Intégrer l'API Google Pay pour vos paiements mobiles

Version du document 1.3

Sommaire

1. HISTORIQUE DU DOCUMENT

Version	Auteur	Date	Commentaire
1.3	Lyra Network	25/04/2019	Apposition de la mention Trade Mark.
1.2	Lyra Network	14/01/2019	• Mise à jour des prérequis • Ajout de la procédure de création d'un contrat
1.1	Lyra Network	19/10/2018	• Mise à jour vers la dernière version de l'API Google Pay™. • Mise à jour des exemples de codes
1.0	Lyra Network	10/07/2018	Version initiale.

2. CONTACTER L'ASSISTANCE TECHNIQUE

Vous cherchez de l'aide? Consultez notre FAQ sur notre site

https://payzen.io/fr-FR/faq/sitemap.html

Pour toute question technique ou demande d'assistance, nos services sont disponibles du lundi au vendredi, de 9h à 18h

par téléphone au :	0811708709 Service 0,06 € / mi + prix appel
par e-mail :	*support@payzen.eu*
via votre Back Office Marchand :	menu **Aide** > **Contacter le support**

Pour faciliter le traitement de vos demandes, il vous sera demandé de communiquer votre identifiant de boutique (numéro à 8 chiffres) .

Cette information est disponible dans l'e-mail d'inscription de votre boutique ou dans le Back Office Marchand (menu **Paramétrage** > **Boutique** > **Configuration**).

3. PRÉSENTATION

Google Pay™ est un moyen de paiement qui permet à l'acheteur de régler ses achats facilement en utilisant la carte de paiement enregistrée dans son compte Google.
C'est un moyen simple, rapide et sécurisé de réaliser des achats en ligne ou depuis une application mobile.

Il existe deux types d'intégration:

- intégration au sein d'une application mobile
- intégration sur un site marchand (le paiement se fait depuis le navigateur de l'acheteur)

Le présent document décrit l'intégration au sein d'un application mobile en utilisant la payload retournée par l'API Google Pay™. Il est néanmoins possible d'adapter les instructions fournies ci-après en utilisant une payload issue du client pour l'API JavaScript Google Pay™.

Checkout rapide Google Pay™:

Lors d'un paiement, l'acheteur doit renseigner son adresse de facturation de livraison dans l'application mobile.

Afin de faciliter les paiements de l'acheteur sur différents sites ou applications, il est possible de gérer les adresses de facturation et de livraison depuis le wallet Google.

L'acheteur sélectionne son moyen de paiement et ses adresses une seule fois puis le site marchand exploite les données contenues dans la réponse (payload).

(Les informations sont transmises automatiquement à la plateforme de paiement et visibles dans le Back Office PayZen).

Points importants:

- L'implémentation actuelle ne permet pas de réaliser une authentification 3D Secure lors d'un paiement Google Pay™. Il n'y a donc pas de transfert de responsabilité (Liability shift).
- L'implémentation actuelle ne permet pas la création de paiements récurrents (paiements par abonnement) avec le moyen de paiement Google Pay™.

4. PRÉREQUIS

Coté marchand:

- Souscrire à une offre PayZen Premium ou Expert

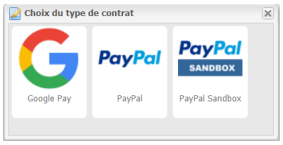
Coté acheteur:

- Disposer d'un compte Google
- Avoir associé un moyen de paiement à son compte Google
- Etre équipé d'un équipement mobile compatible avec les services Google Pay

