TPV-Virtual

Apple Pay, Guía de Instalación de los Comercios

Versión: 3.2

Fecha: 28/03/2023

Referencia: RS.SO.SAD.MAN.0022

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

www.redsys.es

Autorizaciones y control de versión

Versión	Fecha	Afecta	Breve descripción del cambio
1.0	14/06/23	TODO	Versión inicial del documento
1.1	17/02/20	Enrolamiento	Información adicional sobre sobre el Merchant ID y el certificado de procesamiento de pagos
2.0	08/03/2021	TODO	Actualización de procedimiento
2.1	30/12/2021	Punto 3	Se añade una nueva forma de conexión para el comercio con el TPV Virtual enviando los datos decodificados.
2.2	08/08/2022	Punto 1	Se especifica que el comercio debe hacer la integración con ApplePay
3.1	09/03/2023	Todo el documento	
3.2	28/03/2023	Nuevo apartado 2	Integración por redirección

ÍNDICE

1. INTRODUCCIÓN 4

2. INTEGRACIÓN DE APPLE PAY EN REDIRECCIÓN 5

3. INTEGRACIÓN PROPIA APPLE PAY 8

3.1. ENROLAMIENTO DEL COMERCIO CON APPLE PAY 8

3.2. INTEGRACIÓN DE APPLE PAY 15

3.2.1. MOSTRAR BOTÓN APPLE PAY 15

3.2.2. CREAR APPLEPAYSESSION 16

3.2.3. OBTENER TOKEN APPLE PAY 17

3.2.4. AUTORIZACIÓN DEL PAGO (INTEGRACIÓN REDSYS) 21

3.3. CONFIGURACIÓN DEL TPV VIRTUAL 23

3.4. FUNCIONALIDAD AVANZADA. ENVÍO DE DATOS DESCIFRADOS EN COMERCIO 24

1. INTRODUCCIÓN

Este documento tiene como objeto definir y facilitar la integración de los comercios con Apple Pay en el TPV Virtual de Redsys.

Se dispone de dos tipos de integración con Apple Pay:

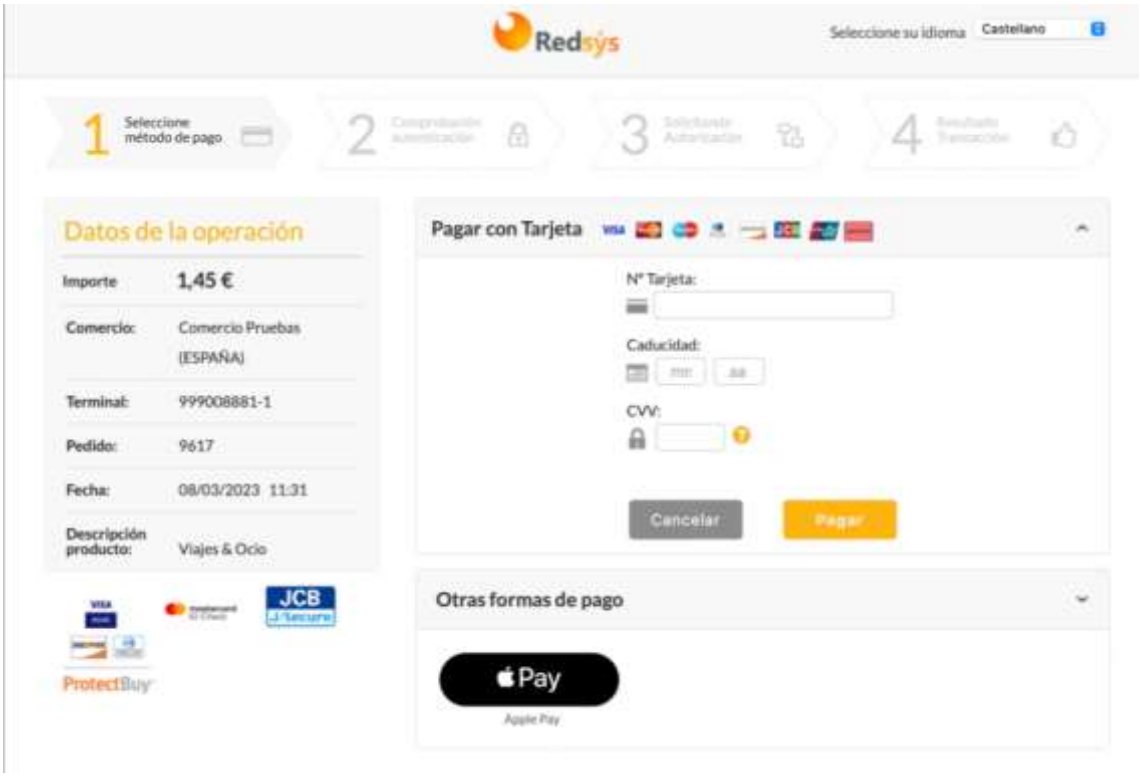
1. Delegada en Redsys. El botón de Apple Pay se mostrará en la página de pago del TPV Virtual. Integración por redirección. El comercio no debe realizar ninguna integración. Esta opción solo requiere de configuración del método de pago por parte de la entidad.
2. Integración propia del comercio en Apple Pay. El comercio será el encargado de realizar la integración del botón de Apple Pay en su página, el cual permitirá obtener el objeto paymentData que será el que tendrá que comunicar al TPV-Virtual de Redsys para el procesamiento de la transacción.

