil
<b>DP</b>	Débito Pacificard
<b>MP</b>	Mastercard Pichincha
<b>DI</b>	Discover
<b>CO</b>	Coop. 29 de Octubre
<b>VM</b>	Visa Medianet
<b>VA</b>	Visa Banco del Austro
<b>MM</b>	Mastercard Medianet
<b>MA</b>	Mastercard Banco del Austro
<b>CS</b>	Crédito Solidario
<b>DS</b>	Débito Solidario
<b>UP</b>	Unión Pay

Para información adicional, escribir a [servbdpago@datafast.com.ec](mailto:servbdpago@datafast.com.ec)

#### 4.3.2 Rechazo de la Transacción (Respuestas del Banco)

Cuando la transacción es rechazada el valor del campo **Response** tiene alguno de los valores que se detalla en la siguiente tabla.

Tabla 10. Código de respuesta emitida por el banco.

##	Descripción
<b>02</b>	Significa que hay un problema con la tarjeta del cliente, y tiene que llamar al Centro de Autorización del banco
<b>03</b>	Establecimiento inválido. Primero debe verificar si el comercio tiene asignado el tipo de crédito reportado, si todo está bien entonces llamar a DF y consultar.
<b>04</b>	Retenga la tarjeta y llame (puede haber sido reportada como robada o está utilizando una T.C. no válida)
<b>05</b>	Transacción rechazada por el banco, no especifica el motivo, en este caso puede llamar al banco y consultar

<b>07</b>	Retenga la tarjeta y llame
<b>12</b>	Transacción inválida, mensaje general emitido por el banco, puede ser por que la tarjeta no tiene permitido compras por internet, generalmente las de débito.
<b>13</b>	Monto inválido
<b>14</b>	Error en el número de tarjeta
<b>15</b>	Error en el número de tarjeta
<b>17</b>	Socio cancelado
<b>19</b>	Transacción rechazada, favor reintente
<b>41</b>	Tarjeta pérdida. Retenga la tarjeta y llame
<b>43</b>	Tarjeta robada. Retenga la tarjeta y llame
<b>51</b>	Fondos insuficientes
<b>54</b>	Tarjeta expirada
<b>57</b>	Transacción inválida o no permitida (diferidos mal enviado o transacción realizada con una tarjeta de débito con restricciones)
<b>61</b>	Monto excede el crédito disponible
<b>62</b>	Tarjeta restringida
<b>76</b>	Cuenta Inválida. En este caso es por el uso de una tarjeta de crédito no valida (Renovaciones)
<b>77</b>	Modalidad inválida. Cuando no tiene autorizado un tipo de crédito (diferido)
<b>79</b>	Vigencia errada (Fecha de caducidad)
<b>80</b>	Establecimiento cancelado
<b>84</b>	Significa que hay un problema con la tarjeta del cliente, y tiene que llamar al Centro de Autorización del banco
<b>88</b>	Transacción rechazada, favor reintente
<b>89</b>	Terminal inválida, se debe verificar si tiene el tipo de crédito autorizado por el banco caso contrario el comercio debe comunicarse con Datafast para revisión.
<b>91</b>	Entidad fuera de linea (Puede ser que el banco tuvo una intermitencia y justamente llegó la transacción y no la procesó)
<b>FORMAT_ERROR</b>	Este mensaje es debido a la validación de la cédula es decir está sobrepasando los 10 caracteres, generalmente cuando envían el RUC
<b>ER</b>	Puede ser por 3 motivos:

	<ol style="list-style-type: none"> <li>1. Transacción con mal cálculo de impuestos.</li> <li>2. No se está enviando el MID y TID correctos (Error en la programación)</li> <li>3. Transacciones con monto menor a \$1.00</li> </ol>
--	---

## 5 Opciones Avanzadas

Esta sección describirá como utilizar las opciones de tipos de créditos otorgados y autorizados por los bancos. Para esto se debe incluir la librería JQuery, por ejemplo:

```
<script type="text/javascript" src="jquery-3.2.1.js"></script>
```

### 5.1 Diferidos

Para los diferidos o cuotas se debe agregar código javascript en la misma página o sección en donde se invoca el formulario de pago, en el ejemplo a continuación se lo agrega de forma estática, pero se recomienda que esto sea dinámico.

```
<script type="text/javascript">
  var wpwlOptions = {
    onReady: function() {
      var numberOfInstallmentsHtml = '<div class="wpwl-label wpwl-label-custom" style="display:inline-block">diferidos</div>' +
        '<div class="wpwl-wrapper wpwl-wrapper-custom" style="display:inline-block">' +
        '<select name="recurring.numberOfInstallments"><option value="0">0</option><option value="3">3</option>' +
        '</select><option value="6">6</option><option value="9">9</option></select>' +
        '</div>';
      $('form.wpwl-form-card').find('.wpwl-button').before(numberOfInstallmentsHtml);
    },
    style: "card",
    locale: "es",
    labels: {cvv: "CVV", cardholder: "Nombre (Igual que en la tarjeta)"}
  }
</script>
```

Transcripción:

```
<script type="text/javascript">
var wpwlOptions = {
onReady: function() {
var numberOfInstallmentsHtml = '<div class="wpwl-label wpwl-label-custom" style="display:inline-block">Diferidos
'<div class="wpwl-wrapper wpwl-wrapper-custom" style="display:inline-block">' +
'<select name="recurring.numberOfInstallments"><option value="0">0</option><option value="3">3' +
'|</option><option value="6">6</option><option value="9">9</option></select>' +
'</div>';
$('form.wpwl-form-card').find('.wpwl-button').before(numberOfInstallmentsHtml);
},
Style: "card",
locale: "es",
labels: {cvv: "CW", cardHolder: "Nombre (Igual que en la tarjeta)"}
}
</script>
```

Figura 13. Javascript para diferidos (Installments).

Este código se puede generar de forma dinámica si acaso se posee los tipos de diferidos aprobados por el banco emisor en alguna tabla de la base de datos y agregarlos en *campos de texto ocultos*, sino puede dejarlo fijo en el formulario,



igual la validación es que si se selecciona la opción 0 “cero” o simplemente no se envía este campo entonces se lo interpreta como una transacción corriente.

**Código:**

```
<script type="text/javascript">
```

```
    var wpwlOptions = {
```

```
        onReady: function() {
```

```

        var numberOfInstallmentsHtml = '<div class="wpwl-label wpwl-label-custom" style="display:inline-block">Diferidos:</div>' +

        '<div class="wpwl-wrapper wpwl-wrapper-custom" style="display:inline-block">' +

        '<select                name="recurring.numberOfInstallments"><option value="0">0</option><option value="3">3'+

        '</option><option                value="6">6</option><option value="9">9</option></select>' +

        '</div>';

    $('form.wpwl-form-card').find('.wpwl-button').before(numberOfInstallmentsHtml);

} } </script>

```

Muy importante que el nombre (atributo ***name***) del elemento HTML (***select***) sea ***recurring.numberOfInstallments*** debido a que es un campo válido y reconocido por la plataforma de pago, cualquier cambio de este nombre no se garantiza el correcto funcionamiento.

Al agregar este JavaScript no se ve afectada la función request, es un campo que el formulario de pago lo envía de forma automática.

