

SDKInApp

Manual de integración iOS

Versión: 2.4.0
Fecha: 30/9/24
Referencia: XXXXX/XX.xx
USO RESTRINGIDO



Redsys, Servicios de Procesamiento, S.L. - c/ Francisco Sancha, 12 - 28034 Madrid (España)

www.redsys.es

Autorizaciones y control de versión

AUTOR: Autor del documento	VALIDADO POR: Área propietaria	APROBADO POR:
Empresa: Redsys	Empresa: Redsys	Empresa: Redsys
Firma:	Firma:	Firma:
Fecha: 15/05/2018	Fecha: 15/05/2018	Fecha: 15/05/2018
<p>Comentarios: La gestión de la documentación impresa es responsabilidad de la persona que la imprime.</p> <p>Las versiones impresas de las normas de seguridad no garantizan ser la última versión aprobada. Para consultar la última versión acceder a la base de datos de Alejandría.</p> <p>Para el tratamiento de la información contenida en este documento se deberá seguir las pautas establecidas en la Normativa Redsys, y en particular en la norma RS.RI.SEG.NOR.0003 NORMA DE CLASIFICACIÓN Y TRATAMIENTO DE LA INFORMACIÓN</p>		

Versión	Fecha	Afecta	Breve descripción del cambio
1.0	15/06/2018	Todo	Versión inicial
2.0	04/09/2018	Subida al App Store	Hemos añadido un script para poder realizar la subida al App Store
2.0.1	05/09/2018	Mensaje de error	Añadimos mensaje de error cuando el comercio configura una URL_OK o URL_KO no valida.
2.0.2	14/09/2018	Compatibilidad	Versión con mejora para la compatibilidad con Objective-C.
2.0.3	05/10/2018	Actualización	Actualización con nueva versión de Xcode y Swift 4.2.
2.0.5	31/10/2018	Todo	Nueva versión de la librería
2.0.8	02/04/2019	Todo	Actualización con nueva versión de XCode 10.2
2.0.9	19/07/2019	Todo	Nueva versión de la librería. Actualización a Swift 5.0
2.0.10	25/09/2019	Todo	Nueva versión de la librería
2.1	03/10/2019	Todo	Actualización con nueva versión de Xcode 11.0.

			Cambio a dos librerías. Eliminado script.
2.1.1	18/10/2019	Integración de la librería Subida al App Store	Actualización para Xcode 11. Distinguir las dos librerías.
2.1.2	04/12/2019	Publicar en el App Store	Distinguir las dos librerías.
2.1.3	14/01/2020	Pago directo. Pago webview.	Nuevo tipo de operación: Autenticación.
2.1.4	30/03/2020	Todo	Nueva versión de la librería.
2.1.5	06/04/2020	Pago webview	Nueva Versión. Nuevo componente WKWebView.
2.1.6	14/07/2020	Pago directo. Pago webview.	Nueva Versión. Nuevo dato de operación: parámetros adicionales.
2.2.1	07/05/2021	Pago webview	Nueva Versión. Nueva variable añadida: Ds_Card_type
2.2.2	01/12/2021	Actualización Compatibilidad	Actualización con nueva versión de Xcode 13.0. Creación de xcframework para soporte en simulador y dispositivos físicos
2.3.0	27/11/2023	Métodos de pago en formato String	Nueva versión. Uso de string para asignar el método de pago.
2.4.0	20/09/2024	Pago webview	Nueva versión. Modificación del flujo de pago por webview para mejorar la seguridad.

ÍNDICE

1. INTRODUCCIÓN	5
1.1.INTRODUCCIÓN	5
1.2.INTEGRACIÓN DE LA LIBRERÍA	5
1.2.1. IMPORTAR LA LIBRERÍA EN EL PROYECTO	5
1.2.2. AÑADIR LOS BINARIOS AL PROYECTO	6
1.2.3. IMPORTAR TPVVINLIBRARY AL PROYECTO	6
2. CONFIGURACIÓN	7
2.1.PARÁMETROS OBLIGATORIOS	7
2.1.1. LICENCIA DE LA APLICACIÓN	7
2.1.2. ENTORNO	7
2.1.3. FUC DEL COMERCIO	8
2.1.4. TERMINAL	8
2.1.5. CÓDIGO DE LA MONEDA UTILIZADA	8
2.2.PARÁMETROS OPCIONALES	8
3. OPERACIONES DISPONIBLES	11
3.1.PAGO DIRECTO	11
3.1.1. CONFIGURACIÓN DEL PAGO DIRECTO	11
3.1.2. PERSONALIZACIÓN DE LA PANTALLA DE PAGO DIRECTO (OPCIONAL)	12
3.1.3. EJEMPLO DE IMPLEMENTACIÓN DE PAGO DIRECTO:	14
3.1.4. EJEMPLO DE IMPLEMENTACIÓN DE PAGO DIRECTO SOLICITUD DE REFERENCIA:	16
3.1.5. EJEMPLO DE IMPLEMENTACIÓN DE PAGO DIRECTO CON REFERENCIA:	18
3.1.6. EJEMPLO DE IMPLEMENTACIÓN DE LA RESPUESTA DEL PAGO DIRECTO:	19
3.2.PAGO WebView	20
3.2.1. CONFIGURACIÓN DEL PAGO WebView	20
3.2.2. EJEMPLO DE IMPLEMENTACIÓN PAGO WebView	21
3.2.3. EJEMPLO DE IMPLEMENTACIÓN PAGO WebView SOLICITUD DE REFERENCIA	24
3.2.4. EJEMPLO DE IMPLEMENTACIÓN PAGO WebView CON REFERENCIA	24
3.2.5. EJEMPLO DE IMPLEMENTACIÓN DE LA RESPUESTA DEL PAGO WebView:	25
3.2.6. CONFIGURACIÓN URLOK, URLOK Y LENGUAJE	26
4. FORMA DE LA RESPUESTA Y CÓDIGOS DE ERROR	27
4.1.RESPUESTA PAGO DIRECTO	27
4.2.RESPUESTA PAGO WebView	29
4.3. CÓDIGOS DE ERROR	31
5. ANEXO	32
5.1.LISTA DE IDIOMAS	32

1. Introducción

1.1.Introducción

El presente documento tiene la finalidad de definir el funcionamiento que contiene el xcframework SDKInApp. Esta librería permite realizar operaciones de comercio eléctrico en aplicaciones de terceros.

El xcframework está preparado para los entornos de integración y producción.