**Esta integración sólo se puede realizar en el entorno de producción del TPV Virtual.

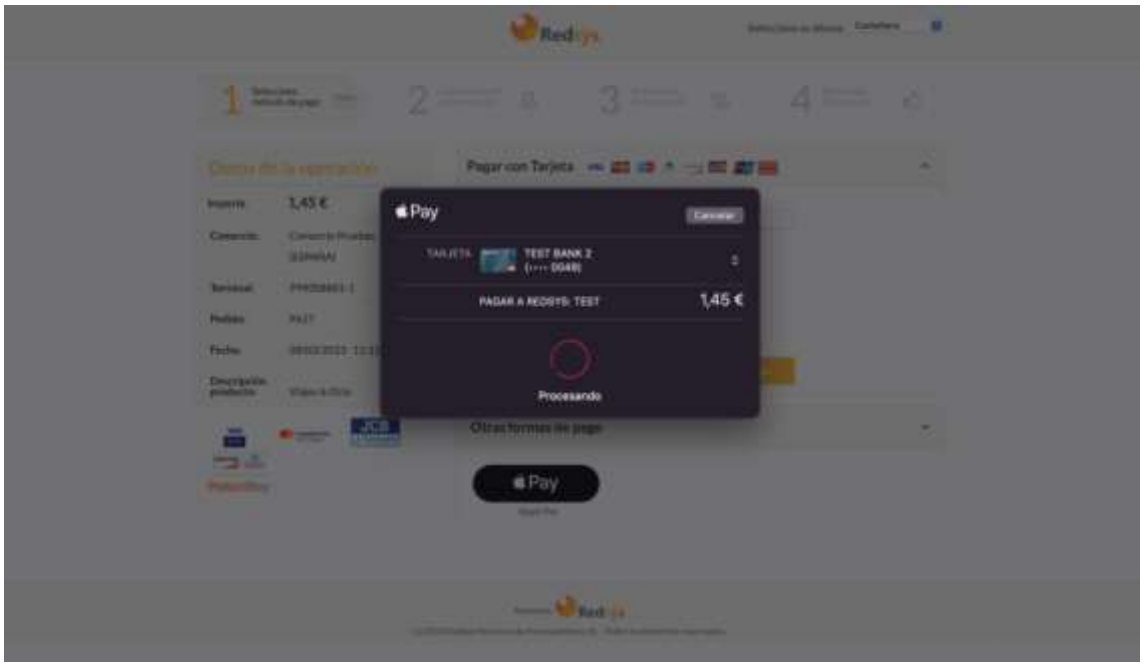
2. INTEGRACIÓN DE APPLE PAY EN REDIRECCIÓN