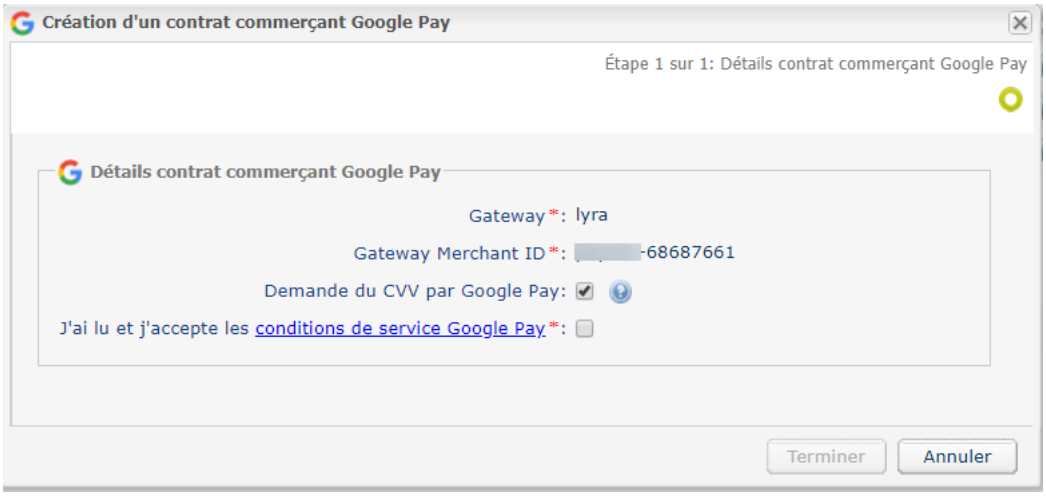
5. DÉCLARER UN CONTRAT GOOGLE PAY™ DANS LE BACK OFFICE MARCHAND

Depuis le menu **Paramétrage** > **Société** > **Contrats**:

1. Cliquez sur **Créer un contrat**.
   La boite de dialogue "Choix du type de contrat" s'affiche.



2. Sélectionnez **Google Pay**.
   L'assistant de création d'un contrat s'ouvre.



   Les champs **Gateway** et **Gateway Merchant ID** sont renseignés. Ces deux informations sont utiles pour l'intégration de Google Pay™ au sein d'une application mobile.

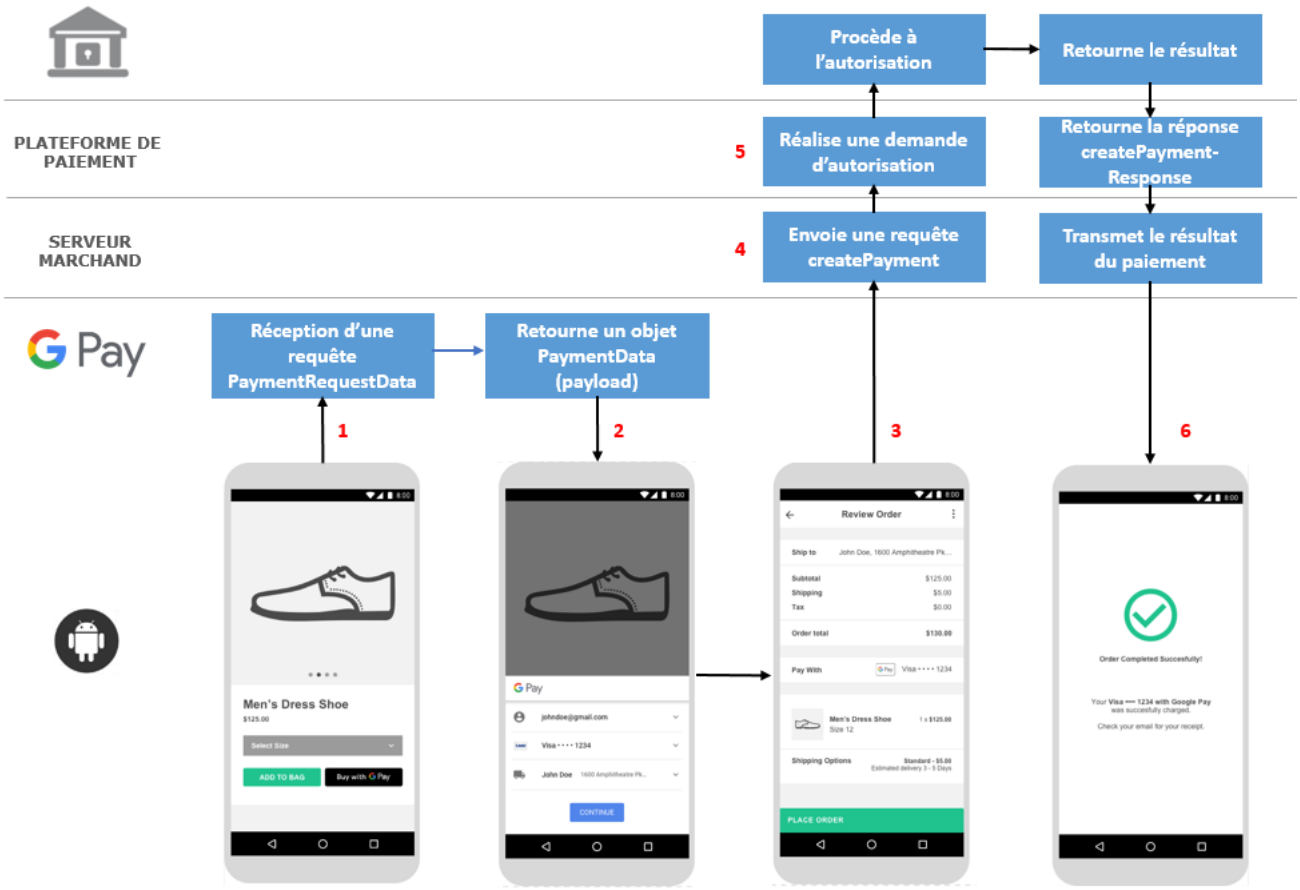
   Contactez PayZen si vous souhaitez faire la saisie du CVV dans l'intégration mobile.