Si se decide en aplicar los diferidos de forma dinámica entonces se sugiere que en un paso anterior a la carga del formulario de pago se le muestre al usuario los tipos de diferidos autorizados y después agregan los campos de manera dinámica a la porción de código Javascript. A continuación, un ejemplo del paso previo a cargar el formulario.

Descripción: Selección de forma de pago, corriente o diferido

Figura 14. Ejemplo de interfaz previa de selección de diferidos

Cuando se agregan las opciones de diferidos, en la respuesta JSON se puede observar un campo adicional en la respuesta de aprobación de una transacción tal como se muestra en la imagen.

```
[recurring] => Array
(
    [numberOfInstallments] => 0
)
```

Transcripción:

[recurring] => Array

(

[numberOfInstallments] => 0

)

Figura 15. Campo de diferidos en respuesta de la transacción

Quedando el formulario con la opción de diferidos de forma estática de la siguiente manera

Descripción: Formulario de tarjeta de crédito visa para poder diferir

Figura 16. Opción de diferidos

## 5.2 Tipo de Crédito

Para utilizar la opción de tipos de crédito se debe conocer con exactitud lo que el banco le haya autorizado, una vez en conocimiento de lo antes expuesto se debe utilizar el campo **customParameters[SHOPPER\_TIPOCREDITO]**. La siguiente tabla muestra los tipos de crédito disponibles con el respectivo valor del parámetro.

Tabla 11. Tipos de crédito

SHOPPER_TIPOCREDITO	Descripción
00	Transacción corriente
01	Diferido corriente
02	Diferido con Interés
03	Diferido sin Interés
07	Diferido con Interés + Meses de Gracia
09	Diferido sin Interés + Meses de Gracia
21	Diferido Plus
22	<i>Duplica tu plazo</i>

La figura siguiente muestra el aspecto de la opción de tipo de crédito utilizando javascript de forma *estática*.



Descripción: ListOption Titulo "Tipo de crédito:" option selected Corriente

Figura 17. Opción Tipo de crédito

El siguiente código de ejemplo se debe agregar en la misma sección en donde se carga el formulario de pago, si lo hacen de forma dinámica entonces en vez del tag select lo reemplazan con un input text de manera oculta pero respetando la propiedad *name* caso contrario el Gateway de pagos no reconocerá el campo.

`var tipocredito =`

```
'<div      class="wpwl-wrapper      wpwl-wrapper-custom"
style="display:inline-block">' +
```



```

value="6">6</option>
<option value="9">9</option></select>' +
'</div>';
$(form.wpwl-form-card").find('.wpwl-button').before(numberOfInstallmentsHtml);
var tipocredito =
<div class="wpwl-wrapper wpwl-wrapper-custom" style="display: inline-block">" +
'Tipo de crédito:<select name="customParameters [SHOPPER_TIPOCREDITO]"><option value="00">Corriente</option>'+
<option value="01">Dif Corriente</option>' +
<option value="02">Dif con int</option>' +
<option value="03">Dif sin int</option>' +
'<option value="07">Dif con int Meses gracia</option>' +
'<option value="09">Dif sin int Meses gracia</option>' +
<option value="21">Dif plus cuotas</option>' +
<option value="22">Dif plus</option>' +
'</div>';
$('form.wpwl-form-card").find('.wpwl-button').before(tipocredito);
$("form.wpwl-form-card").find(".wpwl-button").before(datafast);
},
var datafast= '<br/><br/>'
style: "card",
locale: "es",
labels: (cvv: "CV", cardHolder: "Nombre (Igual que en la tarjeta)", insertCode: "Ingrese el codigo")
</script>

```

Figura 18. Código Javascript de la implementación de Diferidos y Tipos de Crédito

En un navegador web (browser) se mostraría lo siguiente.

Descripción: interfaz datafast tipo visa, formulario con numero de tarjeta, expira, Nombre, cvv diferidos y tipo de crédito

Figura 19. Formulario con las opciones de diferidos y tipos de crédito

Se vuelve a recalcar que todo lo expuesto a nivel de diseño son ejemplos, al final el desarrollador en conjunto con el comercio definirá el flujo y aspecto que más se adapte.

Cuando se ejecuta una transacción con algún tipo de crédito seleccionado entonces en el JSON de respuesta se incluirá este campo.

```
[shipping] => Array
(
    [street1] => Edif. Las Camaras, piso 4
    [country] => EC
)

[customParameters] => Array
(
    [SHOPPER_EndToEndIdentity] => 5856e34692e12a8a342ae7972d18ac
    [SHOPPER TIPOCREDITO] => 00
    [CTPE DESCRIPTOR TEMPLATE] =>
```

Transcripccion:

```
[shipping] => Array
(
    [street1] => Edif. Las Camaras, piso 4
    [country] => EC
)
[customParameters] => Array
(
    [SHOPPER_EndToEndIdentity] => 5856e34692e12a8a342ae7972d18ac
    [SHOPPER TIPOCREDITO] => 00
    [CTPE DESCRIPTOR TEMPLATE] =>
```



Figura 20. Parte de la respuesta de una transacción con tipo de crédito.

### 5.3 Validación campos vacíos

Dentro del formulario del botón de pagos se pueden validar por ejemplo que el campo ***cardholder*** no este vacío para evitar transacciones rechazadas por el banco renombrado el método ***onBeforeSubmitCard***.

```

<script type="text/javascript">
var wpwlOptions = {
  style: "card",
  locale: "es",
  labels: {cvv: "CW", cardholder: "Nombre (Igual que en la tarjeta)"},
  onBeforeSubmitCard: function(){
    if ($("#wpwl-control-cardholder").val() === ""){
      $("#wpwl-control-cardholder").addClass("wpwl-has-error");
      $("#wpwl-control-cardholder").after("<div class='wpwl-hint-cardholder-error'>Campo requerido</div>");
      $("#wpwl-button-pay").addClass("wpwl-button-error").attr("disabled", "disabled");
      return false;
    }else{
      return true;
    }
  }
}
</script>

```

Transcripción:

```

<script type="text/javascript">
var wpwlOptions = {
  style:"card",
  locale:"es",
  labels: {cvv: "CW", cardHolder: "Nombre (Igual que en la tarjeta)"},
  onBeforeSubmit Card: function(){
  if ($("#wpwl-control-cardHolder").val() === ){...
  $("#wpwl-control-cardHolder").addClass("wpwl-has-error");
  $("#wpwl-control-cardHolder").after("<div class='wpwl-hint-cardHolder Error'>Campo requerido</div>");
  $("#wpwl-button-pay").addClass("wpwl-button-error").attr("disabled", "disabled");
  return false;
  }else
  return true;
  }
  }
}
</script>

```

Figura 21. Código para validación del campo cardholder (Nombre del dueño de la tarjeta).

## 5.6 Estilo del formulario

En alguna ocasión se necesita que el formulario se contraste con los colores corporativos, para este caso puede optar con la modalidad “plana”, para esto debe cambiar el parámetro **style** con el valor **plain**. Luego puede aplicar sus propias reglas de estilos.

Tipo de tarjeta	Visa	VISA
Número de la tarjeta	Número de la tarjeta	
Expira	MM / YY	
Nombre del titular de la tarjeta	Nombre del titular de la tarjeta	
Código de control	Código de control	
		Pagar

Formulario de interfaz color blanco

Figura 22. Estilo plano del formulario de pago

## 5.7 Optimización Móvil

Nuestro *Widget* está optimizado para navegadores móviles de forma

predeterminada. Sin embargo, si está creando su propio estilo personalizado para el formulario de pago, aquí hay algunos consejos.

