Lograr la integración del pago móvil vía webview

Guía de implementación

Versión del documento 1.4

Contenido

1. HISTORIAL DEL DOCUMENTO

Versión	Autor	Fecha	Comentario
1.4	Lyra Network	16/10/2019	Renombrar clases en ejemplos de código
1.3	Lyra Network	16/10/2018	Rediseño integral del documento.
1.2	Lyra Network	16/03/2018	Adición del capítulo **¿Cómo puede la aplicación nativa detectar el fin del pago?** Actualización del capítulo **Enviar la solicitud de pago desde el servidor del vendedor**: adición del parámetro *vads_theme_config*
1.1	Lyra Network	04/01/2018	Adición del capítulo sobre la optimización del desempeño
1.0	Lyra Network	10/11/2017	Versión inicial

2. PRESENTACIÓN

PayZen Le ofrece una solución única para la integración del pago móvil a sus aplicaciones.

Nuestra solución abarca las aplicaciones nativas iOS y Android. Se basa en el uso del componente **webview**.

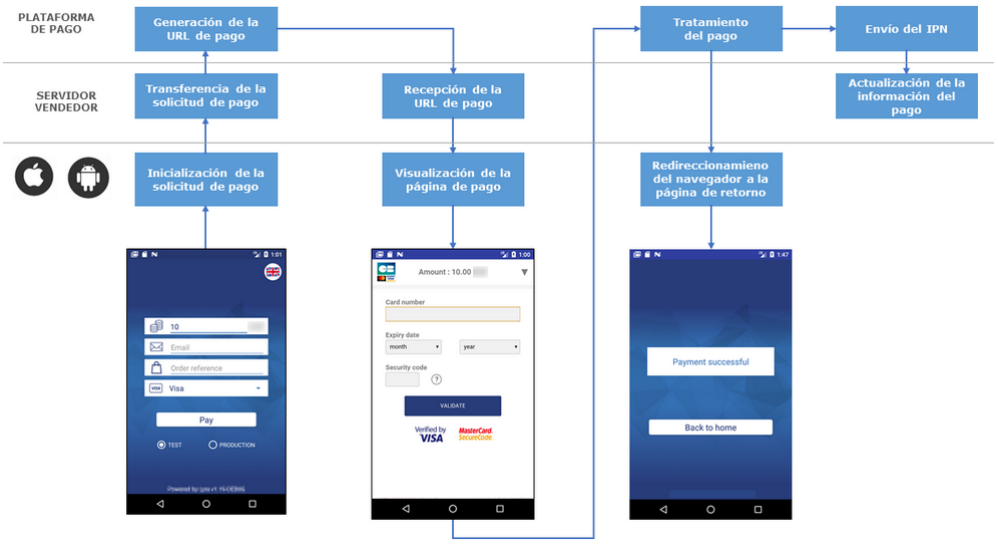
Una webview permite mostrar contenido ya disponible en la web dentro de la aplicación.

De esta forma, la solución PayZen de pago móvil vía webview ofrece varias ventajas al comerciante:

- Una configuración única para el entorno web y móvil.
  Puede volver a utilizar la misma configuración de los pagos de su sitio web.
  Los medios de pago activados, la reglas anti-fraude, etc., se retoman en la aplicación móvil.
- Coherencia en la visualización de la información del proceso del comprador.
  Nuestras páginas de pago ofrecen capacidad de respuesta, por lo cual se pueden adaptar a los diferentes terminales de sus clientes (móvil, tableta o computadora de escritorio).
- Cuenta con un alto nivel de seguridad gracias a nuestro certificado PCI DSS y a la gestión de 3D integrada en el proceso de pago.

PCI DSS (= Payment Card Industry Data Security Standard) es la norma de seguridad de la industria de las tarjetas de pago. Es una norma de seguridad de datos para los principales grupos de tarjetas de pago como Visa, MasterCard, American Express, Discover y JCB.

3. CINEMÁTICA DE PAGO



El comprador valida su canasta.

1. La aplicación móvil realiza una solicitud de pago al servidor del comerciante.
2. El servidor del vendedor envía una solicitud de pago a la plataforma.
3. La plataforma genera una URL de pago y la transmite a su vez a la aplicación móvil.
4. El servidor del vendedor envía la URL de pago a la aplicación móvil.
5. La aplicación móvil abre la página de pago mediante WebView.
6. El comprador ingresa los datos de su tarjeta y hace clic en **Validar**.
7. La plataforma procede al pago y transmite la notificación de pago al servidor del vendedor.
8. El sitio web del vendedor analiza el resultado del pago.
9. El comprador es redirigido automáticamente a la aplicación del vendedor.