3. Lisez et acceptez les CGS de Google Pay™ en cochant la case prévue à cet effet.
   Cette action est obligatoire pour terminer la création du contrat.

4. Cliquez sur **Terminer**.

Une fois le contrat créé, cliquez sur le bouton "**Associer à une boutique**" et associez-le à la boutique de votre choix.

6. CINÉMATIQUE DES ÉCHANGES



1. L'application mobile soumet une requête `paymentDataRequest` à l'API Google Pay™ pour collecter les données de la carte de l'acheteur.
2. L'API Google Pay™ renvoie un objet `PaymentData` contenant les données chiffrées (appelé aussi "payload").
3. L'application mobile transmet la payload au serveur marchand.
4. Le serveur marchand construit et soumet une requête `createPayment` en utilisant la payload pour valoriser la propriété **walletPayload** de l'objet `cardRequest`. La propriété `scheme` est valorisée à **GOOGLEPAY**.

   Les autres propriétés de l'objet cardRequest ne doivent pas être valorisées.
5. La plateforme de paiement analyse la requête, déchiffre les données de carte et procède à la demande d'autorisation. Elle transmet ensuite le résultat du paiement au serveur marchand.
6. Le serveur marchand reçoit une réponse `createPaymentResponse` et analyse le résultat du paiement. Il transmet l'information à l'application qui se charge de la traiter.

7. GÉRER LES DÉLAIS D'ATTENTE (TIMEOUT)

Le traitement d'une requête web service s'articule autour d'un enchaînement d'évènements asynchrones comme :

- l'envoi de la requête via le réseau du site marchand,
- le transport des informations sur le réseau internet,
- le traitement du paiement par la plateforme,
- l'interrogation des serveurs bancaires, etc

Un incident peut survenir à chaque étape et augmenter le temps du traitement (et donc implicitement le temps d'attente pour l'acheteur).

La réponse à une requête peut être retardée pour de multiples raisons :

- Un temps de réponse long de la part de l'émetteur du porteur de la carte (cas des cartes étrangères, cas de période de forte charge comme les soldes, ...).
- Un temps de réponse long de la part de l'acquéreur lors de la transmission et de la réception de la demande d'autorisation.
- Un temps de réponse long du côté de votre application suite à une charge importante.
- Un temps de réponse long de la plateforme de paiement.
- Un problème de peering sur Internet pouvant entraîner des pertes de messages, etc…

Selon les délais d'attente paramétrés, vous pouvez ne pas recevoir de réponse alors que le traitement asynchrone continue à s'exécuter côté plateforme de paiement.