Puedes empezar a aceptar pagos con Apple Pay de forma automática en el TPV Virtual solicitando la activación del método de pago y el tipo de integración a tu entidad bancaria. **No es necesario ningún desarrollo técnico.**

El método de pago será seleccionable en el interfaz del TPV Virtual:



Al pulsar sobre él, se mostrarán las tarjetas asociadas a la cuenta de Apple del dispositivo y se procederá a seleccionar la tarjeta con la que realizar el pago:



Para la realización del pago Apple lanzará la autenticación del titular en el dispositivo (huella, faceID o contraseña).

Si se quiere forzar la ejecución del método de pago Apple Pay (en lugar de mostrar la página de introducción de datos de tarjeta), se debe enviar en el campo DS_MERCHANT_PAYMETHODS el valor 'xpay' (sin comillas). En caso de tener configurado también Google Pay, aparecerá el botón de Google Pay junto al de Apple Pay.

3. INTEGRACIÓN PROPIA APPLE PAY

Para integrar Apple Pay en tu app o sitio web, debes seguir la guía de integración de Apple Pay.

https://developer.apple.com/documentation/apple_pay_on_the_web

Como facilidad para el comercio os indicamos los siguientes puntos a tener en cuenta

El comercio deberá registrarse en la plataforma de Apple Pay, como resultado, se generará un identificador de comercio (merchantId) y un certificado cuya clave privada se utilizará para el posterior desencriptado de los datos del medio de pago.

Para poder realizar la integración con Apple Pay será necesario configurar el servidor del comercio. Esta información podemos consultarla en el siguiente enlace:

https://developer.apple.com/documentation/apple_pay_on_the_web/setting_up_your_server

3.1. Enrolamiento del Comercio con Apple Pay

Creación de la cuenta de Apple

Crear o añade a una cuenta Apple Pay las opciones de desarrollador. (Apple requiere un coste anual).

https://developer.apple.com/help/account/

Creación de merchantID.

Acceder a la Consola de Apple y crear un nuevo MerchantId que identifique al Comercio. Para registrar un Merchant ID:

1. Seleccionar Certificates, Identifiers & Profiles.
2. A la derecha seleccionar Merchant IDs.
3. Clic en el botón de añadir (+).

4. A Continuación, pulse en "*Continue*".

5. Escribe una descripción y un identificador, y haz clic en "*Continue*".

6. Revise la configuración y haga clic en Register.

Más información en
https://developer.apple.com/documentation/applepaywebmerchantregistrationapi/registering_with_apple_pay_and_applying_to_use_the_api

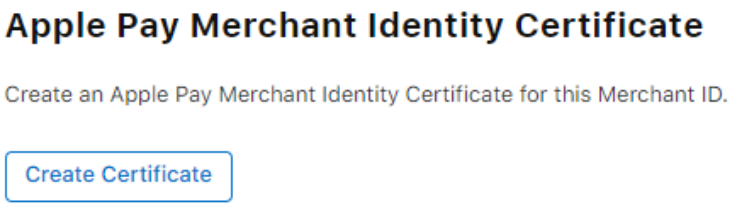
Creación de Merchant Identity Certificate.

1. Se genera una clave privada RSA2048 y una solicitud de firma de certificado (CSR) con dicha clave utilizando la herramienta OpenSSL:

   openssl req -new -newkey rsa:2048 -nodes -keyout merchant_id.key -out merchant_id.csr -subj '/O=NombreComercio /C=ES'

2. Esta acción generará un fichero merchant_ir.csr que deberá utilizarse para generar el certificado final por parte de Apple
3. Acceda al perfil del MerchantId generado en el apartado anterior.