4. INTEGRACIÓN DEL PAGO

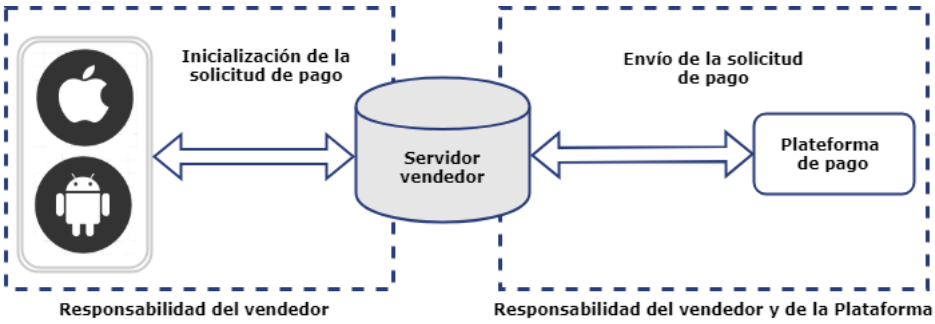
Algunos ejemplos de códigos están disponibles para facilitar la integración:

Servidor Comerciante *https://github.com/lyra/webview-payment-sparkjava-integration-sample*
iOS *https://github.com/lyra/webview-payment-ios-integration-sample*
Android *https://github.com/lyra/webview-payment-android-integration-sample*

IMPORTANTE
Asegúrese de haber leído los comentarios en los archivos Léame antes de iniciar la aplicación. Los archivos `MainActivity.kt` y `app-configuration.properties` deben modificarse según las instrucciones descritas en los comentarios.

La integración se divide en dos etapas:

- integración de los intercambios entre el servidor del vendedor y la plataforma de pago
- integración de los intercambios entre la aplicación móvil y el servidor del vendedor



5. FASE 1: EL SERVIDOR DEL VENDEDOR

5.1. Transferencia de la solicitud de pago

El servidor del vendedor recibe una solicitud de pago de la aplicación móvil y debe transmitirla a la plataforma de pago.

Para ello, el sitio del vendedor genera un formulario de pago HTML que luego enviará a la plataforma de pago.

La integridad de los datos intercambiados está protegida por un intercambio de firmas alfanuméricas entre la plataforma de pago y el sitio web del vendedor.

El servidor del vendedor transmite la firma alfanumérica en el formulario de pago (consultar capítulo **Calcular la firma** de la **Guía de implementación de la API de formularios** disponible en nuestro sitio de documentación).

IMPORTANTE
Todos los datos del formulario deben estar codificados en UTF-8.
De esta forma, los caracteres especiales (acentos, puntuación, etc.) serán interpretados correctamente por la plataforma de pago.
En el caso contrario, el cálculo de la firma será erróneo y el formulario será rechazado.

1. Creación del formulario de pago

Para crear el formulario de pago:

1. Utilice todos los campos presentes en el cuadro para crear la solicitud de pago.

Nombre del campo	Descripción	Formato	Valor
vads_site_id	Identificación de la tienda	n8	Ejemplo: 12345678
vads_currency	Código numérico de la moneda que se utilizará para el pago, según la norma ISO 4217 (código numérico)	n3	Ejemplo: 840 para el dólar norteamericano (USD)
vads_amount	Monto del pago en su unidad monetaria más pequeña (el centavo para el para el dólar estadunidense)	n..12	Ejemplo: 3000 para 30,00 USD
vads_cust_email	Dirección de correo electrónico del comprador	ans..150	Ejemplo: abc@example.com
vads_payment_cards	Tipo de tarjeta.	String	Ejemplo: **VISA** (Consulte la Guía de implementación de la API de formularios para la lista de valores posibles).