Un temps de traitement long ne doit pas être considéré comme un paiement refusé.

Pour cette raison, vous devez configurer votre code pour gérer les problèmes potentiels pouvant survenir avec la connexion à l'API SOAP.

Conseils

Le temps moyen de traitement d'une demande de paiement par la plateforme est inférieur à 5 secondes.

Vous devez définir un délai d'attente de 20 à 30 secondes entre:

- le serveur marchand et la plateforme de paiement,
- l'application mobile et le serveur marchand.

Pendant le temps d'attente, un message doit clairement indiquer à l'acheteur que son paiement est en cours.

Une fois le délai d'attente atteint, vous devez indiquer à l'acheteur que son paiement est refusé.

Afin de gérer correctement ces cas de timeout nous préconisons 2 solutions:

- **Créez vos transactions en validation manuelle**

  Le serveur marchand doit envoyer une requête pour valider chaque transaction réussie.

  Les transactions en timeout ne seront pas validées et expireront (au bout de 7 jours pour un paiement réalisé sur le réseau CB).

  Une fois expirées, elles ne pourront plus être remises en banque.

  Aucun acheteur ne sera débité à tort.

- **Créez vos transactions en validation automatique**

  Le serveur marchand doit interroger la plateforme de paiement pour annuler ou rembourser l'acheteur pour chaque requête en timeout.

  Pour cela le serveur marchand devra s'assurer de transmettre un identifiant de commande unique dans les requêtes de paiement.

  Puis, pour chaque transaction en timeout, le serveur marchand doit

  - interroger la plateforme de paiement pour récupérer l'UUID et le statut de la transaction correspondante (méthode `findPayments`)

    Si l'UUID n'existe pas, cela signifie qu'une erreur technique a empêché la création de la transaction.

    Si la transaction a un statut REFUSED, aucune action n'est nécessaire.

  - annuler ou rembourser la transaction (en utilisant les méthodes `cancelPayment` ou `refundPayment`) si la demande d'autorisation a été acceptée durant le timeout.

8. CONSULTER LES CODES D'EXEMPLES

Pour vous accompagner davantage selon votre application, nous avons mis à votre disposition des exemples complémentaires d'implémentation. Veuillez consulter ces liens ci-dessous :

Serveur marchand

- *https://github.com/lyra/googlepay-payment-sparkjava-integration-sample*

Android

- *https://github.com/lyra/googlepay-payment-android-integration-sample*

9. CONFIGURER VOTRE PROJET

Si vous souhaitez plus d'informations sur l'API de paiement Google Pay™, consultez la documentation officielle: *https://developers.google.com/pay/api/android/*

Suivez les recommandations de Google pour les aspects graphiques (logos et textes):

https://developers.google.com/pay/api/android/guides/brand-guidelines

Si vous rencontrez des problèmes sur les appels à l'API Google Pay™, consultez la page suivante:

https://developers.google.com/pay/api/android/support/troubleshooting

9.1. Activer l'API Google Pay™

Pour activer l'API Google Pay™ et l'utiliser dans votre projet, vous devez ajoute ce qui suit à la balise **<application>** du fichier AndroidManifest.xml:

```
<application>
 ...
 <!-- Enables the Google Pay API -->
 <meta-data
  android:name="com.google.android.gms.wallet.api.enabled"
  android:value="true" />
</application>
```

9.2. Ajouter les dépendances

Ajoutez la dépendance Google Play Services 16.0.x (version minimum requise) à votre fichier build.gradle:

```
dependencies {
 compile 'com.google.android.gms:play-services-wallet:16.0.0'
 compile 'com.android.support:appcompat-v7:24.1.1'
}
```


10. PROCÉDER AU PAIEMENT DEPUIS L'APPLICATION MOBILE

Les étapes suivantes montrent comment intégrer l'API Google Pay™ dans une application mobile.