4. En la parte inferior de la página, haga clic en "*Create Certificate*" bajo el título "*Apple Pay Merchant Identity Certificate*"



5. En "*Choose File*" cargue el archivo *merchant_id.csr* y haga clic en "*Continue*"

6. Haga clic en "*Download*" para descargar un archivo llamado *merchant_id.cer*

7. Se debe convertir el archivo *.cer en un archivo *.pem usando el siguiente comando en la herramienta OpenSSL:

   openssl x509 -inform der -in merchant_id.cer -out merchant_id.pem

   Generación del Certificate Signing Request (CSR).

   Una vez generado el Merchant Id y el Merchant Identity Certificate, es necesario generar un Certificate Signing Request utilizado para el descifrado de la mensajería a partir de una

clave privada. Dicha clave Privada será la que el comercio deba configurar en el Portal de administración del TPV Virtual.

1. Generar par de claves en un archivo de claves de curva elíptica con la herramienta OpenSSL:

   openssl ecparam -genkey -name prime256v1 -out ecckey.key

2. Formatear la clave a formato PKCS8:

   openssl pkcs8 -topk8 -inform PEM -outform PEM -nocrypt -in ecckey.key -out ecckey_pkcs8.key

   ***El contenido del fichero resultante ecckey_pkcs8.key deberá introducirse, junto al merchantId, en el portal de administración del TPV Virtual.**

3. Generar CSR a partir del par de claves en el archivo de claves

   openssl req -new -sha256 -key ecckey.key -out ecccertreq.csr -subj /CN=www.dominiocomercio.com

4. Acceda al perfil del MerchantId generado en el apartado anterior.
5. En la sección Certificados de procesamiento de pagos, haga clic en "*Create Certificate*".

6. Siga las instrucciones para obtener o generar su solicitud de firma de certificado (CSR) y haga clic en "*Continue*".

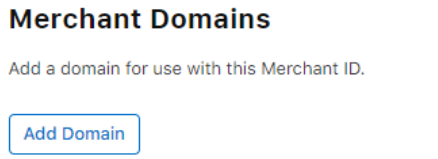
7. Haga clic en Elegir archivo, seleccione su CSR (*ecccertreq.csr*) y haga clic en Contine.

8. Descargue el certificado haciendo clic en Download.

Registrar Dominios

Para poder hacer uso de Apple Pay se deben registrar los dominios desde los que mostraremos su botón.

1. Acceda al perfil del MerchantId generado en el apartado anterior.
2. Revisar el apartado "*Merchant Domains*" y pulsar en "*Add Domain*".



3. Escribir el dominio necesario y pulsar en "*Save*".

4. Pulsar a "*Download*" para descargar el fichero necesario para la verificación del dominio.
5. Alojar el Fichero descargado en la ruta que se especifica (raíz de nuestro dominio)
6. Pulsar en "*Verify*".

7. Una vez verificado el dominio comprobaremos que queda de la siguiente forma:

**Se debe repetir esta operación tantas veces como dominios diferentes tengamos.

3.2.Integración de Apple pay

3.2.1. Mostrar Botón Apple Pay

Para mostrar el botón de Apple Pay incluiremos los siguientes elementos en la página:

```
<script src="https://applepay.cdn-apple.com/jsapi/v1/apple-pay-sdk.js"></script>

<style>

apple-pay-button {

  --apple-pay-button-width: 140px;

  --apple-pay-button-height: 30px;

  --apple-pay-button-border-radius: 5px;

  --apple-pay-button-padding: 5px 0px;

}

</style>


<div id="box">

       <apple-pay-button       buttonstyle="black"       type="buy"       locale="es-ES"
onclick="buttonClicked()></apple-pay-button>

</div>
```

Para más información sobre las propiedades CSS del botón consulte el siguiente enlace:

https://developer.apple.com/documentation/apple_pay_on_the_web/displaying_apple_pay_buttons_using_javascript

Deberemos validar si la API de Apple Pay está disponible y habilitada en el dispositivo desde el que estamos haciendo el pago para mostrar el botón solo en dispositivos Apple.