Nombre del campo	Descripción	Formato	Valor
vads_order_id	Número del pedido	ans..64	Ejemplo: 2-XQ001
vads_version	Versión del protocolo de intercambio con la plataforma de pago	enum	**V2**
vads_theme_config	Permite aumentar el desempeño al desactivar elementos de la página de pago como el selector de idioma, los logotipos de la parte inferior de la página, etc.	map	**SIMPLIFIED_DISPLAY=true**
vads_trans_date	Fecha y hora del formulario de pago en el huso horario UTC	n14	Respete el formato AAAAMMDDhhmmss Ejemplo: 20170701130025
vads_trans_id	Número de la transacción	n6	Ejemplo: 123456
vads_payment_config	Tipo de pago	enum	**SINGLE**para un pago único
vads_page_action	Acción a realizar	enum	**PAYMENT**
vads_ctx_mode	Adquisición de los datos en la plataforma de pago	enum	**TEST** o**PRODUCTION**
vads_action_mode	Modo de adquisición de los datos de la tarjeta	enum	**INTERACTIVE**
signature	Firma que garantiza la integridad de las solicitudes intercambiadas entre el sitio web vendedor y la plataforma de pago.	ans44	Ejemplo: NrHSHyBBBc +TtcauudspNHQ5cYcy4tS4IjvdC0ztFe8=

2. Utilice los campos a continuación para gestionar el regreso a la aplicación móvil al final del pago.

   Un pago puede terminar en 4 estados diferentes:

   - Pago aceptado
   - Pago denegado
   - Pago en error
   - Pago abandonado por el comprador

   Debe asociar una URL a cada estado:

Nombre del campo	Descripción	Formato	Valor
vads_url_success	URL donde se redirigirá al comprador **si el pago es exitoso**.	ans..1024	Ejemplo: **http://webview.success**
vads_url_refused	URL donde se redirigirá al comprador **si el pago es denegado**.	ans..1024	Ejemplo: **http://webview.refused**
vads_url_cancel	URL donde se redirigirá al comprador **en caso de abandono** o expiración (timeout).	ans..1024	Ejemplo: **http://webview.cancel**
vads_url_error	URL donde se redirigirá al comprador **en caso de error**.	ans..1024	Ejemplo: **http://webview.error**

3. Utilice los campos a continuación para configurar los plazos de redirección a la aplicación móvil al final del pago.

Nombre del campo	Descripción	Formato
vads_redirect_success_timeout	Define el plazo de espera antes del redirección, luego de un pago exitoso. Este plazo se expresa en segundos y debe estar entre 0 y 300 segundos. **Asigne a este campo el valor "0" para no mostrar el recibo del pago y redirigir automáticamente al comprador a la aplicación móvil.**	n..3
vads_redirect_error_timeout	Define el plazo de espera antes del redirección, luego de un pago denegado.	n..3

Nombre del campo	Descripción	Formato
	Este plazo se expresa en segundos y debe estar entre 0 y 300 segundos. **Asigne a este campo el valor "0" para no mostrar la página de rechazo del pago y redirigir automáticamente al comprador a la aplicación móvil.**	

Tabla 1: Lista de campos opcionales disponibles.

4. Agregue los demás campos opcionales en función de sus necesidades (consultar capítulo **Utilizar funciones complementarias** de la **Guía de implementación de la API de formularios** disponible en nuestro sitio de documentación).

5. Consulte el capítulo **Calcular la firma** de la **Guía de implementación de la API de formularios** y calcule el valor del campo**signature**.

2. Envío de la solicitud de pago

La API de creación del pago está disponible en modo POST en la siguiente dirección:

https://secure.payzen.lat/vads-payment/entry.silentInit.a

IMPORTANTE

La URL de la API de creación de pago es distinta de la URL de la página de pago, tal como se describe en la Guía de implementación de la API de formularios.