#### 5.7.1 Habilitar el diseño responsivo en el HTML

Agregar el siguiente código entre las etiquetas *head*

```
<meta name = "viewport" content = "ancho = ancho del dispositivo, escala inicial = 1">
```

#### 5.7.2 Mobile First (CSS)

Mejore su formulario con Media Queries, para construir sus estilos móviles para viewports más amplios (como tabletas) y pantallas de alta resolución. Se recomienda que adapte estas consultas multimedia en función del contenido de su sitio en lugar de reflejar las dimensiones fijas de dispositivos específicos.

```
/* Valor por defecto, aplica a todos los dispositivos */
```

```
.button {width: 100%}
```

```
@media (min-width: 480px) {
```

```
    /* esta regla se aplica solo a dispositivos con un ancho de pantalla  
    mínimo de 480px*/
```

```
    .button {
```

```
        width: 50%;
```

```
    }
```

```
}
```

### 5.7.3 Botones y Campos de texto (CSS)

Los campos y botones de entrada más grandes hacen que el uso de su formulario de pago sea mucho más fácil. Apple dice que el toque de dedo promedio es de aproximadamente 44x44 píxeles. Por supuesto, casi todos los campos y botones de entrada son más grandes que 44px. En la mayoría de los casos, solo necesita establecer la altura.

```
.input, .button { height: 44px; width: 100%;}
```

El aspecto sería el que se muestra a continuación, recordar que la adaptación depende de la experiencia de la persona que lo está implementando, todo el código expuesto es referencial.



Figura 23. Formulario adaptado para aplicaciones móviles

## 6 OneclickCheckout (Pago con un click)

Esta guía le permite lograr una aceleración significativa del proceso de compra reutilizando los datos que un cliente ingresó para una transacción.

### 6.2 Preparación del pago (primera compra)

En este punto se repite lo de la [sección 3.2](#), la diferencia radica en agregar código JavaScript como se muestra a continuación.



```
<script type="text/javascript">
  var upwloptions = {
    onReady: function(onReady) {

      var createRegistrationHtml = '<div class="customLabel">Desea guardar de manera segura sus datos?</div>'+
        '<div class="customInput"><input type="checkbox" name="createRegistration" /></div>';
      $('form.wpwl-form-card').find('.wpwl-button').before(createRegistrationHtml);
    },
    style: "card",
    locale: "es",

    labels: {cvv: "Codigo de verificación", cardHolder: "Nombre(Igual que en la tarjeta),"},

    registrations: {
      requireCvv: true,
      hideInitialPaymentForms: true
    }
  }
</script>
```

Transcripción:

```
<script type="text/javascript">
var upwloptions = {
  onReady: function(onReady) {
    var createRegistrationHtml = '<div class="customLabel">Desea guardar de manera segura sus datos?</div>'+
      '<div class="customInput"><input type="checkbox" name="createRegistration" /></div>';
    $('form.wpwl-form-card').find('.wpwl-button').before(createRegistrationHtml);
  },
  style: "card",
  locale: "es",
  labels: {cvv: "Codigo de verificación", cardHolder: "Nombre(Igual que en la tarjeta),"},
  registrations: {
    requireCvv: true,
    hideInitialPaymentForms: true
  }
}
</script>
```

Figura 24. Código para activar la captura (tokenización) de los datos de la tarjeta del cliente.

A continuación, el código.

```
var createRegistrationHtml = '<div class="customLabel">Desea guardar
de manera segura sus datos?</div><div class="customInput"><input
type="checkbox" name="createRegistration" /></div>';
```

```
      $('form.wpwl-form-card').find('.wpwl-
button').before(createRegistrationHtml);
```

Tipo de tarjeta: Visa

Número de la tarjeta:

Expira:

Nombre (igual que en la tarjeta):

Código de verificación:

Nombre (igual que en la tarjeta):

Código de verificación:

¿Desea guardar de manera segura sus datos?

☐

Powered by

 Datafast

    VISA

Figura 25. Formulario incluyendo la opción de captura de datos.

En el Json de respuesta de la transacción podrá observar un nuevo campo llamado **registrationId** el cual contiene el token asociado a los datos de la tarjeta de crédito. Este campo debe ser guardado en una tabla de su base de datos asociada a un id del usuario del sistema, de esta manera usted podrá consultar en la siguiente compra si ese usuario tiene tokens asociados.

En la segunda compra, de existir tokens asociados a ese cliente entonces usted los agregará a la primera función request() usando el parámetro **registrations** de la siguiente manera.

&registrations[0].id= 8ac7a49f6ab04d18016ab2067a2731b2

&registrations[1].id= 8ac7a4a06ab05883016ab2067d3841d0

Quedando el formulario de la siguiente forma.

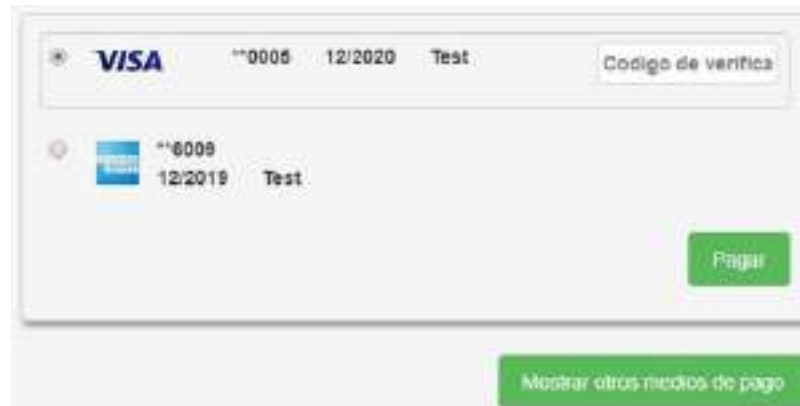


Figura 26. Formulario de pago OneClickCheckout (Pago con un click)

Cabe mencionar que debido a que el formulario cambia su aspecto entonces también cambie sus reglas CSS aplicadas por lo que si desean a este formulario de pago rápido agregar las opciones de diferidos y tipos de crédito entonces el único cambio que deben realizar en la sección de Javascript es cambiar la regla CSS a la cual se hace referencia.

```
var numberOfInstallmentsOcchk = '<div class="wpwl-label wpwl-label-custom" style="display: inline-block">'+  
'<div class="wpwl-wrapper wpwl-wrapper-custom" style="display: inline-block">'+  
'<select name="recurring.numberofInstallments"><option value="8">8</option><option value="6">6</option><option value="9">9</option></select>'+  
'</div>';  
$('form.wpwl-form-registrations').find('.wpwl-button').before(numberOfInstallmentsOcchk);
```

Figura 27. Cambio de regla CSS para los diferidos y tipos de crédito OneClickCheckout

#### Transcripción:

```
var numberOfInstallmentsOcchk = '<div class="wpwl-label wpwl-label-custom" style="display: inline-block">'+  
'<div class="wpwl-wrapper wpwl-wrapper-custom" style="display: inline-block">'+  
'<select name="recurring.numberofInstallments"><option value="8">8</option><option value="6">6</option><option value="9">9</option></select>'+  
'</div>';  
$('form.wpwl-form-registrations').find('.wpwl-button').before (numberOfInstallmentsOcchk);
```