10.1. Définir la version Google Pay™ utilisée

Créez un objet **BaseRequest** contenant les propriétés ci-dessous qui seront utilisées dans toutes vos requêtes:

Extrait de l'exemple de code:

```
private fun getBaseRequest(): JSONObject {
 return JSONObject()
  .put("apiVersion", 2)
  .put("apiVersionMinor", 0)
}
```

10.2. Définir la passerelle de paiement

Implémentez la méthode **getTokenizationSpecification** comme ci-dessous pour spécifier que vous utilisez la plateforme de paiement "lyra".

Extrait de l'exemple de code:

```
private const val GATEWAY_TOKENIZATION_NAME = "lyra"
private const val GATEWAY_MERCHANT_ID = "<REPLACE_ME>"

private fun getTokenizationSpecification(gatewayMerchantId: String): JSONObject {
 val params = JSONObject().put("gateway", GATEWAY_TOKENIZATION_NAME)
 params.put("gatewayMerchantId", gatewayMerchantId)

 return JSONObject().put("type", "PAYMENT_GATEWAY").put("parameters", params)
}
```

*Remplacez <REPLACE_ME> par le numéro de contrat Google Pay™ visible dans (Menu **Paramétrage** > **Société** > **Contrats**).*
Si vous utilisez le code d'exemple, renseignez votre numéro de contrat Google Pay™ dans le fichier MainActivity.kt (variable `GATEWAY_MERCHANT_ID`).

10.3. Définir les méthodes de paiements supportées

Implémentez la méthode **getAllowedCardNetworks** comme ci-dessous pour spécifier les types de cartes autorisés pour le paiement.

Extrait de l'exemple de code:

```
private const val SUPPORTED_NETWORKS = "AMEX, VISA, MASTERCARD, DISCOVER, JCB"
this.supportedNetworks = supportedNetworks.split(Regex(",[ ]*")).toTypedArray()

private fun getAllowedCardNetworks(): JSONArray {
    val allowedCardNetworks = JSONArray()
    for (network in supportedNetworks) {
        allowedCardNetworks.put(network)
    }
    return allowedCardNetworks
}
```


Implémentez la méthode `getAllowedCardAuthMethods` et assignez lui la valeur **PAN_ONLY**.

Extrait de l'exemple de code:

```
private val SUPPORTED_METHODS = Arrays.asList("PAN_ONLY")!!

private fun getAllowedCardAuthMethods(): JSONArray {
    val allowedCardAuthMethods = JSONArray()
    for (method in SUPPORTED_METHODS) {
        allowedCardAuthMethods.put(method)
    }
    return allowedCardAuthMethods
}
```

Pour plus de détails, consultez la documentation de l'objet CardParameters.

10.4. Transmettre les méthodes de paiement supportées

Implémentez la fonction **`getCardPaymentMethod`**.

Créez un objet `cardPaymentMethod` pour transmettre les méthodes de paiement supportées pour le mode de paiement `CARD` ainsi que les informations de la passerelle de paiement.

Extrait de l'exemple de code:

```
private fun getCardPaymentMethod(additionalParams: JSONObject?, tokenizationSpecification:
 JSONObject?): JSONObject {
    val params = JSONObject()
            .put("allowedAuthMethods", getAllowedCardAuthMethods())
            .put("allowedCardNetworks", getAllowedCardNetworks())

    // Additional parameters provided?
   if (additionalParams != null && additionalParams.length() > 0) {
        val keys = additionalParams.keys()
        while (keys.hasNext()) {
            val key = keys.next()
            params.put(key, additionalParams.get(key))
        }
    }
    val cardPaymentMethod = JSONObject()
    cardPaymentMethod.put("type", "CARD")
    cardPaymentMethod.put(
            "parameters", params)
    if (tokenizationSpecification != null) {
        cardPaymentMethod.put(
                "tokenizationSpecification", tokenizationSpecification)
    }
    return
}
```

Pour plus de détails, consultez la documentation de l'objet CardParameters.