Extracto del código de ejemplo:

```
List<NameValuePair> formParameters = new ArrayList<>();

formParameters.add(new BasicNameValuePair("vads_action_mode", "INTERACTIVE"));
formParameters.add(new BasicNameValuePair("vads_amount", amount));
formParameters.add(new BasicNameValuePair("vads_ctx_mode", mode));
formParameters.add(new BasicNameValuePair("vads_currency", currency));
if (StringUtils.isNotEmpty(email)) {
formParameters.add(new BasicNameValuePair("vads_cust_email", email));
}

formParameters.add(new BasicNameValuePair("vads_language", language));
if (StringUtils.isNotEmpty(orderId)) {
 formParameters.add(new BasicNameValuePair("vads_order_id", orderId));
}
formParameters.add(new BasicNameValuePair("vads_page_action", "PAYMENT"));

//Set the card type if provided
if (StringUtils.isNotEmpty(cardType)) {
 formParameters.add(new BasicNameValuePair("vads_payment_cards", cardType.toUpperCase()));
}
formParameters.add(new BasicNameValuePair("vads_payment_config", "SINGLE"));
formParameters.add(new BasicNameValuePair("vads_site_id", merchantSiteId));
formParameters.add(new BasicNameValuePair("vads_theme_config", "SIMPLIFIED_DISPLAY=true"));
formParameters.add(new BasicNameValuePair("vads_trans_date",
 calculateDateFormatInUTC("yyyyMMddHHmmss")));
formParameters.add(new BasicNameValuePair("vads_trans_id", String.format("%06d",
 transactionId)));
formParameters.add(new BasicNameValuePair("vads_url_cancel", "http://webview_" +
 merchantSiteId + ".cancel"));
formParameters.add(new BasicNameValuePair("vads_url_error", "http://webview_" +
 merchantSiteId + ".error"));
formParameters.add(new BasicNameValuePair("vads_url_refused", "http://webview_" +
 merchantSiteId + ".refused"));
formParameters.add(new BasicNameValuePair("vads_url_return", "http://webview_" +
 merchantSiteId + ".return"));
formParameters.add(new BasicNameValuePair("vads_url_success", "http://webview_" +
 merchantSiteId + ".success"));
formParameters.add(new BasicNameValuePair("vads_version", "V2"));

//Create the string to sign
String concatenateMapParams = "";
for (NameValuePair pair : formParameters) {
 concatenateMapParams += pair.getValue() + "+";
}
```

```
//Add private key in signature
concatenateMapParams += usedMerchantKey;

//Add signature to form parameters
formParameters.add(new BasicNameValuePair("signature", hmacSha256(concatenateMapParams,
 usedMerchantKey)));
```


5.2. Recepción de la URL de notificación

La plataforma de pago devuelve una respuesta en formato JSON que contiene un código de estado HTTP de éxito o error.

Éxito

En caso de éxito, la plataforma de pago emite un código de estado HTTP `200 OK`.

La respuesta contiene la URL de pago hacia la cual la aplicación debe redirigir al comprador.

```
{
 "status":"INITIALIZED",
 "redirect_url":"https://secure.payzen.lat:443/vads-payment/
exec.refresh.a;jsessionid=CE2Cb9daEDe7f6dBF31FE65e.vadpayment01bdx"
}
```

Error

En caso de error, la plataforma de pago emite un código de estado HTTP `400 Bad Request` O `500 Internal Server Error`.

La respuesta contiene los detalles del error.

```
{
 "status":"ERROR",
 "error":" {"code": "09", "value": "Missing or invalid parameter value"}"
}
```

Para más información, consulte la lista de los códigos de error de la API de formularios:

https://payzen.io/lat/error-code/error-00.html

Extracto del código de ejemplo:

```
//If the HTTP return code is 200 (OK) we prepare the generated URL
if (httpResponseCode == 200) {
 if ("INITIALIZED".equals(responseData.get("status"))) {
  redirectionUrl = responseData.get("redirect_url");
 } else {
  //Payment could  not be created. Maybe a missing parameter, an invalid value or signature?
  //Use logs here in order to detect and fix the real cause
  throw new RuntimeException("Error in payment initialization. Returned error: " +
 responseData.get("error"));
 }
} else {
 throw new RuntimeException("Error in payment initialization. HTTP errorCode: " +
 httpResponse);
}
```

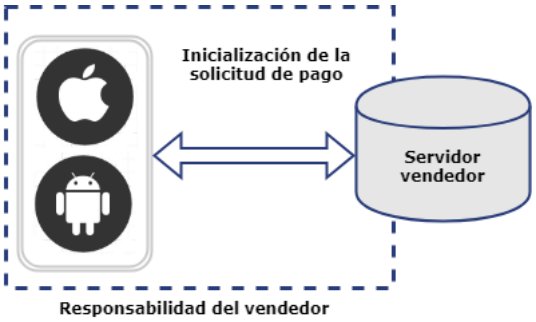
5.3. Procesamiento de la notificación de fin de pago (IPN)

Una vez realizado el pago, la plataforma de pago informa el resultado de la transacción al servidor del vendedor.

Los datos se envían a la URL de notificación definida en el .

Consulte la **Guía de implementación de la API de formularios**, disponible en nuestro sitio de documentación, para configurar las reglas de notificación y el análisis de los datos transmitidos.