Esto lo podemos verificar de dos formas:

- canMakePayments. Verifica que el dispositivo tiene ApplePay habilitado, pero no verifica si el usuario ya tiene una tarjeta guardada en su dispositivo.
- canMakePaymentsWithActiveCard. Verifica tanto si es dispositivo tiene ApplePay habilitado como si tiene una tarjeta guardada en su dispositivo.

Con el siguiente código solo se mostrará el botón de Apple Pay si estamos en un dispositivo válido. En la variable merchantIdentifier debemos poner nuestro identificador generado en los pasos anteriores. Para ello crear la función ***buttonClicked*** definida en el evento ***onclick*** del botón ***apple-pay-button:***

```
//Mostrar ApplePay

if (window.ApplePaySession) {

console.log(window.ApplePaySession);

var merchantIdentifier = ' merchant.com.example.mystore';

var promise = ApplePaySession.canMakePaymentsWithActiveCard(merchantIdentifier);

promise.then(function (canMakePayments) {
        document.getElementById("box").style.display = "block";

}, function(rejection){

 document.getElementById("box").style.display = "none";}

);

} else {

document.getElementById("box").style.display = "none";

}

function buttonClicked() {

}
```

3.2.2. Crear ApplePaySession

Lo primero que deberemos hacer al pulsar el botón será construir el objeto ApplePaySession donde se enviará el diccionario ApplePayPaymentRequest con toda la información de la transacción.

A continuación, se muestra un ejemplo de ApplePayPaymentRequest con los campos obligatorios que se deberá añadir a la función ***buttonClicked***:

```
if (!ApplePaySession) {

        return;

}

//ApplePayPaymentRequest

const request = {

        "countryCode": "ES",

        "currencyCode": ”EUR”

        "merchantCapabilities": [

                "supports3DS"

        ],
```

```
        "supportedNetworks": [
                "visa",
                "masterCard",
                "amex",
                "discover"
        ],
        "total": {
                "label": "Nombre Comercio",
                "type": "final",
                "amount": "4.99"
        }
};
```

Para más información sobre campos opcionales (descuentos, opciones de envío, pagos recurrentes, suscripciones...) puede consultar el siguiente enlace:

https://developer.apple.com/documentation/apple_pay_on_the_web/applepaypaymentrequest

Una vez tengamos preparado el diccionario ApplePayPaymentRequest con toda la información crearemos el objeto ApplePaySession y llamaremos el método ***begin*** para iniciar el proceso de validación del comercio. Este código se debe añadir a la función ***buttonClicked.***

```
// Create ApplePaySession
    const session = new ApplePaySession(3, request);
    session.begin();
```

3.2.3. Obtener token Apple Pay

A continuación mostraremos un ejemplo de integración de esta parte completa y explicaremos punto a punto los detalles. Tened en cuenta que este parte de la integración debe desarrollarte tanto en parte web como en parte servidor. Este código se debe añadir a la función ***buttonClicked.***