10.5. Créer une instance PaymentsClient

Créez une instance de `PaymentsClient` dans la méthode `onCreate` de votre `Activity` pour mettre en place l'interaction avec l'API `PaymentsClient`.

```
private fun initPaymentsClient(activity: Activity, mode: String): PaymentsClient {
 val builder = Wallet.WalletOptions.Builder()
 if (mode == "TEST") {
  builder.setEnvironment(WalletConstants.ENVIRONMENT_TEST)
 } else {
  builder.setEnvironment(WalletConstants.ENVIRONMENT_PRODUCTION)
 }
 return Wallet.getPaymentsClient(activity, builder.build())
}
```


Durant la phase d'intégration, si vous utilisez le code d'exemple, la variable `PAYMENT_MODE` doit être valorisée à **TEST** dans le fichier MainActivity.kt.

10.6. Vérifier si le paiement est possible

Implémentez la méthode **isReadyToPayRequest** pour vérifier si les versions Android et Google Play installées sur l'équipement mobile sont supportées par l'API PaymentsClient.

Suivant la réponse, il sera par exemple possible de cacher/afficher le bouton de paiement.

Extrait de l'exemple de code:

```
private fun prepareIsReadyToPayRequest(): IsReadyToPayRequest {
 val isReadyToPayRequest = getBaseRequest()
 isReadyToPayRequest.put(
  "allowedPaymentMethods", JSONArray()
  .put(getCardPaymentMethod(null, null)))
 return IsReadyToPayRequest.fromJson(isReadyToPayRequest.toString())
}

PayZenPayment.isPaymentPossible(paymentsClient).addOnCompleteListener { task ->
 try {
  val result = task.getResult(ApiException::class.java)
  if (result) {
   // show Google Pay as a payment option
   payBtn.visibility = View.VISIBLE
  } else {
   payBtn.visibility = View.VISIBLE
   Toast.makeText(this, "isPaymentPossible return false", Toast.LENGTH_LONG).show()
  }
 } catch (e: ApiException) {
  Toast.makeText(this, "isPaymentPossible exception catched", Toast.LENGTH_LONG).show()
 }
}
```

10.7. Créer un objet PaymentDataRequest

Lorsque l'acheteur est prêt à payer, créez un objet `PaymentDataRequest` et assignez lui votre objet **BaseRequest**.

Cet objet devra contenir les données utiles au paiement:

- informations de la passerelle de paiement,
- méthodes de paiement supportées par votre application,
- informations supplémentaires que vous souhaitez obtenir dans la réponse.

Définissez les méthodes de paiement supportées en valorisant la propriété `allowedPaymentMethods` avec le résultat de la fonction `getCardPaymentMethod` (voir chapitre Transmettre les méthodes de paiement supportées à la page 14).

Définissez le détail de la transaction dans l'objet `TransactionInfo`:

propriété	totalPriceStatus
valeur	**"FINAL"**
	Paramètre obligatoire lors d'un appel `createPaymentDataRequest()`. Sa valeur ne sera pas prise en compte pour le paiement.

propriété	totalPrice
valeur	"10.00"
	Paramètre obligatoire lors d'un appel `createPaymentDataRequest()`. Cependant seule la valeur transmise lors de l'appel à la méthode `createPayment` de l'API Webservices SOAP v5 sera prise en compte pour le paiement.

propriété	currencyCode
valeur	"EUR"
	Paramètre obligatoire lors d'un appel `createPaymentDataRequest()`. Cependant seule la valeur transmise lors de l'appel à la méthode createPayment de l'API Webservices SOAP v5 sera prise en compte pour le paiement.

Pour plus de détails, consultez la documentation de l'objet TransactionInfo.