6. FASE 2: LA APLICACIÓN MÓVIL



6.1. Inicialización de la solicitud de pago

Cuando el comprador valida su pedido, la aplicación genera una “payload” que contiene los datos del carrito, los datos de contacto del comprador, la información sobre la entrega, etc.

El ejemplo proporcionado utiliza los siguientes datos:

```
{
 "email": "example@email.com",
 "orderId": "myOrderId-1",
 "amount": "200",
 "currency": "840",
 "mode": "TEST",
 "language": "es",
 "cardType": ""
}
```

La aplicación móvil transmite la solicitud de pago al servidor del vendedor mediante una solicitud POST.

Extracto del código de ejemplo para Android:

```
val conn = URL(serverUrl).openConnection() as HttpURLConnection
conn.requestMethod = "POST"
conn.setRequestProperty("Content-type", "application/json")
conn.setRequestProperty("Accept", "*/*")
conn.doInput = true
conn.doOutput = true
conn.connectTimeout = 15000

val os = conn.outputStream
val writer = BufferedWriter(OutputStreamWriter(os, "UTF-8"))
writer.write(payload.toString())
writer.flush()
writer.close()
os.close()

conn.connect()

val out = OutputStreamWriter(conn.outputStream)
```


Extracto del código de ejemplo para iOS:

```
/// Build an URLRequest according required payment information : server url, email, amount,
 mode, lang
///
/// - Returns: URLRequest object
func buildRequest() -> URLRequest? {
 let serverUrl: NSURL = NSURL(string: PaymentProvider.SERVER_URL)!
 var urlRequest = URLRequest(url:serverUrl as URL)
 urlRequest.httpMethod = "POST"
 var params: [String: String] = ["amount": paymentInfo.amount, "currency":
 paymentInfo.currency, "mode": paymentInfo.mode, "language": paymentInfo.lang]
 if !paymentInfo.email.isEmpty{
  params["email"] = paymentInfo.email
 }
 if !paymentInfo.cardType.isEmpty{
  params["cardType"] = paymentInfo.cardType
 }
 do{
  let jsonParam = try JSONSerialization.data(withJSONObject: params, options: [])
  urlRequest.httpBody = jsonParam
 }
 catch{
  return nil
 }

 return urlRequest
}
```

6.2. Visualización de la página de pago en WebView

Una vez procesada la solicitud, la plataforma de pago transmite la URL de pago a la aplicación móvil.

La aplicación inicia WebView y muestra la página de pago.

Extracto del ejemplo de código para Android (**Kotlin**)

```
val webView = WebView(this)

// Url loading
webView.loadUrl(url)

// Enable javascript
webView.settings.javaScriptEnabled = true

// To allow debug WebView from Chrome Dev Tools
WebView.setWebContentsDebuggingEnabled(false)

// Define new web view client by overriding shouldOverrideUrlLoading method in order to check
 urls
webView.webViewClient = object: WebViewClient() {
 override fun onPageFinished(view: WebView, url: String) {
  progressBar.visibility = View.GONE
  super.onPageFinished(view, url)
 }

 @Suppress("OverridingDeprecatedMember")
 override fun shouldOverrideUrlLoading(view: WebView, url: String): Boolean {
  return checkUrl(webView, url)
 }

 @TargetApi(Build.VERSION_CODES.LOLLIPOP)
 override fun shouldOverrideUrlLoading(view: WebView, webResourceRequest: WebResourceRequest):
 Boolean {
  return checkUrl(webView, webResourceRequest.url.toString())
 }
}
webView.canGoForward()
```