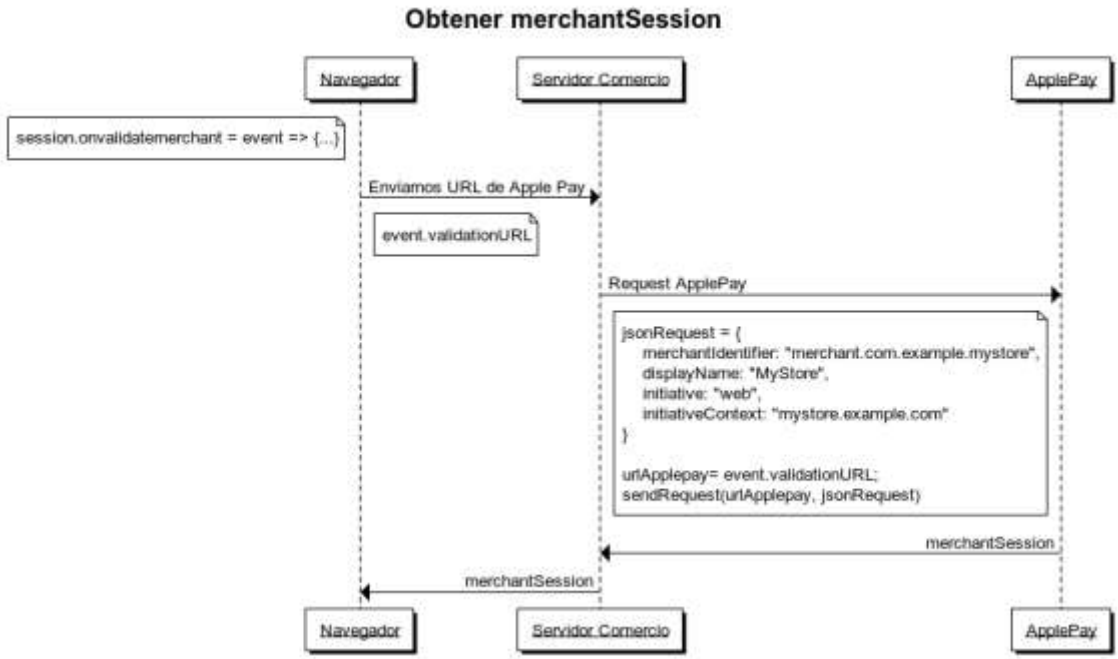
```
session.onvalidatemerchant = event => {
    //1. Llamar a vuestro servidor para generar un merchantSession llamando a Apple.
        //Pseudocódigo de ejemplo
        const merchantSession = serverValidateMerchant(event.validationURL);
    //2. Completamos la validación del comercio
        session.completeMerchantValidation(merchantSession);
    //3. Información de la transacción en la página de pago
    session.onpaymentmethodselected = event => {
            // Define ApplePayPaymentMethodUpdate based on the selected payment
            method.
            // No updates or errors are needed, pass an empty object.
            const update = {
                    newTotal: {
                            "label": "Nombre Comercio",
                                    "type": "final",
                                    "amount":"4.99"
                    }
            };
            session.completePaymentMethodSelection(update);
    };
    //4. Autorización del pago
    session.onpaymentauthorized = event => {
            // Token cifrado de Apple
            var paymentData = event.payment.token.paymentData);
            //Llamar a vuestro servidor para enviar el token de pago a Redsys
            const resultado = await serverAutorizationPayRedsys(paymentData);
            const result = {
                    "status": resultado;
            };
            session.completePayment(result);
    };
};
```

1. Llamar a vuestro servidor para generar un merchantSession llamando a Apple.

Antes de mostrar la página de pago de Apple Pay al cliente deberemos generar una sesión de pago válida (merchantSession). Para ello deberemos comunicarnos con los servidores de Apple Pay, y por razones de seguridad deberemos hacerlo desde nuestro servidor y no desde nuestro cliente web.

Es la única parte de la integración que deberemos hacer desde el lado del servidor.

Diagrama flujo validación del comercio en el servidor:



2. Completamos la validación del comercio

Una vez obtengamos el objeto de sesión recibido por parte de Apple Pay en nuestro navegador deberemos completar el método **onvalidatemerchant()** pasando este nuevo objeto de sesión al método **completeMerchantValidation().** Una vez completado se mostrará la página de pago de Apple Pay.

3. Información de la transacción en la página de pago

Una vez completada la validación del comerciante Apple Pay proporciona información de la transacción en la página de pago (método de pago, dirección de facturación, dirección o método de envío) para así calcular el costo total del pago.