### **6.3 Eliminación del Token**

Una vez almacenado, un token se puede eliminar usando el método DELETE HTTP usando la siguiente función.



```
function request() {
    $url = "https://eu-test.oppwa.com/v1/registrations/8a8294174b7ecb28014b9699220015ca";
    $url .= "?entityId=$entityId";

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Authorization: Bearer OGE4MjkOMTc0YjdIY2lyODAxNGI5Njk5MjIwMDE1Y2N8c3k250pzVDg='));
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'DELETE');
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $responseData = curl_exec($ch);
    if(curl_errno($ch)) {
        return curl_error($ch);
    }
    curl_close($ch);
    return $responseData;
}
```

Transcripción:

```
function request() {
    $url = "https://eu-test.oppwa.com/v1/registrations/8a8294174b7ecb28014b9699220015ca";
    $url. "?entityId=$entityId";
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Authorization: Bearer OGE4MjkOMTc0YjdIY2lyODAxNGI5Njk5MjIwMDE1Y2N8c3k250pzVDg='));
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'DELETE');
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $responseData = curl_exec($ch);
    if (curl_errno($ch)) {
    }
    return curl_error($ch);
    curl_close($ch);
    return $responseData;
}
```

Figura 28. Función de eliminación del token.

La respuesta al ejecutar esta función está en formato JSON en donde podrá obtener el código correspondiente si fue eliminado correctamente.

```
{
  "id": "8ac7a4a26db4c658016db60e28fe3f51", "referencedid": "8ac7a4a16db4c3ba016db609e49733d8", "merchantTransactionid": "transaction 306", "result":
  "code": "000.100.112", "description": "Request successfully processed in
  Merchant in Connector Test Mode", "customer":
  "merchantCustomerId": "1", "build Number": "acd90696fa1135b6bcca17bf1be027856159bce4@2019-
  10-08 14:06:34 +0000", "timestamp": "2019-10-10
  14:24:04-0000", "ndc": "8a8294185a65bf5e015a6c8b89a10d8d_d22020ef08d1490a8d9d3ea80eeb9503"}
}
```

Transcripción:

```
("id": "8ac7a4a26db4c658016db60e28fe3f51", "referencedid": "8ac7a4a16db4c3ba016db609e49733d8", "merchant
Transactionid": "transaction 306", "result":
("code": "000.100.112", "description": "Request successfully processed in
Merchant in Connector Test Mode"], "customer":
merchantCustomerId": "1", "build Number: acd90696fa1135b6bcca17bf1be027856159bce4@2019-
10-08 14:06:34 +0000", "timestamp": "2019-10-10
14:24:04-0000", "ndc": "8a8294185a65bf5e015a6c8b89a10d8d_d22020ef08d1490a8d9d3ea80eeb9503")")
```

Figura 29. JSON de respuesta al ejecutar la función de eliminación.

Si desea comprobar que el token está eliminado puede intentar realizar una transacción con dicho token y obtendrán el siguiente mensaje.

Result Code and Description	Registration Error (64) registration is already deregistered (100.150.202)
-----------------------------	---

Transcripcion:

Result Code and description

Registration Error (64) registration is already deregistered (100.150.202)

Figura 30. Mensaje de error en la transacción ya que no existe el token.

## 7 Anulaciones

En este apartado se indicará el proceso de anulación que debe ser implementado por el comercio de acuerdo con sus metodologías y estándares.

Cada transacción de compra aprobada en su respuesta dentro del formato JSON viene un campo llamado id, tal como se muestra en la siguiente figura.

```

Array
(
    [id] => 8ac7a4a072e61a030172e8629eaf2af9
    [paymentType] => DB
    [paymentBrand] => VISA
    [amount] => 1.12
    [currency] => USD
    [descriptor] => 1459.4271.7847 Low Risk Sector
    [merchantTransactionId] => transaction_644
    [result] => Array
        (
            [code] => 000.100.112
            [description] => Request successfully pro
        )
)

```

Transcripción:

```

Array
(
[id] => 8ac7a4a072e61a030172e8629eaf2af9
[payment Type] => DB
[paymentBrand] => VISA
[amount] => 1.12
[currency] => USD
[descriptor] => 1459.4271.7847 Low Risk Sector
[merchantTransactionId] => transaction_644
[result] => Array
(
[code] => 000.100.112
[description] => Request successfully pro
)

```

Figura 31. Parte del contenido del JSON en una transacción aprobada

En el método mostrado a continuación, lo marcado en color amarillo es el **id** de la transacción aprobada y esta debe concatenarse a la URL que se observa, también el campo **paymentType** debe ser igual a RF y así los otros campos mostrados.

```

function request() {
    $url = "https://eu-test.qppwa.com/v1/payments/{id}";
    $data = "entityId=8a8294174b7ecb28014b9099228015ca" .
        "&amount=10.00" .
        "&currency=USD" .
        "&paymentType=RF" .
        "&testMode=EXTERNAL";

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Authorization:Bearer DGE4Mjk8NTc0Yjd1Y2IyODAxNGI5NjkSMj1uMDE1Y2N8c3k2S0pzVDg='));
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $responseData = curl_exec($ch);
    if(curl_errno($ch)) {
        return curl_error($ch);
    }
    curl_close($ch);
    return $responseData;
}

```

Transcripción:

```
function request() {  
  $url = "https://eu-test.oppwa.com/v1/payments/{id}";  
  $data "entityId=8a8294174b7ecb2801469699220015ca".  
  "&amount-10.00".  
  "&currency-USD".  
  "&paymentType=RF".  
  "&testMode=EXTERNAL";  
  $ch = curl_init();  
  curl_setopt($ch, CURLOPT_URL, $url);  
  curl_setopt($ch, CURLOPT_HTTPSADDER, array(  
    "Authorization: Bearer OGE4Mjk0MTc0YjdIY2IyODAxNGI5Njk5MjIwMDE1Y2N8c3k25@pzVDg="));  
  curl_setopt($ch, CURLOPT_POST, 1);  
  curl_setopt($ch, CURLOPT_POSTFIELDS, $data);  
  curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in production  
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);  
  $responseData = curl_exec($ch);  
  if(curl_errno($ch)) {  
  }  
  return curl_error($ch);  
  curl_close($ch);  
  return $responseData;  
}
```

Figura 32. Método de anulación

Una vez ejecutado el método se obtendrá la respuesta en formato JSON de la siguiente manera.

```

Array
{
  [id] => 8ac7a4a072e61a030172e801495e51e0
  [referencedId] => 8ac7a49f72e6114e0172e7b98da03751
  [paymentType] => RF
  [amount] => 1.12
  [currency] => USD
  [descriptor] => 5636-2635.2023 Low Risk Sector
  [merchantTransactionId] => transaction_636
  [result] => Array
  (
    [code] => 000.100.112
    [description] => Request successfully processed in 'Merchant in Connector Test Mode'
  )
  [resultDetails] => Array
  (
    [ExtendedDescription] => Transaction succeeded (Transacción Aprobada)
    [clearingInstituteName] => Datafast
    [AuthCode] => 058682
    [ConnectorTxID1] => 8ac7a4a072e61a030172e801495e51e0
    [ReferenceNbr] => 200624_000045
    [AcquirerResponse] => 00_04VG
  )
  [customer] => Array
  (
    [merchantCustomerId] => 880000000001
  )
  [buildNumber] => alba3f77948f3988ab60064f38489ae29b2a63da@2020-06-24 16:41:11 +0000
  [timestamp] => 2020-06-24 20:24:59+0000
  [ndc] => 8a8294185a65bf5e815a6c8b89a10d8d_c91b3c90bc854145a00611e3843948bd
}