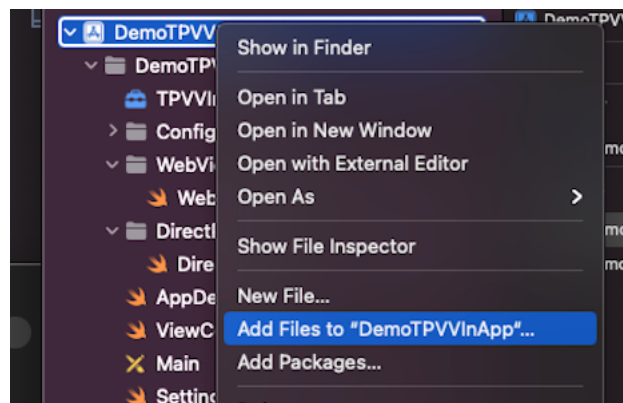
Existen dos tipos de pagos en comercio electrónico: directo y por WebView. El pago directo presenta una pantalla propia del framework, personalizable por el comercio. Esta pantalla recoge los datos de la tarjeta, impidiendo que la aplicación los vea. Este método no está sujeto a autenticación 3DSecure. En cambio, el pago en WebView si permite los diferentes métodos de autenticación.

1.2.Integración de la librería

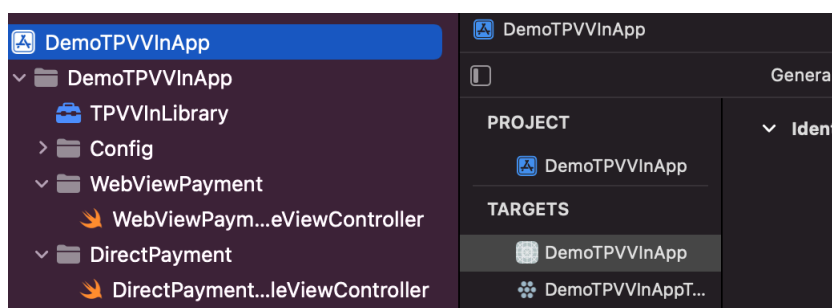
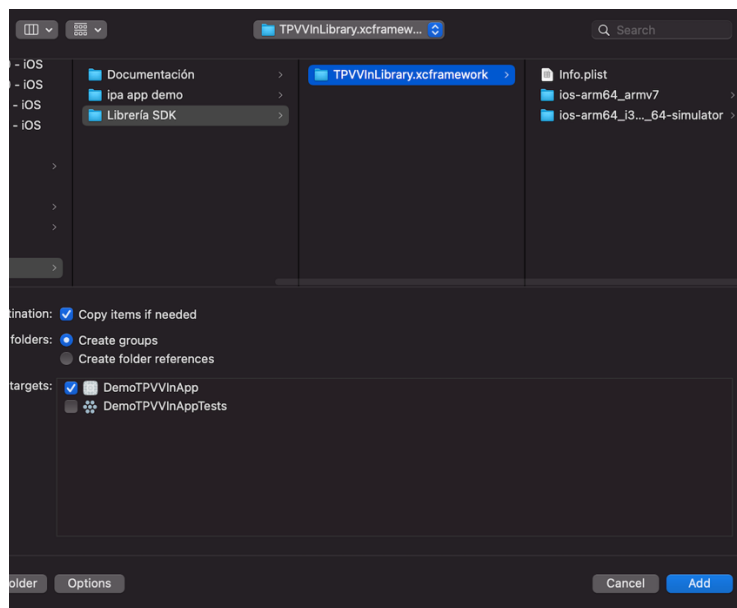
Para incluir en su proyecto la librería es necesario realizar los siguientes pasos:

1.2.1. Importar la librería en el proyecto

En el proyecto, selecciona la opción "Add files to xxx".

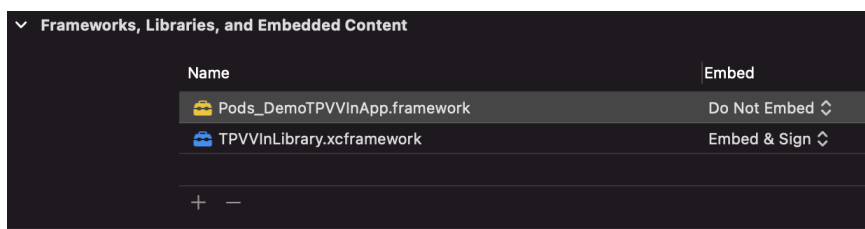


Marcar la opción de "Copy items if needed".



1.2.2. Añadir los binarios al proyecto

En el apartado de "Frameworks, Libraries, and Embedded Content" del Target seleccionar "Embed & Sign".



1.2.3. Importar TPVInLibrary al proyecto

Swift

```
import TPVInLibrary
```

Objective-C

```
#import <TPVInLibrary/TPVInLibrary-Swift.h>
```

2. Configuración

2.1. Parámetros obligatorios

Para utilizar la librería es necesario configurar una serie de parámetros **obligatorios**. Para configurar la librería se utiliza el objeto `TPVVConfiguration`.

2.1.1. Licencia de la aplicación

Es un código alfanumérico proporcionado por Redsys para validar las aplicaciones que hacen uso de la librería.

Swift

```
TPVVConfiguration.shared.appLicense = "XXXXXXXXXX"
```

Objective-C

```
TPVVConfiguration.shared.appLicense = @"XXXXXXXXXX";
```

2.1.2. Entorno

El entorno se selecciona utilizando las constantes definidas en el objeto de configuración `TPVVConfiguration`.

Swift

- Integración:
 - `TPVVConfiguration.shared.appEnvironment = EnvironmentType.Integration`
- Producción:
 - `TPVVConfiguration.shared.appEnvironment = EnvironmentType.Real`
- Test:
 - `TPVVConfiguration.shared.appEnvironment = EnvironmentType.Test`

Objective-C

- Integración:
 - `TPVVConfiguration.shared.appEnvironment = EnvironmentTypeIntegration;`
- Producción:
 - `TPVVConfiguration.shared.appEnvironment = EnvironmentTypeReal;`
- Test:
 - `TPVVConfiguration.shared.appEnvironment = EnvironmentTypeTest;`

2.1.3. Fuc del comercio

Código de identificación del comercio:

Swift

```
TPVVConfiguration.shared.appFuc = "XXXXXXXXXXXX"
```

Objective-C

```
TPVVConfiguration.shared.appFuc = @"XXXXXXXXXX";
```

2.1.4. Terminal

Terminal asociado al número de comercio sobre el que se va a operar.

Swift

```
TPVVConfiguration.shared.appTerminal = "XX"
```

Objective-C

```
TPVVConfiguration.shared.appTerminal = @"XX";
```

2.1.5. Código de la moneda utilizada

Por defecto, es "978" (euros)

Swift

```
TPVVConfiguration.shared.appCurrency = "978"
```

Objective-C

```
TPVVConfiguration.shared.appCurrency = @"978";
```

2.2. Parámetros opcionales

Son parámetros configurables por el comercio, pero que no son obligatorios para el correcto funcionamiento de la librería.