Précisez les informations supplémentaires que vous souhaitez obtenir dans la réponse:

propriété	billingAddressRequired
valeur	true
	Rend obligatoire la saisie de l'adresse de facturation. L'adresse sélectionnée sera retournée dans la payload. Utile si vous souhaitez proposer un checkout rapide à l'acheteur.
propriété	**emailRequired**
valeur	true
	Permet de transmettre l'email associé au compte Google sélectionné.
propriété	**shippingAddressRequired**
valeur	true
	Rend obligatoire la saisie de l'addresse de livraison. L'adresse sélectionnée par l'acheteur sera retournée dans la payload. Utile si vous souhaitez proposer un checkout rapide à l'acheteur.
propriété	**phoneNumberRequired**
valeur	true
	Rend obligatoire la saisie du numéro de téléphone dans l'addresse de livraison et de facturation. Si `shippingAddressRequired` ou `billingAddressRequired` est valorisé à `true`, le numéro de téléphone sera transmis dans la payload. Utile si vous souhaitez proposer un checkout rapide à l'acheteur.

Pour plus de détails, consultez la documentation de l'objet PaymentDataRequest.

Extrait du code d'exemple:

```
private fun preparePaymentDataRequest(price: String, currency: String, gatewayMerchantId:
 String): PaymentDataRequest {
    val paymentDataRequestJson = getBaseRequest()
    val additionalParams = JSONObject()
    val transactionJson = JSONObject()
    transactionJson
            .put("totalPriceStatus", "FINAL")
            .put("totalPrice", price)
            .put("currencyCode", currency)
    additionalParams.put("billingAddressRequired", true)
    additionalParams
            .put("billingAddressParameters", JSONObject()
            .put("format", "FULL").put("phoneNumberRequired", false))
    paymentDataRequestJson
            .put("allowedPaymentMethods", JSONArray()
            .put(getCardPaymentMethod(additionalParams,
 getTokenizationSpecification(gatewayMerchantId))))

    paymentDataRequestJson.put("shippingAddressRequired", true)
    paymentDataRequestJson.put("emailRequired", true)
    paymentDataRequestJson.put("transactionInfo", transactionJson)
    return PaymentDataRequest.fromJson(paymentDataRequestJson.toString())
}
```


10.8. Déclarer un gestionnaire d'évenement utilisateur

Définissez un `OnClickListener` sur votre bouton "Payer" afin de déclencher la création de l'objet `PaymentDataRequest`. Utilisez la méthode `resolveTask` de l'`AutoResolveHelper` avec le `PaymentDataRequest` afin d'afficher le Bottom Sheet Google Pay.

Le résultat sera ensuite transmis à votre méthode onActivityResult().

Extrait de l'exemple de code:

```
fun onPayClick(view: View) {
 progressBar.visibility = View.VISIBLE

 val paymentDataRequest = preparePaymentDataRequest()
 AutoResolveHelper.resolveTask(
  paymentsClient.loadPaymentData(paymentDataRequest),
  this,
  GooglePayManagement.GOOGLE_PAYMENT_CODE_RESULT
 )
}
```

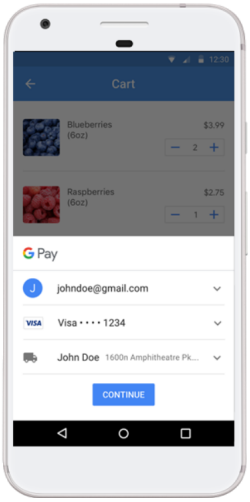

10.9. Sélection du moyen de paiement

Pour pouvoir payer avec Google Pay™, l'acheteur doit avoir enregistré un moyen de paiement en cours de validité dans son wallet Google Pay™.

Afin de réaliser la phase d'intégration dans l'environnement de test, l'intégrateur devra:

- disposer d'un compte Google,
- associer un moyen de paiement valide à son compte Google Pay™.

Lors de l'affichage du bottom sheet Google Pay™, sélectionnez votre carte et cliquez sur le bouton Continuer.



En environnement de test, les données retournées dans l'objet PaymentData sont fictives et ne correspondront pas aux données de la carte enregistrée dans votre compte Google.