```

Transcripción:

Array

```

(
  [id] => 8ac7a4a072e61a030172e801495e51e0
  [referencedId] => 8ac7a49f72e6114e0172e7b98da03751
  [paymentType] => RF
  [amount] => 1.12
  [currency] => USD
  [descriptor] => 5636-2635.2023 Low Risk Sector
  [merchantTransactionId] => transaction_636
  [result] => Array
  (
    [code] => 000.100.112
    [description] => Request successfully processed in "Merchant in Connector Test Mode"
  )
  [resultDetails] => Array
  (
    [ExtendedDescription] => Transaction succeeded (Transacción Aprobada)
    [clearingInstituteName] => Datafast
    [AuthCode] => 058682
    [ConnectorTxID1] => 8ac7a4a072e61a030172e801495e51e0
    [ReferenceNbr] => 200624_000045
    [AcquirerResponse] => 00_04VG
  )
  [customer] => Array
  (
    [merchant CustomerId] => 000000000001
  )
  [buildNumber] => alba3f77948f3988ab60064f38489ae29b2a63da@2020-06-24 16:41:11 +0000
  [timestamp] => 2020-06-24 20:24:59+0000
  [ndc] => 888294185a65bf5e815a6c8b89a10d8d_c91b3c90bc854145a00611e3843948bd
)

```

Figura 33. Respuesta de una anulación en formato JSON

## 8 Otros métodos de integración

Una manera alterna para la integración de nuestro botón de pagos es haciéndolo con la modalidad **server-to-server**, pero esta opción no es la ofrecida por defecto a los comercios debido a que esto implica la validación de algunos factores de riesgos y certificados.

Esta modalidad debe ser validada por Datafast y las entidades financieras involucradas.

## 9 Personalización

El botón de pagos puede ser personalizado en su apariencia (CSS), solo basta conocer el nombre de las clases asociadas actualmente a los elementos. Para adicional información se debe solicitar a Datafast.

```

<script>
var wpwlOptions = {
  iframeStyles: {
    'card-number-placeholder': {
      'color': '#ff0000',
      'font-size': '16px',
      'font-family': 'monospace'
    },
    'cvv-placeholder': {
      'color': '#0000ff',
      'font-size': '16px',
      'font-family': 'Arial'
    }
  }
}
</script>

```

Transcripción:

```

<script>
var wpwlOptions = {
  iframeStyles: {
    'card-number-placeholder': {
      'color':
      '#ff0000',
      'font-size':
      '16px',
      'font-family':
      'monospace'
    },
    'cvv-placeholder': {
      'color': '#0000ff',
      'font-size': '16px',
      'font-family': 'Arial'
    }
  }
}
</script>

```

Figura 34. Imagen de referencia en la personalización de reglas de estilos.

Para validaciones que involucren JavaScript seguir la referencia del apartado [5.5](#)

## 10 Verificador de transacciones

En el caso de que el portal no reciba el mensaje de aprobación o rechazo de la transacción se ha implementado la funcionalidad de verificar si la misma consta en nuestra base, de ser así usted podrá realizar una consulta y obtener los datos de esta e ingresarla a su sistema. Existen dos maneras de realizar las verificaciones.

### 9.1 Verificar por identificador de la transacción

En esta modalidad se podrá consultar mediante el identificador de la transacción es decir por el **paymentId**

```

function request() {
    $url = "https://eu-test.oppwa.com/v1/query/$identificador";
    $url .= "?entityId=8a8294174b7ecb28014b9699220015ca";

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Authorization: Bearer OGE4Mjk0MTc0Yjd1Y2lyODAxNGI5Njk5MjIwMDE1Y2N8c3k25@pzVDg=*');
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $responseData = curl_exec($ch);
    if(curl_errno($ch)) {
        return curl_error($ch);
    }
    curl_close($ch);
    return $responseData;
}

```

Transcripción:

```

function request() {
    $url = "https://eu-test.oppwa.com/v1/query/$identificador";
    $url .= "?entityId=8a8294174b7ecb28014b9699220015ca";
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        "Authorization: Bearer OGE4Mjk0MTc0Yjd1Y2lyODAxNGI5Njk5MjIwMDE1Y2N8c3k25@pzVDg=*"
    ));
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $responseData = curl_exec($ch);
    if(curl_errno($ch)) {
    }
    return curl_error($ch);
    curl_close($ch);
    return $responseData;
}

```

Figura 35. Función para verificar y obtener si una transacción fue procesada (aceptada/rechazada)



Cuando se ejecuta la función se obtendrá una respuesta en formato JSON en donde encontrará los datos de la transacción la cual puede ser procesada por la aplicación del comercio.

```
[resultDetails] => Array
(
    [RiskFraudStatusCode] => ACCEPT
    [RequestId] => 350545885144
    [ConnectorTxID1] => 8ac7a4a26ca94245016cab600999378d
    [AuthCode] => 123134
    [ReferenceNbr] => 190819_000014
    [EXTERNAL_SYSTEM_LINK] => https://csi-stage.redworldwide.c
    [OrderId] => 447145618125
    [ExtendedDescription] => Transaction succeeded
    [RiskStatusCode] => PENDING
    [clearingInstituteName] => Datafast
    [RiskResponseCode] => 0100
    [AcquirerResponse] => 00_04VG
    [RiskOrderId] => 000537010000XC120190819153454054
)
```

Transcripción:

```
[resultDetails] => Array
(
    [RiskFraudStatusCode] => ACCEPT
    [RequestId] => 350545885144
    [ConnectorTxID1] => 8ac7a4a26ca94245016cab600999378d
    [AuthCode] => 123134
    [ReferenceNbr] => 190819_000014
    [EXTERNAL_SYSTEM_LINK] => https://csi-stage.redworldwide.c
    [OrderId] => 447145618125
    [ExtendedDescription] => Transaction succeeded
    [RiskStatusCode] => PENDING
    [clearingInstituteName] => Datafast
    [RiskResponseCode] => 0100
    [AcquirerResponse] => 00_04VG
    [RiskOrderId] => 000537010000XC120190819153454054
)
```

Figura 36. Muestra parcial del JSON de respuesta donde encontrarán datos importantes

Identificador de la transacción:				
8ac9a4a16c8b54c6016caa0e66e55a4a				
<div> <div>Buscar</div> <div>Home</div> </div>				
Transacciones encontradas:				
Id Transaccion	Payment Brand	Amount	Paymet Type	Transaction id
8ac7a4a26ca94245016cab600999378d	VISA	11.20	DB	transaction_463

Transcripcion:

Identificador de la transacción:

8ac9a4a16c8b54c6016caa0e66e55a4a

Buscar

Borrar

Transacciones encontradas

Id Transaccion

Payment Brand

Amount

Paymet Type

Transaction id

8ac7a4a26ca94245016cab600999378d

VISA

11.20

DB

transaction\_463

Descripcion tabla de ejmplo con datos

Figura 37. Ejemplo de funcionamiento en un portal web

## 9.2 Verificar por el número de transacción del comercio

En esta modalidad se puede consultar por el identificador único de la transacción del comercio (**merchantTransactionId**).

```
function request() {
    $url = "https://eu-test.oppwa.com/v1/query";
    $url .= "?entityId=8a8294174b7ecb28014b9699220015ca";
    $url .= "&merchantTransactionId=test123";

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Authorization: Bearer OGE4Mjk0MTc0YjdY2lyODAxNGI5Njk5MjIwMDE1Y2N8c3k2S@pzVDg='));
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $responseData = curl_exec($ch);
    if(curl_errno($ch)) {
        return curl_error($ch);
    }
    curl_close($ch);
    return $responseData;
}
```

Transcripción:

```
function request() {
    $url = "https://eu-test.oppwa.com/v1/query";
    $url .= "?entityId=8a8294174b7ecb28014b9699220015ca";
    $url .= "&merchant TransactionId=test123";
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Authorization: Bearer OGE4Mjk0MTc0YjdY2lyODAxNGI5Njk5MjIwMDE1Y2N8c3k2S@pzVDg='));
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true in production
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    ++ $responseData = curl_exec($ch);
    if(curl_errno($ch)) {
```

```
}  
return curl_error($ch);  
curl_close($ch);  
return $responseData;  
}
```

Figura 38. Función para verificar y obtener si una transacción fue procesada (aceptada/rechazada) mediante el **merchantTransactionId**

Al igual que la primera modalidad esta función retorna como respuesta un JSON en el cual podrán obtener los datos de la transacción. Deben tomar en cuenta que cambian ciertas claves (**keys**) dentro del array y que puede retornar más de un dato en el caso de que se desea consultar una transacción y esta fue anulada, por lo que en la consulta se obtendrán dos registros, el DB y el RF

## 11 Paso a Producción (Cronograma *estimado*)

Para salir a producción se debe tener presente el siguiente procedimiento de certificación.

Una vez que se ha desarrollado el botón en ambiente de pruebas se realiza lo siguiente:

Tabla 12. Pasos para la salida a producción

	Descripción	Tiempo Estimado	Responsable
<b>Script de Pruebas</b>	Datafast enviará un archivo con las instrucciones necesarias para llenarlo con datos de las transacciones de pruebas	1 día	Comercio
<b>Validación de Script</b>	Una vez que el comercio ha entregado vía correo electrónico el archivo de las transacciones entonces se procede a validar que cumpla con los requisitos establecidos en este documento. Si existe alguna inconsistencia en los datos se le hará conocer para su corrección por parte del Comercio	1 a 2 días	Datafast / Comercio
<b>Escaneo de Vulnerabilidades</b>	Datafast con el fin de garantizar que las transacciones se realicen en un ambiente seguro procede a escanear el sitio web del comercio en busca de vulnerabilidades. Se entrega un informe al comercio, si hay remediaciones por hacer entonces será el comercio quien estime sus tiempos de correcciones, una vez entregada las modificaciones se reinicia el proceso de escaneo	2 días laborables (Escaneo e informe)	Datafast / Comercio

		lo que involucra otra vez de 24 a 48 horas laborables para entregar el informe.		
<b>Gestión de códigos bancarios</b>	<b>de</b>	Datafast se encargará de gestionar la creación de códigos bancarios que autoricen al comercio poder transacciones con el botón de pagos. Es importante tener claro los tipos de créditos habilitados por la entidad financiera	4 días laborables	Datafast
<b>Gestión de códigos transaccionales</b>	<b>de</b>	Datafast se encarga de la creación de MID y TID una vez que el banco haya proporcionado los códigos como se menciona en el ítem anterior.	1 día	Datafast
<b>Ajuste de códigos</b>	<b>de</b>	El comercio se encargará de realizar los ajustes al código para salir a producción ( <a href="#">ANEXO I</a> ), notificará a Datafast cuando haya terminado para la revisión respectiva	1 día	Comercio
<b>Salida Producción</b>	<b>a</b>	Se coordina fecha y hora para realizar una transacción en producción para esto el comercio deberá tener una tarjeta de crédito real, así como un ítem de prueba en su sistema con un valor no mayor a \$1.00	1 día	Datafast/Comercio

## ANEXOS

### Anexo A.1 Php

Paso 1. Código básico en lenguaje PHP, función para obtener el CheckoutId.

```
function request() {  
  
    $url = "https://eu-test.oppwa.com/v1/checkouts";  
  
    $data = "entityId=8a829418533cf31d01533d06f2ee06fa" .  
  
        "&amount=92.00" .  
  
        "&currency=USD" .  
  
        "&paymentType=DB";  
  
    $ch = curl_init();  
  
    curl_setopt($ch, CURLOPT_URL, $url);  
  
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(  
  
        'Authorization:Bearer  
OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUVOWA==');  
  
    curl_setopt($ch, CURLOPT_POST, 1);  
  
    curl_setopt($ch, CURLOPT_POSTFIELDS, $data);  
  
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true  
in production  
  
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);  
  
    $responseData = curl_exec($ch);  
  
    if(curl_errno($ch)) {  
  
        return curl_error($ch);  
  
    }  
  
    curl_close($ch);  
  
    return $responseData;  
  
}
```

## Anexo A.2

PASO 3. Código básico en PHP, función de comprobación del estado de la transacción. Previamente ya se ha obtenido el parámetro \$resourcePath basado en la URL en el atributo action del tag FORM

```
function request() {  
  
    $url = "https:// eu-test.oppwa.com".$resourcePath;  
  
    $url .= "?entityId=".$entityId;  
  
  
    $ch = curl_init();  
  
    curl_setopt($ch, CURLOPT_URL, $url);  
  
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(  
        'Authorization:Bearer  
OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUVOWA==');  
  
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');  
  
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // this should be set to true  
in production  
  
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);  
  
    $responseData = curl_exec($ch);  
  
    if(curl_errno($ch)) {  
        return curl_error($ch);  
    }  
  
    curl_close($ch);  
  
    return $responseData;  
}
```

## Anexo B.1 Java

Paso 1. Código básico en lenguaje JAVA, función para obtener el checkoutId

```
private String request() throws IOException {

    URL url = new URL("https://eu-test.oppwa.com/v1/checkouts");

    HttpURLConnection conn = (HttpURLConnection) url.openConnection();

    conn.setRequestMethod("POST");

    conn.setRequestProperty("Authorization", "Bearer
OGE4Mjk0MTc0YjdIyODAxNGI5Njk5MjIwMDE1Y2N8c3k2S0pzVDg=");

    conn.setDoInput(true);

    conn.setDoOutput(true);

    String data = ""

        + "entityId=8a8294174b7ecb28014b9699220015ca"

        + "&amount=92.00"

        + "&currency=USD"

        + "&paymentType=DB";

    DataOutputStream wr = new DataOutputStream(conn.getOutputStream());

    wr.writeBytes(data);

    wr.flush();

    wr.close();

    int responseCode = conn.getResponseCode();

    InputStream is;

    if (responseCode >= 400) is = conn.getErrorStream();

    else is = conn.getInputStream();

    return IOUtils.toString(is);

}
```