Swift

- Nombre del titular:
 - `TPVVConfiguration.shared.appMerchantTitular = "XXX"`
- Métodos de Pago: Esta configuración solo afecta al pago por WebView.
 - Card: *Forma de pago con tarjeta*

- `TPVVConfiguration.shared.appMerchantPayMethods = PaymentMethod.card`
- Transfer: *Forma de pago por transferencia*
 - `TPVVConfiguration.shared.appMerchantPayMethods = PaymentMethod.transfer`
- Domiciliation: *Forma de pago por domiciliación*
 - `TPVVConfiguration.shared.appMerchantPayMethods = PaymentMethod.domiciliation`
- Paypal: *Forma de pago con paypal*
 - `TPVVConfiguration.shared.appMerchantPayMethods = PaymentMethod.paypal`
- immediatePayment: *Forma de pago con BIZUM*
 - `TPVVConfiguration.shared.appMerchantPayMethods = PaymentMethod.immediatePayment`
- A especificar por el comercio: *String, diferentes formas de pago se encuentran disponibles en <https://pagosonline.redsys.es/parametros-entrada-salida.html>*
 - `TPVVConfiguration.shared.appMerchantPayMethods = "XXX"`
 - `TPVVConfiguration.shared.appMerchantPayMethods = ""` // La pasarela de pago muestra los diferentes métodos en la web, por defecto.
- URL del comercio:
 - `TPVVConfiguration.shared.appMerchantURL = "XXX"`
- Nombre del comercio:
 - `TPVVConfiguration.shared.appMerchantName = "XXX"`
- Datos adicionales del comercio:
 - `TPVVConfiguration.shared.appMerchantData = "XXX"`
- Descripción del comercio:
 - `TPVVConfiguration.shared.appMerchantDescription = "XXX"`
- Fuc del grupo de comercio:
 - `TPVVConfiguration.shared.appMerchantGroup = "XX"`

Objective-C

- Nombre del titular:
 - `TPVVConfiguration.shared.appMerchantTitular = @"XXX";`
- Métodos de Pago: Esta configuración solo afecta al pago por WebView.
 - Card: *Forma de pago con tarjeta*
 - `TPVVConfiguration.shared.appMerchantPayMethods = PaymentMethodCard;`

- Transfer: *Forma de pago por transferencia*
 - `TPVVConfiguration.shared.appMerchantPayMethods = PaymentMethodTransfer;`
- Domiciliation: *Forma de pago por domiciliación*
 - `TPVVConfiguration.shared.appMerchantPayMethods = PaymentMethodDomiciliation;`
- Paypal: *Forma de pago con paypal*
 - `TPVVConfiguration.shared.appMerchantPayMethods = PaymentMethodPaypal;`
- immediatePayment: *Forma de pago con BIZUM*
 - `TPVVConfiguration.shared.appMerchantPayMethods = PaymentMethodImmediatePayment;`
- A especificar por el comercio: *String, diferentes formas de pago se encuentran disponibles en <https://pagosonline.redsys.es/parametros-entrada-salida.html>*
 - `TPVVConfiguration.shared.appMerchantPayMethods = @"XXX";`
 - `TPVVConfiguration.shared.appMerchantPayMethods = @""; // La pasarela de pago muestra los diferentes métodos en la web, por defecto.`
- URL del comercio:
 - `TPVVConfiguration.shared.appMerchantURL = @"XXX";`
- Nombre del comercio:
 - `TPVVConfiguration.shared.appMerchantName = @"XXX";`
- Datos adicionales del comercio:
 - `TPVVConfiguration.shared.appMerchantData = @"XXX";`
- Descripción del comercio:
 - `TPVVConfiguration.shared.appMerchantDescription = @"XXX";`
- Fuc del grupo de comercio:
 - `TPVVConfiguration.shared.appMerchantGroup = @"XXX";`

3. Operaciones disponibles

El framework implementa dos métodos que permiten realizar las siguientes operativas:

- Pago directo: Normal, con solicitud de referencia y con referencia.
- Pago WebView: Normal, con solicitud de referencia y con referencia.

En la operativa “**normal**” se introducen los datos de la tarjeta (número de tarjeta, caducidad y código de seguridad) para realizar el pago.

En la operativa “**con solicitud de referencia**” se introducen los datos de la tarjeta, la diferencia con el pago normal es que en el resultado de la operación el servidor devuelve una **referencia** que se puede utilizar para no tener que introducir de nuevo los datos de la tarjeta en futuros pagos.

Por último, en la operativa “**con referencia**” no es necesario introducir los datos de la tarjeta, se utiliza la **referencia** obtenida anteriormente en un pago “**con solicitud de referencia**”, refiriéndose al segundo tipo de pago, explicado en el párrafo anterior.

3.1. Pago directo

3.1.1. Configuración del pago directo

Para realizar un pago directo es necesario haber configurado previamente los parámetros obligatorios en el objeto `TPVVConfiguration`.

Para cada pago es necesario indicar los siguientes campos:

- `orderNumber` = Código alfanumérico identificativo de la operación (debe ser único) (**obligatorio**)
- `amount` = Importe de la operación (**obligatorio**)
- `identifier` = En caso de no indicar nada realiza un pago normal, en caso de introducir una referencia valida se realiza un pago con referencia, y por último, para realizar un pago con solicitud de referencia hay que utilizar el valor de la constante `TPVVConfiguration.shared.REQUEST_REFERENCE` (**opcional**)
- `productDescription` = Descripción del producto (**opcional**)
- `extraParams`: Diccionario de variables con clave-valor. En caso de querer utilizar parámetros adicionales en la operación se deben introducir tantas como se desee; en caso contrario, dejar a nil (**opcional**)
- `transactionType`: (Utilizando las constantes definidas en `TransactionType`) (**obligatorio**)

Swift

- Normal
 - `TransactionType.normal`
- Pre-autorización:
 - `TransactionType.preauthorization`
- Tradicional:

- `TransactionType.traditional`
- Autenticación:
 - `TransactionType.paymentTypeAuthentication`

Objective-C

- Normal
 - `TransactionTypeNormal;`
 - Pre-autorización:
 - `TransactionTypePreauthorization;`
 - Tradicional:
 - `TransactionTypeTraditional;`
 - Autenticación:
 - `TransactionTypePaymentTypeAuthentication;`
- `uiViewConfig` = Configuración de la vista del pago directo (**obligatorio**)

3.1.2. Personalización de la pantalla de pago directo (opcional)

Es posible personalizar algunos aspectos de la pantalla del pago directo tales como el logo, fondo de pantalla, colores y literales.

Para llevar a cabo la configuración de la pantalla del pago directo se hace uso del objeto del tipo `TPVInAppUIConfig`. Son personalizables los siguientes elementos de la pantalla.

- Logo (Logo)
- Color de fondo de pantalla (`setBackgroundViewColor`)
- Imagen de fondo de pantalla (`setBackgroundImageView`)
- Texto de botón cancelar (`setCancelButtonText`)
- Texto de botón Continuar (`setContinueButtonText`)
- Texto, color y fuente de número de tarjeta (`setNumberCardLabel`)
- Texto color y fuente de fecha de caducidad (`setExpireDateLabel`)
- Texto color y fuente de código de seguridad (`setCVVLabel`)
- Texto color y fuente de texto descriptivo (`setInfoPaymentLabel`)

Configuración de ejemplo:

Creamos la configuración de los diferentes UILabel que contiene la pantalla de pago directo.