Para gestionar estos ajustes podemos impleentar los siguientes event handler:

- **onpaymentmethodselected**. Método de pago.
- **oncouponcodechanged**. Insertar o actualizar un cupón.
- **onshippingmethodselected**. Método de envío
- **onshippingcontactselected**. Contacto de envío

Es obligatorio que todos estos eventos contengan un objeto **newTotal**. Para más información sobre los diferentes eventos puede consultar los siguientes enlaces:

https://applepaydemo.apple.com/ - **Respond to Payment Sheet Interactions**

https://developer.apple.com/documentation/apple_pay_on_the_web/applepaysession/1778013-onpaymentmethodselected

4. Obtención del token

Finalmente tenemos que implementar el evento **onpaymentauthorized** para obtener el token de la transacción. Una vez el usuario autentique la transacción con Face ID, Touch ID o contraseña desde su dispositivo obtendremos el token encriptado que deberemos enviar posteriormente a Redsys para que procese el pago (Integración Redsys).

Después de recibir la respuesta de Redsys deberemos llamar al método **completePayment** con el resultado de la operación: **STATUS_SUCCESS** o **STATUS_FAILURE** acompañado de un mensaje **ApplePayError**.

Para más información sobre la autorización y la finalización de la operación puede consultar los siguientes enlaces:

https://developer.apple.com/documentation/apple_pay_on_the_web/applepaysession/1778020-onpaymentauthorized

https://applepaydemo.apple.com/ - Authorize the Payment

3.2.4. Autorización del pago (Integración Redsys)

En la operación de autorización enviada al TPV Virtual, se deben añadir los siguientes parámetros adicionales:

DS_XPAYDATA	*10000 / A-N*	**Obligatorio. Campo en el que se incluye el valor del objeto "paymentData" enviado por Apple en formato Hexadecimal**
DS_XPAYTYPE	*10 / A-N*	Obligatorio. Valor fijo "Apple"
DS_XPAYORIGEN	*5 / A-N*	Obligatorio. Origen de la petición. Los posibles valores son: - WEB - InApp

Ejemplo de respuesta de autenticación recibida de Apple. En negrita se indica el objeto que ha de informarse en formato Hexadecimal.

{
"token":{
"paymentData":{
"data":"CZOOPbi/R7UBdW3as7T...0YYWuQ1iZhvtjAfx+A==",
"signature":"MIAGCSqGSIb3DQEHAqCA...3gAtcDwfdZIAAAAAAAA=",
"header":{
"publicKeyHash":"i4BsP3h7AdaM3DU30UA2pucLcPYT1J9bAj3gi8eAOzw=",
"ephemeralPublicKey":"MFkwEwYHKoZIzj0CAQYI...OyHpyVnRod+CpBTMxQ==",
"transactionId":"a9bf8e71ca58173d42af5f24b57ce047528fb285759b296c12f3d8f2e5926644"
},
"version":"EC_v1"
},
"paymentMethod":{
"displayName":"Visa 0492",
"network":"Visa",
"type":"debit"
},
"transactionIdentifier":"A9BF8E71CA58173D42AF5F24B57CE047528FB285759B296C12F3D8F2E5926644"

```
        }
}
```

A continuación, se incluye un ejemplo de petición al TPV Virtual

<DATOSENTRADA>

<DS_MERCHANT_AMOUNT>42</DS_MERCHANT_AMOUNT>

<DS_MERCHANT_ORDER>813734</DS_MERCHANT_ORDER>

<DS_MERCHANT_MERCHANTCODE>999008881</DS_MERCHANT_MERCHANTCODE>

<DS_MERCHANT_CURRENCY>978</DS_MERCHANT_CURRENCY>

<DS_MERCHANT_TRANSACTIONTYPE>0</DS_MERCHANT_TRANSACTIONTYPE>

<DS_MERCHANT_TERMINAL>871</DS_MERCHANT_TERMINAL>

**<DS_XPAYDATA>7B2276657273696F6E223A2245435F7631222C2264617461223A224875
3175315039414370305 94A53316C2B57746B6D6330626857544838557855484259617446
6D365583634515978596662 4D4846757964674D4136414B465556676C693869746C462
B7965495A704532572B686E6E796A56446F74714644472F483850425579526A737847436
F782B4B71726741756A504B3146565578595448304 72F7A427862763556506174316C344
54E7132635A6D6A686251377341504C7A39584E415039756C3463384668697834652F473
848447A6A5139307770616A6D4E57342B4A71444B72516F7A504B4B6B56723661617272
76512B4746594230594976447458744251586A57494E424B774B524B4E524A6B532F6445
507A736770456245684374333336426E4D53493555306C7131466B3873724676412B554B
526B77466572446E617479465A4F764A6B5942624D424C30434531726C6C6842466E667
0595A6F70772B764652725A356749413531484E2F52486E6D4F4C313839767A6B30652F5
0514F37412B2B54754F.....</DS_XPAYDATA>**