## Anexo B.2

PASO 3. Código básico en JAVA, función de comprobación del estado de la transacción. Previamente ya se ha obtenido el parámetro String resourcePath basado en la URL en el atributo action del tag FORM

```
private String request() throws IOException {

    URL url = new URL("https://eu-test.oppwa.com/v1/checkouts/{id}/payment
?entityId 8a829418533cf31d01533d06f2ee06fa");

    HttpURLConnection conn = (HttpURLConnection) url.openConnection();

    conn.setRequestMethod("GET");

    conn.setRequestProperty("Authorization", "Bearer
OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUVOWA==");

    int responseCode = conn.getResponseCode();

    InputStream is;

    if (responseCode >= 400) is = conn.getErrorStream();

    else is = conn.getInputStream();

    return IOUtils.toString(is);

}
```

## Anexo C.1 C#

Paso 1. Código básico en lenguaje C#, función para obtener el checkoutId

```
public Dictionary<string, dynamic> Request() {  
    Dictionary<string, dynamic> responseData;  
    string data=" entityId=8a829418533cf31d01533d06f2ee06fa" +  
        "&amount=92.00" +  
        "&currency=USD" +  
        "&paymentType=DB";  
    string url = "https://eu-test.oppwa.com/v1/checkouts";  
    byte[] buffer = Encoding.ASCII.GetBytes(data);  
    HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);  
    request.Method = "POST";  
    request.Headers["Authorization"] = "Bearer  
OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUVOWA==";  
    request.ContentType = "application/x-www-form-urlencoded";  
    Stream postData = request.GetRequestStream();  
    postData.Write(buffer, 0, buffer.Length);  
    postData.Close();  
    using (HttpWebResponse response =  
(HttpWebResponse)request.GetResponse())  
    {  
        Stream dataStream = response.GetResponseStream();  
        StreamReader reader = new StreamReader(dataStream);  
        var s = new JavaScriptSerializer();  
        responseData = s.Deserialize<Dictionary<string,  
dynamic>>(reader.ReadToEnd());  
        reader.Close();  
        dataStream.Close();  
    }    return responseData;  
}
```

## Anexo C.2

PASO 3. Código básico en C#, función de comprobación del estado de la transacción. Previamente ya se ha obtenido el parámetro String resourcePath basado en la URL en el atributo action del tag FORM

```
public Dictionary<string, dynamic> Request() {  
    Dictionary<string, dynamic> responseData;  
  
    string data="entityId=8a829418533cf31d01533d06f2ee06fa";  
  
    string url = "https://eu-test.oppwa.com/v1/checkouts/{id}/payment?" + data;  
  
    HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);  
  
    request.Method = "GET";  
  
    request.Headers["Authorization"] = "Bearer  
OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUVOWA==";  
  
    using (HttpWebResponse response =  
(HttpWebResponse)request.GetResponse())  
    {  
        Stream dataStream = response.GetResponseStream();  
  
        StreamReader reader = new StreamReader(dataStream);  
  
        var s = new JavaScriptSerializer();  
  
        responseData = s.Deserialize<Dictionary<string,  
dynamic>>(reader.ReadToEnd());  
  
        reader.Close();  
  
        dataStream.Close();  
    }  
  
    return responseData;  
}
```

## Anexo D.1 Python

Paso 1. Código básico en lenguaje Python, función para obtener el checkoutId

```
import urllib, urllib2, json
```

```
def request():
```

```
    url = "https://eu-test.oppwa.com/v1/checkouts"
```

```
    data = {
```

```
        'entityId' : '8a829418533cf31d01533d06f2ee06fa',
```

```
        'amount' : '92.00',
```

```
        'currency' : 'USD',
```

```
        'paymentType' : 'DB'
```

```
    }
```

```
    try:
```

```
        opener = urllib2.build_opener(urllib2.HTTPHandler)
```

```
        request = urllib2.Request(url, data=urllib.urlencode(data))
```

```
        request.add_header('Authorization', 'Bearer  
OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUVOWA==')
```

```
        request.get_method = lambda: 'POST'
```

```
        response = opener.open(request)
```

```
        return json.loads(response.read());
```

```
    except urllib2.HTTPError, e:
```

```
        return e.code;
```

## Anexo D.2

PASO 3. Código básico en Python, función de comprobación del estado de la transacción. Previamente ya se ha obtenido el parámetro String resourcePath basado en la URL en el atributo action del tag FORM

```
import urllib, urllib2, json
```

```
def request():
```

```
    url = "https://eu-test.oppwa.com/v1/checkouts/{id}/payment"
```

```
    url += '?entityId=8a829418533cf31d01533d06f2ee06fa'
```

```
    try:
```

```
        opener = urllib2.build_opener(urllib2.HTTPHandler)
```

```
        request = urllib2.Request(url, data="")
```

```
        request.add_header('Authorization', 'Bearer  
OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUVOWA==')
```

```
        request.get_method = lambda: 'GET'
```

```
        response = opener.open(request)
```

```
        return json.loads(response.read());
```

```
    except urllib2.HTTPError, e:
```

```
        return e.code;
```

## Anexo E.1 Node.js

Paso 1. Código básico en Node.js, función para obtener el checkoutId

```
const https = require('https');

const querystring = require('querystring');

const request = async () => {

    const path='/v1/checkouts';

    const data = querystring.stringify({

        'entityId':'8a8294174b7ecb28014b9699220015ca',

        'amount':'92.00',

        'currency':'EUR',

        'paymentType':'DB'

    });

    const options = {

        port: 443,

        host: 'eu-test.oppwa.com',

        path: path,

        method: 'POST',

        headers: {

            'Content-Type': 'application/x-www-form-urlencoded',

            'Content-Length': data.length,

            'Authorization':'Bearer

OGE4Mjk0MTc0YjdIY2IyODAxNGI5Njk5MjIwMDE1Y2N8c3k2S0pzVDg='

        }

    };

    return new Promise((resolve, reject) => {

        const postRequest = https.request(options, function(res) {

            const buf = [];

            res.on('data', chunk => {

                buf.push(Buffer.from(chunk));

            });

        });

    });

}
```

```

    });
    res.on('end', () => {
        const jsonString = Buffer.concat(buf).toString('utf8');
        try {
            resolve(JSON.parse(jsonString));
        } catch (error) {
            reject(error);
        }
    });
});
postRequest.on('error', reject);
postRequest.write(data);
postRequest.end();
});
};

```

## Anexo E.2

PASO 3. Código básico en Node.js, función de comprobación del estado de la transacción. Previamente ya se ha obtenido el parámetro String resourcePath basado en la URL en el atributo action del tag FORM

```
const https = require('https');
```

```
const querystring = require('querystring');
```

```
const request = async () => {
```

```
    var path='/v1/checkouts/{id}/payment';
```

```
    path += '?entityId=8a8294174b7ecb28014b9699220015ca';
```

```
    const options = {
```

```
        port: 443,
```

```

        host: 'eu-test.oppwa.com',
        path: path,
        method: 'GET',
        headers: {
            'Authorization': 'Bearer
OGE4Mjk0MTc0YjdIY2IyODAxNGI5Njk5MjIwMDE1Y2N8c3k2S0pzVDg='
        }
    };

    return new Promise((resolve, reject) => {
        const postRequest = https.request(options, function(res) {
            const buf = [];
            res.on('data', chunk => {
                buf.push(Buffer.from(chunk));
            });
            res.on('end', () => {
                const jsonString = Buffer.concat(buf).toString('utf8');
                try {
                    resolve(JSON.parse(jsonString));
                } catch (error) {
                    reject(error);
                }
            });
        });
        postRequest.on('error', reject);
        postRequest.end();
    });
};