Swift

```
let labelCardNumber: PaymentUILabelConfig = PaymentUILabel.create(text: "N.º de tarjeta", textColor:
.black, textFont: .systemFont(ofSize: 16, weight: .semibold))

let labelExpiredDate: PaymentUILabelConfig = PaymentUILabel.create(text: "Caducidad", textColor:
.black, textFont: .systemFont(ofSize: 16, weight: .semibold))

let labelCVC: PaymentUILabelConfig = PaymentUILabel.create(text: "Cód. seguridad", textColor: .black,
textFont: .systemFont(ofSize: 16, weight: .semibold))

let infoPayment: PaymentUILabelConfig = PaymentUILabel.create(text: "Por favor, compruebe que el
número de tarjeta y el resto de datos son exactamente los mismos que aparecen en la tarjeta.",
textColor: .black, textFont: .systemFont(ofSize: 16, weight: .semibold))
```

Objective-C

```
PaymentUILabelConfig *labelCardNumber = [[PaymentUILabelConfig alloc] initWithText:@"Número Card"
textColor:UIColor.blackColor textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

PaymentUILabelConfig *labelExpiredDate = [[PaymentUILabelConfig alloc] initWithText:@"Expiry
date" textColor:UIColor.blackColor textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

PaymentUILabelConfig *labelCVC = [[PaymentUILabelConfig alloc]
initWithText:@"Security code" textColor:UIColor.blackColor
textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

PaymentUILabelConfig *infoPayment = [[PaymentUILabelConfig alloc]
initWithText:@"Payment info" textColor:UIColor.blackColor
textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];
```

Creamos el objeto TPVInAppConfig que contiene toda la configuración de la vista del pago directo. Y llamamos a los métodos de los elementos que queremos personalizar.

Swift

```
let uiConfig = TPVInAppUIConfig()
    .setCancelButtonText("Cancelar")
    .setContinueButtonText("Continuar")
    .setNumberCardLabel(labelCardNumber)
    .setExpireDateLabel(labelExpiredDate)
    .setCVVLabel(labelCVC)
    .setInfoPaymentLabel(infoPayment)
    .setLogo(logo: UIImage(named:"logoEntidad"))
    .setBackgroundImageView(image: UIImage(named:"background"))
    .setBackgroundViewColor(color: UIColor.black)
```

Objective-C

```
TPVInAppUIConfig *config = [TPVInAppUIConfig new];
[config setCancelButtonText:@"Cancel"];
[config setContinueButtonText:@"Continue"];
[config setNumberCardLabel:labelCardNumber];
[config setExpireDateLabel:labelExpiredDate];
[config setCVVLabel:labelCVC];
[config setInfoPaymentLabel:infoPayment];
[config setLogoWithLogo:[UIImage imageNamed:@"appleIcon"]];
[config setBackgroundImageWithImage:[UIImage imageNamed:@"background"]];
[config setBackgroundViewColorWithColor:[UIColor.lightGrayColor];
```

Todos los métodos son opcionales. Si no configura alguno de ellos, el objeto TPVInAppUIConfig utilizará su valor por defecto.

3.1.3. Ejemplo de implementación de pago directo:

Swift

```
let labelCardNumber: PaymentUILabelConfig = PaymentUILabel.create(text: "N.º de tarjeta", textColor:
.black, textFont: .systemFont(ofSize: 16, weight: .semibold))

let labelExpiredDate: PaymentUILabelConfig = PaymentUILabel.create(text: "Caducidad", textColor:
.black, textFont: .systemFont(ofSize: 16, weight: .semibold))

let labelCVC: PaymentUILabelConfig = PaymentUILabel.create(text: "Cód. seguridad", textColor: .black,
textFont: .systemFont(ofSize: 16, weight: .semibold))

let infoPayment: PaymentUILabelConfig = PaymentUILabel.create(text: "Por favor, compruebe que el
número de tarjeta y el resto de datos son exactamente los mismos que aparecen en la tarjeta.",
textColor: .black, textFont: .systemFont(ofSize: 16, weight: .semibold))

// caso sin parámetros adicionales añadidos
let extraParams: [String:String] = nil
// caso con 2 parámetros adicionales añadidos
let extraParams: [String:String] = ["Extra1":"valor1","Extra2":"valor2"]

let uiConfig = TPVInAppUIConfig()
    .setCancelButtonText("Cancelar")
    .setContinueButtonText("Continuar")
    .setNumberCardLabel(labelCardNumber)
    .setExpireDateLabel(labelExpiredDate)
    .setCVVLabel(labelCVC)
    .setInfoPaymentLabel(infoPayment)
    .setLogo(logo: UIImage(named:"logoEntidad"))
    .setBackgroundImage(image: UIImage(named:"background"))
    .setBackgroundViewColor(color: .lightGray)

let dpView = DirectPaymentViewController(orderNumber: order, amount: amount.floatValue,
productDescription: productDescription, transactionType: operationTypeValue, identifier:
paymentIdentifier, extraParams: extraParams, uiViewConfig: uiConfig)

dpView.delegate = self

present(dpView, animated: true, completion: nil)
```

Objective-C

```

PaymentUILabelConfig *labelCardNumber = [[PaymentUILabelConfig alloc]
initWithText:@"Número Card" textColor:UIColor.blackColor textFont:[UIFont
fontWithName:@"HelveticaNeue" size:16]];

    PaymentUILabelConfig *labelExpiredDate = [[PaymentUILabelConfig alloc]
initWithText:@"Expiry date" textColor:UIColor.blackColor textFont:[UIFont
fontWithName:@"HelveticaNeue" size:16]];

    PaymentUILabelConfig *labelCVC = [[PaymentUILabelConfig alloc]
initWithText:@"Security code"
textColor:UIColor.blackColor textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

    PaymentUILabelConfig *infoPayment = [[PaymentUILabelConfig alloc]
initWithText:@"Payment info"
textColor:UIColor.blackColor textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];
    // caso con 2 parámetros adicionales añadidos
    NSMutableDictionary *extraParams = [[NSMutableDictionary alloc]
initWithObjectsAndKeys:@{ @"Extra1" : "valor1",
@"Extra2" : "valor2"}];

    TPVInAppUIConfig *config = [TPVInAppUIConfig new];

    [config setCancelButtonText:@"Cancel"];
    [config setContinueButtonText:@"Continue"];
    [config setNumberCardLabel:labelCardNumber];
    [config setExpireDateLabel:labelExpiredDate];
    [config setCVVLabel:labelCVC];
    [config setInfoPaymentLabel:infoPayment];
    [config setLogoWithLogo:[UIImage imageNamed:@"appleIcon"]];
    [config setBackgroundImageWithImage:[UIImage imageNamed:@"background"]];
    [config setBackgroundViewColorWithColor:UIColor.lightGrayColor];

    DirectPaymentViewController *pay = [[DirectPaymentViewController alloc]
initWithOrderNumber:@"" amount: amount productDescription:@""
transactionType:TransactionTypeNormal identifier:@"" extraParams: extraParams
uiViewConfig:config];

    pay.delegate = self;

    [self presentViewController:pay animated:true completion:nil];