Extracto del ejemplo de código para iOS (**Swift**)

```
/// Call server to get payment url, supply a block completion (callback)
///
/// - Returns: status boolean, payment url
func getPaymentContext(completion: @escaping (Bool, String, NSError?) -> ()){
 // Build request
 let urlRequest = buildRequest()
 // Call server to obtain a payment Url
 // Completion is a callback, giving call status, and payment url if success
 if let request = urlRequest{
  let task = URLSession.shared.dataTask(with: request) { (data: Data?, response: URLResponse?,
 error: Error?) in
    if error != nil{
     completion(false, "", NSError.init(domain:PaymentProvider.ERROR_DOMAIN, code:
 PaymentProvider.ERROR_NO_CONNECTION.errorCode, userInfo: [NSLocalizedFailureReasonErrorKey:
 PaymentProvider.ERROR_NO_CONNECTION.errorMsg]))
    }
    if let httpResponse = response as? HTTPURLResponse {
     let json = try? JSONSerialization.jsonObject(with: data!, options: .mutableContainers) as?
 NSDictionary
     var redirectionUrl = ""
     var errorMsg = ""
     if let jsonResponse = json {
      redirectionUrl = (jsonResponse!["redirectionUrl"] as? String)!
      errorMsg = (jsonResponse!["errorMessage"] as? String)!
     }
     switch(httpResponse.statusCode){
      case 200:
       completion(true, redirectionUrl, nil)
      case 400, 500:
       completion(false, "", NSError.init(domain:PaymentProvider.ERROR_DOMAIN, code:
 PaymentProvider.ERROR_SERVER.errorCode, userInfo: [NSLocalizedFailureReasonErrorKey:
 PaymentProvider.ERROR_SERVER.errorMsg + errorMsg]) )
      default:
       completion(false, "", NSError.init(domain:PaymentProvider.ERROR_DOMAIN, code:
 PaymentProvider.ERROR_UNKNOW.errorCode, userInfo: [NSLocalizedFailureReasonErrorKey:
 PaymentProvider.ERROR_UNKNOW.errorMsg]))
     }
    }
    else{
     completion(false, "", NSError.init(domain:PaymentProvider.ERROR_DOMAIN, code:
 PaymentProvider.ERROR_TIMEOUT.errorCode, userInfo: [NSLocalizedFailureReasonErrorKey:
 PaymentProvider.ERROR_TIMEOUT.errorMsg]))
    }
 }
 task.resume()
 } else{
  completion(false, "", NSError.init(domain:PaymentProvider.ERROR_DOMAIN, code:
 PaymentProvider.ERROR_UNKNOW.errorCode, userInfo: [NSLocalizedFailureReasonErrorKey:
 PaymentProvider.ERROR_UNKNOW.errorMsg]))
 }
}
```

6.3. Detección del fin del pago

Para detectar el fin del pago, es necesario analizar las distintas URL que pasan por WebView.

En función de la URL, la aplicación móvil podrá:

- Aceptar la propagación de la URL y mostrar la página en WebView
  (por ejemplo, durante una solicitud de pago, para permitir que el comprador realice su pago)
- Rechazar la propagación de la URL y retomar el control.
  (por ejemplo, al final de un pago exitoso, para permitir que el comprador regrese a la aplicación nativa)

Gracias a un mecanismo de vigilancia de las URL, usted mantiene el control sobre la cinemática de pago y puede decidir en qué momento se vuelve a su aplicación nativa.

Extracto del ejemplo de código para Android (Kotlin)

```
private fun checkUrl(view: WebView, url: String): Boolean {
 val isCallBack = isCallbackUrl(url)

 when {
  // payment is finish
  isCallBack -> {
   view.stopLoading()
   goToFinalActivity(url)
  }
  else -> view.loadUrl(url)
 }
 return (!isCallBack)
}
```

Extracto del ejemplo de código para iOS (**Swift**):

```
func notifyPaymentFinish(navigationAction: WKNavigationAction){
 let webViewUrlResponse = self.buildWebviewUrlResponse(navigationAction: navigationAction)
 var error: NSError?
 switch webViewUrlResponse.paymentStatus {
  case "success":
   error = nil
  case "cancel":
   error = NSError.init(domain: PaymentProvider.ERROR_DOMAIN, code:
 PaymentProvider.ERROR_PAYMENT_CANCELATION.errorCode, userInfo:
 [NSLocalizedFailureReasonErrorKey: PaymentProvider.ERROR_PAYMENT_CANCELATION.errorMsg])
  case "refused":
   error = NSError.init(domain:PaymentProvider.ERROR_DOMAIN, code:
 PaymentProvider.ERROR_PAYMENT_REFUSED.errorCode, userInfo: [NSLocalizedFailureReasonErrorKey:
 PaymentProvider.ERROR_PAYMENT_REFUSED.errorMsg])
  default:
   error = NSError.init(domain:PaymentProvider.ERROR_DOMAIN, code:
 PaymentProvider.ERROR_UNKNOW.errorCode, userInfo: [NSLocalizedFailureReasonErrorKey:
 PaymentProvider.ERROR_UNKNOW.errorMsg])

 }
 self.dismiss(animated: true) {
  self.paymentDelegate?.didPaymentProcessFinish(error: error)
 }
}
```