```



## Anexo F.1 VB.net

Paso 1. Código básico en VB.net, función para obtener el checkoutId

```
Public Function Request() As Dictionary(Of String, Object)

    Dim url As String = "https://eu-test.oppwa.com/v1/checkouts"

    Dim data As String = "" +
        "entityId=8a829418533cf31d01533d06f2ee06fa" +
        "&amount=92.00" + "&currency=USD" + "&paymentType=DB"

    Dim req As WebRequest = WebRequest.Create(url)

    req.Method = "POST"

    req.Headers.Add("Authorization", "Bearer
OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUVOWA==")

    req.ContentType = "application/x-www-form-urlencoded"

    Dim byteArray As Byte() = Encoding.UTF8.GetBytes(data)

    req.ContentLength = byteArray.Length

    Dim dataStream As Stream = req.GetRequestStream()

    dataStream.Write(byteArray, 0, byteArray.Length)

    dataStream.Close()

    Dim res As WebResponse = req.GetResponse()

    Dim resStream = res.GetResponseStream()

    Dim reader As New StreamReader(resStream)

    Dim response As String = reader.ReadToEnd()

    reader.Close()

    resStream.Close()

    res.Close()

    Dim jss As New System.Web.Script.Serialization.JavaScriptSerializer()

    Dim dict As Dictionary(Of String, Object) = jss.Deserialize(Of Dictionary(Of String,
Object))(response)

    Return dict

End Function
```

## Anexo F.2

PASO 3. Código básico en VB.net, función de comprobación del estado de la transacción. Previamente ya se ha obtenido el parámetro String resourcePath basado en la URL en el atributo action del tag FORM

```
Public Function Request() As Dictionary(Of String, Object)
```

```
    Dim url As String = "https://eu-test.oppwa.com/v1/checkouts/{id}/payment" +  
        "?entityId=8a829418533cf31d01533d06f2ee06fa"
```

```
    Dim req As WebRequest = WebRequest.Create(url)
```

```
    req.Method = "GET"
```

```
    req.Headers.Add("Authorization", "Bearer  
OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUVOWA==")
```

```
    req.ContentType = "application/x-www-form-urlencoded"
```

```
    Dim res As WebResponse = req.GetResponse()
```

```
    Dim resStream = res.GetResponseStream()
```

```
    Dim reader As New StreamReader(resStream)
```

```
    Dim response As String = reader.ReadToEnd()
```

```
    reader.Close()
```

```
    resStream.Close()
```

```
    res.Close()
```

```
    Dim jss As New System.Web.Script.Serialization.JavaScriptSerializer()
```

```
    Dim dict As Dictionary(Of String, Object) = jss.Deserialize(Of Dictionary(Of String,  
Object))(response)
```

```
    Return dict
```

```
End Function
```

```
responseData = Request()["result"]["description"]
```

## Anexo G

Comprobación de conectividad con el gateway de pagos

```
curl https://eu-test.oppwa.com/v1/checkouts -d  
"entityId=8a829418533cf31d01533d06f2ee06fa" -d "amount=92.00" -d  
"currency=USD" -d "paymentType=DB" -H "Authorization: Bearer  
OGE4Mjk0MTg1MzNjZjMxZDAxNTMzZDA2ZmQwNDA3NDh8WHQ3RjlyUUVOWA=="
```

## Anexo H (Requerimientos Técnicos)

Los requerimientos técnicos se mencionan a continuación:

1. Sitio Web en producción con solución de carrito de compras que contenga el stock de todos los productos e información del Dominio (Esta deberá encontrarse en la página web c/u en una sección independiente):
  - a. Política de Privacidad,
  - b. Contacto (correo y # de Teléfono),
  - c. Políticas de Envío y Entrega (no aplica en comercios cuyos giros son servicios),
  - d. Términos y condiciones del comercio hacia el cliente.
2. Análisis de contenido Web y términos de Servicio establecidos por las Marcas de Tarjeta Mastercard (BRAM) y Visa (GBPP): todos los productos que se comercializan deben ser los permitidos por las marcas.

Su organización deberá verificar que sus sistemas admitan uno de los cifrados disponibles de esta lista para continuar conectándose a nuestro API de la plataforma de pagos.

TLS v1.3 (suites en orden de preferencia del servidor)

- TLS\_AES\_256\_GCM\_SHA384
- TLS\_CHACHA20\_POLY1305\_SHA256
- TLS\_AES\_128\_GCM\_SHA256

TLS v1.2 (suites en orden de preferencia del servidor)

- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_CHACHA20\_POLY1305\_SHA256

3. Tener dos ambientes de integración (Desarrollo y producción)
4. El proveedor de Web Host del sitio debe ser PCI DSS.
5. El sitio no debe tener Malware o HTML Spam.
6. La IP debe ser única y no debe alojar otros dominios.
7. El sitio web no debe alojar vínculos hacia otros sitios web considerados como dañinos.

Posterior al escaneo inicial, se efectuarán Escaneos mensuales donde se evaluarán los puntos de escaneo por primera vez, más cambios significativos realizados.

## Anexo I

### Actualización de código para el paso a producción:

1.- Cambiar las URL de test (<https://eu-test.oppwa.com>) a producción (<https://eu-prod.oppwa.com>) en todos los métodos del API implementado en su portal web, también en el front-end (Javascript).

2.- Eliminar el parámetro `testMode=EXTERNAL` de todos los métodos (transacciones y anulaciones, tokenización, etc.).

3.- Si su código es PHP modificar el parámetro de los métodos a *true*.

```
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
```

4.- Cambiar las credenciales de test a producción (Estas deben ser proporcionadas por DF).

- a) ID de entidad
- b) ACCESS TOKEN

5.- Cambiar el MID y TID (Estas deben ser proporcionadas por DF).

- c) MID
- d) TID

6.- Agregar imagen de certificación. Si no tiene JS entonces agregue esto debajo del formulario de pago(<form></form>)

```
<script type="text/javascript">
```

```
var wpwlOptions = {
```

```
    onReady: function() {
```

```
        var datafast= '<br/><br/><img
```

```
src='+""https://www.datafast.com.ec/images/verified.png"
```

```
style='+""display:block;margin:0 auto; width:100%;">';
```

```
$('#form.wpwl-form-card').find('.wpwl-button'). before(datafast);
```

```
    },style: "card",locale: "es", labels: {cvv: "CVV", cardHolder: "Nombre(Igual que en la tarjeta)"}  
}
```

```
}
```

```
</script>
```

Una vez realizado esto, por favor enviar una captura de pantalla del botón ya que con este cambio, debe mostrarse el **Powered by Datafast** con la imagen de las marcas de tarjetas. Para ALIA, Datafast debe proporcionarle la instrucción para incluir el logo.



No realizar transacciones de prueba en producción sin coordinación de Datafast, una vez que se confirme la prueba con fecha y hora, se procederá a revisar y si todo está correcto se dará por terminado el proceso por medio de un correo electrónico, es importante nos comunique antes de realizar la primera y única transacción en producción sin importar la respuesta obtenida debido a que en este ambiente las reglas de prevención de fraude están activas y pueden bloquear el exceso de pruebas locales, adicional se recomienda un monto mínimo de \$1.00 para dicha prueba.

Finalmente, una vez en producción cualquier consulta o requerimiento deberá ser canalizada a través de nuestra cuenta de correo [servbdpago@datafast.com.ec](mailto:servbdpago@datafast.com.ec)