```

3.1.4. Ejemplo de implementación de pago directo con solicitud de referencia:

Swift

```

let labelCardNumber: PaymentUILabelConfig = PaymentUILabel.create(text: "N.º de tarjeta", textColor:
.black, textFont: .systemFont(ofSize: 16, weight: .semibold))

let labelExpiredDate: PaymentUILabelConfig = PaymentUILabel.create(text: "Caducidad", textColor:
.black, textFont: .systemFont(ofSize: 16, weight: .semibold))

let labelCVC: PaymentUILabelConfig = PaymentUILabel.create(text: "Cód. seguridad", textColor: .black,
textFont: .systemFont(ofSize: 16, weight: .semibold))

let infoPayment: PaymentUILabelConfig = PaymentUILabel.create(text: "Por favor, compruebe que el
número de tarjeta y el resto de datos son exactamente los mismos que aparecen en la tarjeta.",
textColor: .black, textFont: .systemFont(ofSize: 16, weight: .semibold))

// caso sin parámetros adicionales añadidos
let extraParams: [String:String] = nil
// caso con 2 parámetros adicionales añadidos
let extraParams: [String:String] = ["Extra1":"valor1","Extra2":"valor2"]

let uiConfig = TPVInAppUIConfig()
    .setCancelButtonText("Cancelar")
    .setContinueButtonText("Continuar")
    .setNumberCardLabel(labelCardNumber)
    .setExpireDateLabel(labelExpiredDate)
    .setCVVLabel(labelCVC)
    .setInfoPaymentLabel(infoPayment)
    .setLogo(logo: UIImage(named:"logoEntidad"))
    .setBackgroundImageView(image: UIImage(named:"background"))
    .setBackgroundViewColor(color: .lightGray)

let dpView = DirectPaymentViewController(orderNumber: order, amount: amount.floatValue,
productDescription: productDescription, transactionType: operationTypeValue, identifier:
TPVVConfiguration.shared.REQUEST_REFERENCE, extraParams: extraParams, uiViewConfig: uiConfig)

dpView.delegate = self

present(dpView, animated: true, completion: nil)

```


Objective-C

```

PaymentUILabelConfig *labelCardNumber = [[PaymentUILabelConfig alloc] initWithText:@"Número Card"
textColor:UIColor.blackColor textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

PaymentUILabelConfig *labelExpiredDate = [[PaymentUILabelConfig alloc] initWithText:@"Expiry
date" textColor:UIColor.blackColor textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

PaymentUILabelConfig *labelCVC = [[PaymentUILabelConfig alloc]
initWithText:@"Security code" textColor:UIColor.blackColor
textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

PaymentUILabelConfig *infoPayment = [[PaymentUILabelConfig alloc]
initWithText:@"Payment info" textColor:UIColor.blackColor
textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

// caso con 2 parámetros adicionales añadidos
NSMutableDictionary *extraParams = [[NSMutableDictionary alloc]
initWithObjectsAndKeys:@{ @"Extra1" : "valor1",
@"Extra2" : "valor2"}];

TPVInAppUIConfig *config = [TPVInAppUIConfig new];

[config setCancelButtonText:@"Cancel"];
[config setContinueButtonText:@"Continue"];
[config setNumberCardLabel:labelCardNumber];
[config setExpireDateLabel:labelExpiredDate];
[config setCVVLabel:labelCVC];
[config setInfoPaymentLabel:infoPayment];
[config setLogoWithLogo:[UIImage imageNamed:@"appleIcon"]];
[config setBackgroundImageWithImage:[UIImage imageNamed:@"background"]];
[config setBackgroundViewColorWithColor:UIColor.lightGrayColor];

DirectPaymentViewController *pay = [[DirectPaymentViewController alloc] initWithOrderNumber:@""
amount: amount productDescription:@"" transactionType:TransactionTypeNormal identifier:
TPVVConfiguration.shared.REQUEST_REFERENCE extraParams: extraParams uiViewConfig:config];

pay.delegate = self;

[self presentViewController:pay animated:true completion:nil];

```

3.1.5. Ejemplo de implementación de pago directo con referencia:

Swift

```
let labelCardNumber: PaymentUILabelConfig = PaymentUILabel.create(text: "N.º de tarjeta", textColor:
.black, textFont: .systemFont(ofSize: 16, weight: .semibold))

let labelExpiredDate: PaymentUILabelConfig = PaymentUILabel.create(text: "Caducidad", textColor:
.black, textFont: .systemFont(ofSize: 16, weight: .semibold))

let labelCVC: PaymentUILabelConfig = PaymentUILabel.create(text: "Cód. seguridad", textColor: .black,
textFont: .systemFont(ofSize: 16, weight: .semibold))

let infoPayment: PaymentUILabelConfig = PaymentUILabel.create(text: "Por favor, compruebe que el
número de tarjeta y el resto de datos son exactamente los mismos que aparecen en la tarjeta.",
textColor: .black, textFont: .systemFont(ofSize: 16, weight: .semibold))

// caso sin parámetros adicionales añadidos
let extraParams: [String:String] = nil
// caso con 2 parámetros adicionales añadidos
let extraParams: [String:String] = ["Extra1":"valor1","Extra2":"valor2"]

let uiConfig = TPVInAppUIConfig()
    .setCancelButtonText("Cancelar")
    .setContinueButtonText("Continuar")
    .setNumberCardLabel(labelCardNumber)
    .setExpireDateLabel(labelExpiredDate)
    .setCVVLabel(labelCVC)
    .setInfoPaymentLabel(infoPayment)
    .setLogo(logo: UIImage(named:"logoEntidad"))
    .setBackgroundImageView(image: UIImage(named:"background"))
    .setBackgroundViewColor(color: .lightGray)

let dpView = DirectPaymentViewController(orderNumber: order, amount: amount.floatValue,
productDescription: productDescription, transactionType: operationTypeValue, identifier:
self.reference, extraParams: extraParams, uiViewConfig: uiConfig)

dpView.delegate = self

present(dpView, animated: true, completion: nil)
```