<DS_XPAYTYPE>Apple</DS_XPAYTYPE>

<DS_XPAYORIGEN>InApp</DS_XPAYORIGEN>

</DATOSENTRADA>

3.3.Configuración del TPV Virtual

Desde la configuración del TPV Virtual se rellenarán los campos correspondientes al merchantIs del comercio y la clave privada generada por el comercio (formato PKCS8) en el proceso de enrolamiento:

**Esta integración sólo se puede realizar en el entorno de producción del TPV Virtual.

3.4. Funcionalidad avanzada. Envío de datos descifrados en comercio

El comercio se encargará en este caso de desencriptar los datos, obteniendo la información necesaria.

https://developer.apple.com/library/archive/documentation/PassKit/Reference/PaymentTokenJSON/PaymentTokenJSON.html

Key	Valor	Descripción
applicationPrimaryAccountNumber	string	Número de cuenta específico del dispositivo de la tarjeta que financia esta transacción.
applicationExpirationDate	date as a string	Fecha de caducidad de la tarjeta en formato AAMMDD.
onlinePaymentCryptogram	A Base64 encoded string	Criptograma de pago en línea, según lo definido por 3-D Secure.
eciIndicator	string	Indicador ECI, según lo definido por 3-D Secure.

Estos datos se enviarán al TPV Virtual en el parámetro **Ds_XPayDecodedData**, en formato Json y con la siguiente relación:

```
{
    "cryptogram":"XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "eciInd":"XX",
    "expirationDate":"AAMM",
```

```
    "token": "XXXXXXXXXXXXXXXXXXX"
}
```

cryptogram = Referencia a onlinePaymentCryptogram.

eciInd = Referencia al eciIndicator.

expirationDate = Referencia del campo applicationExpirationDate. Se tiene que enviar con formato **AAMM**.

token = Referencia al campo applicationPrimaryAccountNumber.

En la operación de autorización enviada al TPV Virtual, se deben añadir los siguientes parámetros adicionales:

Parámetro	Valor
Ds_XPayDecodedData	Objeto JSON con la información requerida
Ds_XPayType	Apple
Ds_XPayOrigen	InApp - Si la integración es sobre una App móvil WEB - Si la integración es Web

A continuación, se adjunta un ejemplo de petición al TPV Virtual:

```
<DATOSENTRADA>
<DS_MERCHANT_AMOUNT>42</DS_MERCHANT_AMOUNT>
<DS_MERCHANT_ORDER>813734</DS_MERCHANT_ORDER>
<DS_MERCHANT_MERCHANTCODE>999008881</DS_MERCHANT_MERCHANTCODE>
<DS_MERCHANT_CURRENCY>978</DS_MERCHANT_CURRENCY>
<DS_MERCHANT_TRANSACTIONTYPE>0</DS_MERCHANT_TRANSACTIONTYPE>
<DS_MERCHANT_TERMINAL>871</DS_MERCHANT_TERMINAL>
<DS_XPAYDECODEDDATA>{"cryptogram":
"AgAAAAABk4DWZ4C28yUQAAAAAA=","eciInd":
"05","expirationDate":"2612","token":
"489537******3478"}</DS_XPAYDECODEDDATA>
<DS_XPAYTYPE>Apple</DS_XPAYTYPE>
<DS_XPAYORIGEN>InApp</DS_XPAYORIGEN>
</DATOSENTRADA>
```