Objective-C

```
PaymentUILabelConfig *labelCardNumber = [[PaymentUILabelConfig alloc] initWithText:@"Número Card"
textColor:UIColor.blackColor textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

PaymentUILabelConfig *labelExpiredDate = [[PaymentUILabelConfig alloc] initWithText:@"Expiry
date" textColor:UIColor.blackColor textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

PaymentUILabelConfig *labelCVC = [[PaymentUILabelConfig alloc]
initWithText:@"Security code" textColor:UIColor.blackColor
textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

PaymentUILabelConfig *infoPayment = [[PaymentUILabelConfig alloc]
initWithText:@"Payment info" textColor:UIColor.blackColor
textFont:[UIFont fontWithName:@"HelveticaNeue" size:16]];

// caso con 2 parámetros adicionales añadidos
NSMutableDictionary *extraParams = [[NSMutableDictionary alloc]
initWithObjectsAndKeys:@{ @"Extra1" : "valor1",
@"Extra2" : "valor2"}];

TPVInAppUIConfig *config = [TPVInAppUIConfig new];

[config setCancelButtonText:@"Cancel"];
[config setContinueButtonText:@"Continue"];
[config setNumberCardLabel:labelCardNumber];
[config setExpireDateLabel:labelExpiredDate];
[config setCVVLabel:labelCVC];
[config setInfoPaymentLabel:infoPayment];
[config setLogoWithLogo:[UIImage imageNamed:@"appleIcon"]];
[config setBackgroundImageWithImage:[UIImage imageNamed:@"background"]];
[config setBackgroundViewColorWithColor:UIColor.lightGrayColor];

DirectPaymentViewController *pay = [[DirectPaymentViewController alloc] initWithOrderNumber:@""
amount: amount productDescription:@"" transactionType:TransactionTypeNormal identifier:
self.reference extraParams: extraParams
uiViewConfig:config];

pay.delegate = self;

[self presentViewController:pay animated:true completion:nil];
```

3.1.6. Ejemplo de implementación de la respuesta del pago directo:

Para poder obtener la respuesta de la operación deberá implementar el siguiente delegado `DirectPaymentResponseDelegate`.

Swift

```
extension DirectPaymentTableViewController: DirectPaymentResponseDelegate {
    func responseDirectPaymentK0(response: (DirectPaymentResponseK0)) {
    }

    func responseDirectPaymentOK(response: (DirectPaymentResponseOK)) {
    }
}
```

Objective-C

```
@interface ViewController () <DirectPaymentResponseDelegate>
@end

- (void)responseDirectPaymentOKWithResponse:(DirectPaymentResponseOK *)response {
}

- (void)responseDirectPaymentK0WithResponse:(DirectPaymentResponseK0 *)response{
}
```

3.2.Pago WebView

3.2.1. Configuración del pago WebView

Para realizar un pago a través de WebView es necesario haber configurado previamente los parámetros obligatorios en el objeto TPVVConfiguration

Para cada pago es necesario indicar los siguientes campos:

- orderNumber = Código alfanumérico identificativo de la operación (debe ser único) **(obligatorio)**
- amount = Importe de la operación **(obligatorio)**
- identifier = En caso de no indicar nada (cadena vacía) se realiza un pago normal, en caso de introducir una referencia valida se realiza un pago con referencia y para realizar un pago con solicitud de referencia hay que utilizar el valor de la constante `TPVVConfiguration.shared.REQUEST_REFERENCE` **(opcional)**
- productDescription = Descripción del producto **(opcional)**
- extraParams: Diccionario de variables con clave-valor. En caso de querer utilizar parámetros adicionales en la operación se deben introducir tantas como se desee; en caso contrario, dejar a nil **(opcional)**
- transactionType: (Utilizando las constantes definidas en TransactionType) **(obligatorio)**

Swift

- Normal
 - `TransactionType.normal`
- Pre-autorización:
 - `TransactionType.preauthorization`
- Tradicional:
 - `TransactionType.traditional`
- Autenticación:
 - `TransactionType.paymentTypeAuthentication`

Objective-C

- Normal
 - `TransactionTypeNormal;`
- Pre-autorización:
 - `TransactionTypePreauthorization;`
- Tradicional:
 - `TransactionTypeTraditional;`
- Autenticación:
 - `TransactionTypePaymentTypeAuthentication;`

3.2.2. Ejemplo de implementación pago WebView

Para realizar un pago a través de WebView es necesario seguir una serie de pasos, entre los cuales se harán varias llamadas a funciones de la clase pública SDK_INAPP de la sdk:

- 3.2.2.1.

En primer lugar, es necesario llamar a la función “initializeWebPayment” pasando los parámetros indicados en el punto anterior (3.2.1), lo cual devolverá:

- Una cadena con la codificación en base 64 de los parámetros pasados en la llamada a la función junto a otros parámetros necesarios que almacena la propia SDK si ha ido todo bien.
- Un objeto `WebViewPaymentResponseKO`, que contiene un parámetro `code` (Int) y un parámetro `desc` (String) si se ha producido algún error, o si el importe no es válido. Será decisión del comercio que hacer en caso de recibir un error.

La cadena es necesaria para continuar con el flujo de pago, por lo que si se devuelve algún caso de error, se deberá volver a intentar generar, revisando que los parámetros de entrada sean correctos.

Swift

```
// caso sin parámetros adicionales añadidos
let extraParams: [String:String] = nil
// caso con 2 parámetros adicionales añadidos
let extraParams: [String:String] = ["Extra1":"valor1","Extra2":"valor2"]
let cadena64: String = []
SDK_INAPP.initializeWebPayment(orderNumber: orderNumber, amount: amount, productDescription:
productDescription, transactionType: operationTypeValue, identifier: identifier, extraParams:
extraParams, success: { response in
    self.cadena64 = response
}, failure: { error in
    //usar objeto de error
})
```

Objective-C

```
// caso sin parámetros adicionales añadidos
NSDictionary<NSString *, NSString *> *extraParamsNil = nil;

// caso con 2 parámetros adicionales añadidos
NSDictionary<NSString *, NSString *> *extraParams = @{@"Extra1": @"valor1", @"Extra2": @"valor2"};

__block NSMutableString *cadena64 = [[NSMutableString alloc] init];

[SDK_INAPP initializeWebPaymentWithOrderNumber: orderNumber amount: amount productDescription:
productdescription transactionType:TransactionTypeNormal identifier:identifier
extraParams:extraParams success:^(id response) {
    cadena64 = [NSMutableString stringWithFormat:@"%@", response];
}
failure:^(WebViewPaymentResponseK0 *error) {
    //usar objeto de error
}];
```

3.2.2.2.

Una vez obtenida la cadena en base64, será responsabilidad del comercio generar una firma segura a partir de esta cadena en base 64.

Para calcular la firma es necesario utilizar una clave específica para cada terminal. Se puede obtener la clave accediendo al Portal de Administración del Tpv virtual, opción Consulta datos de Comercio, en el apartado “Ver clave”, tal y como se muestra en la siguiente imagen:



Una vez se tiene la cadena de datos en base 64 y la clave específica del terminal, se debe calcular la firma siguiendo los siguientes pasos:

- Se genera una clave específica por operación. Para obtener la clave derivada a utilizar en una operación se debe realizar un cifrado 3DES entre la clave del comercio, la cual debe ser previamente decodificada en base 64, y el valor del número de pedido de la operación (Ds_Merchant_Order).
- Se calcula el HMAC SHA256 de la cadena en base 64 y la clave obtenida en el paso anterior.
- El resultado obtenido se codifica en base 64, y el resultado de la codificación será el valor de la firma.

Como ejemplo práctico, supondremos que tras generar la firma, se almacena en una variable “firma” de tipo cadena.

- 3.2.2.3.

Una vez obtenida la firma, se hará una segunda llamada a la función “doWebViewPayment” pasando como parámetros la firma generada y el tipo de versión de la firma. Suponiendo como ejemplo el uso de SHA256, este último parámetro sería “HMAC_SHA256_V1”. La función devolverá un controlador inicializado de tipo “WebViewPaymentController”, siendo responsabilidad del comercio mostrar este controller de la forma que deseen.

Swift

```
webView = SDK_INAPP.doWebViewPayment(signature: firma, signatureVersion: "HMAC_SHA256_V1")
webView.delegate = self

self.present(webView, animated: true, completion: nil)
```

Objective-C

```
WebViewPaymentController *webView = [SDK_INAPP doWebViewPaymentWithSignature:firma
signatureVersion:@"HMAC_SHA256_V1"];

webView.delegate = self;
[self presentViewController:webView animated:YES completion:nil];
```

En la versión actual de la sdk, los valores soportados para el parámetro “signatureVersion” son:

- HMAC_SHA256_V1
- HMAC_SHA512_V1

En caso de no especificar ninguno, se utilizará “HMAC_SHA256_V1” como valor por defecto.

3.2.3. Ejemplo de implementación pago WebView con solicitud de referencia

Para realizar el pago con solicitud de referencia, el proceso sería exactamente el mismo, pero incluyendo el código de la solicitud de referencia en la llamada a la función para generar la cadena codificada en base64.

Swift

```
SDK_INAPP.inicializeWebPayment(orderNumber: orderNumber, amount: amount, productDescription:
productDescription, transactionType: operationTypeValue, identifier:
"TPVVConfiguration.shared.REQUEST_REFERENCE", extraParams: extraParams, success: { response in
    self.cadena64 = response
}, failure: { error in
    //usar objeto de error
})
```

Objective-C

```
SDK_INAPP inicializeWebPaymentWithOrderNumber: orderNumber amount: amount productDescription:
productDescription transactionType:TransactionTypeNormal identifier:
TPVVConfiguration.shared.REQUEST_REFERENCE extraParams:extraParams success:^(id response) {
    cadena64 = [NSMutableString stringWithFormat:@"%@", response];
}
failure:^(WebViewPaymentResponseK0 *error) {
    //usar objeto de error
}];
```

3.2.4. Ejemplo de implementación pago WebView con referencia

Para realizar el pago con referencia, el proceso sería exactamente el mismo, pero incluyendo el código de la referencia en la llamada a la función para generar la cadena codificada en base64.

Swift

```
SDK_INAPP.inicializeWebPayment(orderNumber: ordernumber, amount: amount, productDescription:
productDescription, transactionType: operationTypeValue, identifier: self.reference, extraParams:
extraParams, success: { response in
    self.cadena64 = response
}, failure: { error in
    //usar objeto de error
})
```

Objective-C

```
SDK_INAPP inicializeWebPaymentWithOrderNumber: orderNumber amount: amount productDescription:
productDescription transactionType:TransactionTypeNormal identifier: self.reference
extraParams:extraParams success:^(id response) {
    cadena64 = [NSMutableString stringWithFormat:@"%@", response];
}
failure:^(WebViewPaymentResponseK0 *error) {
    //usar objeto de error
}];
```


3.2.5. Ejemplo de implementación de la respuesta del pago WebView:

Para poder obtener la respuesta de la operación deberá implementar el siguiente delegado `WebViewPaymentResponseDelegate`.

Swift

```
extension WebViewPaymentTableViewController: WebViewPaymentResponseDelegate {  
    func responsePaymentK0(response: (WebViewPaymentResponseK0)) {  
    }  
    func responsePaymentOK(response: (WebViewPaymentResponseOK)) {  
    }  
}
```

Objective-C

```
@interface ViewController () <WebViewPaymentResponseDelegate>  
  
@end  
- (void)responsePaymentK0WithResponse:(WebViewPaymentResponseK0 *)response{  
}  
- (void)responsePaymentOKWithResponse:(WebViewPaymentResponseOK *)response {  
}
```

3.2.6. Configuración URLOK, URLOK y lenguaje

En el caso del WebView además es posible configurar el idioma y las url de resultado en caso de que la operación se haya realizado correctamente o con algún error.

Swift

```
TPVVConfiguration.shared.appURL0K = "XXX"  
TPVVConfiguration.shared.appURLK0 = "XXX"  
  
TPVVConfiguration.shared.appMerchantConsumerLanguage = "X"
```

Objective-C

```
TPVVConfiguration.shared.appURLK0 = @"XXX";  
TPVVConfiguration.shared.appURL0K = @"XXX";  
  
TPVVConfiguration.shared.appMerchantConsumerLanguage = @"X";
```

En caso de indicar una URL de resultado, una vez se produzca la redirección, sería necesario pulsar sobre el botón “Atrás” para así cerrar el WebView y volver al flujo principal de la aplicación.

En caso de no indicar una URL de resultado, una vez terminada la operación se vuelve al flujo principal de la aplicación sin hacer ninguna redirección. Por otro lado, para los lenguajes hay que indicar un código de idioma válido, en caso de no indicar nada se configura por defecto en español.

*En el anexo se incluye una tabla con los diferentes códigos de idioma.

En caso de querer cancelar la operación, sería necesario pulsar sobre el botón “Cancelar” para así cerrar el WebView.

4. Forma de la respuesta y códigos de error

4.1.Respuesta pago directo

Para el pago directo (`DirectPaymentViewController`) tenemos dos objetos respuesta `DirectPaymentResponseOK` y `DirectPaymentResponseKO` que contienen los siguientes parámetros:

Swift

`DirectPaymentResponseOK`

```
public var code: Int
public var desc: String
public var Ds_Card_Country: String
public var Ds_Amount:String
public var Ds_MerchantData:String
public var Ds_Currency:String
public var Ds_Order:String
public var Ds_MerchantCode:String
public var Ds_Card_Type:String
public var Ds_Card_Brand:String
public var Ds_CardNumber:String
public var Ds_AuthorisationCode:String
public var Ds_Language:String
public var Ds_SecurePayment:String
public var Ds_Response:String
public var Ds_TransactionType:String
public var Ds_Terminal:String
public var Ds_ExpiryDate:String
public var Ds_Merchant_Identifier:String
public var Ds_Extra_Params:[String:String]
```

`DirectPaymentResponseKO`

```
public var code:Int
public var desc:String
```

Objective-C

DirectPaymentResponseOK

```

@interface DirectPaymentResponseOK : NSObject
@property (nonatomic) NSInteger code;
@property (nonatomic, copy) NSString * _Nonnull desc;
@property (nonatomic, copy) NSString * _Nonnull Ds_Card_Country;
@property (nonatomic, copy) NSString * _Nonnull Ds_Amount;
@property (nonatomic, copy) NSString * _Nonnull Ds_MerchantData;
@property (nonatomic, copy) NSString * _Nonnull Ds_Currency;
@property (nonatomic, copy) NSString * _Nonnull Ds_Order;
@property (nonatomic, copy) NSString * _Nonnull Ds_MerchantCode;
@property (nonatomic, copy) NSString * _Nonnull Ds_Card_Type;
@property (nonatomic, copy) NSString * _Nonnull Ds_Card_Brand;
@property (nonatomic, copy) NSString * _Nonnull Ds_CardNumber;
@property (nonatomic, copy) NSString * _Nonnull Ds_AuthorisationCode;
@property (nonatomic, copy) NSString * _Nonnull Ds_Language;
@property (nonatomic, copy) NSString * _Nonnull Ds_SecurePayment;
@property (nonatomic, copy) NSString * _Nonnull Ds_Response;
@property (nonatomic, copy) NSString * _Nonnull Ds_TransactionType;
@property (nonatomic, copy) NSString * _Nonnull Ds_Terminal;
@property (nonatomic, copy) NSString * _Nonnull Ds_ExpiryDate;
@property (nonatomic, copy) NSString * _Nonnull Ds_Merchant_Identifier;
@property (nonatomic, copy) NSMutableDictionary * _Nonnull Ds_Extra_Params;

```

DirectPaymentResponseKO

```

@interface DirectPaymentResponseKO : NSObject
@property (nonatomic) NSInteger code;
@property (nonatomic, copy) NSString * _Nonnull desc;

```

4.2.Respuesta pago WebView

Para el pago WebView (`WebViewPaymentController`) tenemos dos objetos respuesta `WebViewPaymentResponseOK` y `WebViewPaymentResponseKO` que contienen los siguientes parámetros:

Swift

`WebViewPaymentResponseOK`

```
public var code: Int
public var desc: String
public var Ds_Date:String
public var Ds_Hour:String
public var Ds_SecurePayment:String
public var Ds_Amount:String
public var Ds_Currency:String
public var Ds_Order:String
public var Ds_MerchantCode:String
public var Ds_Terminal:String
public var Ds_Response:String
public var Ds_Signature:String
public var Ds_TransactionType:String
public var Ds_MerchantData:String
public var Ds_AuthorisationCode:String
public var Ds_ExpiryDate:String
public var Ds_Merchant_Identifier:String
public var Ds_ConsumerLanguage:String
public var Ds_Card_Country:String
public var Ds_Card_Brand:String
public var Ds_Card_Number:String
public var Ds_Card_Type:String
public var Ds_Extra_Params:[String:String]
```

`WebViewPaymentResponseKO`

```
public var code: Int
public var desc: String
```

Objective-C

WebViewPaymentResponseOK

```

@interface WebViewPaymentResponseOK : NSObject
@property (nonatomic) NSInteger code;
@property (nonatomic, copy) NSString * _Nonnull desc;
@property (nonatomic, copy) NSString * _Nonnull Ds_Date;
@property (nonatomic, copy) NSString * _Nonnull Ds_Hour;
@property (nonatomic, copy) NSString * _Nonnull Ds_SecurePayment;
@property (nonatomic, copy) NSString * _Nonnull Ds_Amount;
@property (nonatomic, copy) NSString * _Nonnull Ds_Currency;
@property (nonatomic, copy) NSString * _Nonnull Ds_Order;
@property (nonatomic, copy) NSString * _Nonnull Ds_MerchantCode;
@property (nonatomic, copy) NSString * _Nonnull Ds_Terminal;
@property (nonatomic, copy) NSString * _Nonnull Ds_Response;
@property (nonatomic, copy) NSString * _Nonnull Ds_Signature;
@property (nonatomic, copy) NSString * _Nonnull Ds_TransactionType;
@property (nonatomic, copy) NSString * _Nonnull Ds_MerchantData;
@property (nonatomic, copy) NSString * _Nonnull Ds_AuthorisationCode;
@property (nonatomic, copy) NSString * _Nonnull Ds_ExpiryDate;
@property (nonatomic, copy) NSString * _Nonnull Ds_Merchant_Identifier;
@property (nonatomic, copy) NSString * _Nonnull Ds_ConsumerLanguage;
@property (nonatomic, copy) NSString * _Nonnull Ds_Card_Country;
@property (nonatomic, copy) NSString * _Nonnull Ds_Card_Brand;
@property (nonatomic, copy) NSString * _Nonnull Ds_Card_Number;
@property (nonatomic, copy) NSString * _Nonnull Ds_Card_Type;
@property (nonatomic, copy) NSMutableDictionary * _Nonnull Ds_Extra_Params;

```

WebViewPaymentResponseK0

```

@interface WebViewPaymentResponseK0 : NSObject
@property (nonatomic) NSInteger code;
@property (nonatomic, copy) NSString * _Nonnull desc;

```

4.3. Códigos de error

Los códigos de error propios de la librería móvil son los siguientes:

Código	Descripción
11	La firma del mensaje no es correcta
29	Fallo en la criptografía del servicio
31	La aplicación es incorrecta
60	Formato de JSON incorrecto
61	Error al obtener la clave de firma del comercio
62	Tipo de firma del terminal no soportado
78	Error propio del TPV virtual de Redsys
5550	Error de conexión
5548	Error interno de la librería

* Los errores con código 78 contienen en su descripción el error correspondiente del TPV virtual de Redsys. Para más información consultar el documento: "TPV-Virtual Manual Integración – Redirección"

5. Anexo

5.1. Lista de idiomas

Código	Idioma
0	Por defecto
1	Español
2	English - Ingles
3	Català
4	Français - Frances
5	Deutsch - Aleman
6	Nederlands - Holandes
7	Italiano
8	Svenska - Sueco
9	Português
10	Valencià
11	Polski - Polaco
12	Galego
13	Euskara
100	български език - Bulgaro (Bulgaro)
156	Chino
191	Hrvatski - Croata (Croatian)
203	Čeština - Checo (čeština Czech)
208	Dansk - Danes
233	Eesti keel - Estonio (Estonian)
246	Suomi - Finlandes (Finnish)
300	ελληνικά - Griego (Greek)
348	Magyar - Hungaro (Hungarian)
392	Japonés
428	Latviešu valoda - Leton (Latvian Latviešu valoda)
440	Lietuvių kalba - Lituano (Lithuanian u)
470	Malti - Maltes (Maltese)
642	Română - Rumano (Romanian româna)

643	ру́сский язы́к – Ruso (Russkiy)
703	Slovenský jazyk - Eslovaco (Slovak slovenský jazyk)
705	Slovenski jezik - Esloveno (Slovenian)
792	Türkçe - Turco

Ejemplo de configuración appMerchantConsumerLanguage:

Swift

```
TPVVConfiguration.shared.appMerchantConsumerLanguage = "4" //idioma seleccionado  
francés
```

Objective-C

```
TPVVConfiguration.shared.appMerchantConsumerLanguage = @"4"; //idioma seleccionado  
francés
```

*La configuración del idioma únicamente afecta al pago por WebView. Para el pago directo habría que personalizar la pantalla con las traducciones de los textos correspondientes.