Votre carte ne sera donc jamais débitée tant que votre application est configurée dans l'environnement de TEST.

En environnement de test, le message suivant sera affiché dans le bottom sheet Google Pay:
"*Application inconnue. Veuillez continuer uniquement si vous faites confiance à cette application*."

10.10. Récupérer les données de Google Pay™

Les données de l'acheteur sont retournées à l'application mobile dans un objet `PaymentData` (appelé aussi "payload").

Pour traiter les données de la payload, surchargez la méthode `onActivityResult`() de votre `activity` comme suit:

Extrait de l'exemple de code:

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    // Manage Google Pay result
   if (requestCode == GooglePayManagement.GOOGLE_PAYMENT_CODE_RESULT) {
        when (resultCode) {
            Activity.RESULT_OK -> {
                if (data != null) {
                    val paymentData = PaymentData.getFromIntent(data)
                    val googlePayData = paymentData!!.toJson()
                    // Execute payment
                 PayZenPayment.executeTransaction(googlePayData, this)
                } else {
                    PayZenPayment.returnsResult(false,
 PayZenPaymentErrorCode.UNKNOWN_ERROR, "Unknown error", this)
```

```
                }
            }
            Activity.RESULT_CANCELED -> {
                PayZenPayment.returnsResult(false,
 PayZenPaymentErrorCode.PAYMENT_CANCELLED_ERROR, "Payment cancelled by user", this)
            }
            AutoResolveHelper.RESULT_ERROR -> {
                PayZenPayment.returnsResult(false,
 PayZenPaymentErrorCode.UNKNOWN_ERROR, "Unknown error", this)
            }
        }
    }
}
```

Si vous avez opté pour un checkout rapide, vous devez transformer la payload en objet JSON puis l'analyser afin de sauvegarder les données nécessaires (coordonnées de l'acheteur, adresse de livraison etc..).

Exemple de récupération de l'adresse de facturation:

```
when (resultCode) {
 Activity.RESULT_OK -> {
  if (data != null) {
   val paymentData = PaymentData.getFromIntent(data)
   val googlePayData = paymentData!!.toJson()
   val billingAddress =
       JSONObject(
        JSONObject(
         JSONObject(
          googlePayload!!.toJson()
         ).getString("paymentMethodData")
        ).getString("info")
       ).getString("billingAddress")
```

Exemple d'objet PaymentData:


```
JSON result Example :{
 "apiVersionMinor":0,
 "apiVersion":2,
 "paymentMethodData":{
  "description":"Visa •••• 3513",
  "tokenizationData":{
   "type":"PAYMENT_GATEWAY",
   "token":"{\"signature\":\"MEUCIC1+XSqhtgCWQa\/3k7OFIesJX0SYBtCeOKjD
+O2JBWO9AiEAg03FhXVVwD0NRhiPnjGWdfTqTXm39fnzr+XvHdbecY4\\u003d\",\"protocolVersion
\":\"ECv1\",\"signedMessage\":\"{\\\"encryptedMessage\\\":\\\"qjXoUw7ptqWE\/G
+PSOb8QoR+hqzM7h1S7iQ3jH\/mxmjJi\/\/MY5Kem6WmuQxMQbz0pXeX5mxjmUxvbIQzGhtLLELoTWxuAF
\/mLSuoIznPbvJNxQ9xGOVMQYuo0jTKqUUv6QrjMBTSCcQx2ktkDS3EjHOWBPTw22mI\/
kXz0wn5x6jlMLEUTFkNlPZYjeoFRvUXpw0XoECNdouiiJwub52zmyebY7pz1JlG5DQOMWRvrCnj5Kgs9sxzXp1VJJ
+rIqewoBOgl\/BgKsf87fOxVJFYqOmq+LR\/wk8PDKE9VE1XakQBG8kxhSflMQaqC
+ObObJfoUC9ruIDK7iFkT8EFV3FVWHu8ZVYmoek3HWabZ36NdRPUyGtDCjOR+CmT8tneRHyqRzy8S4Yk5LyCObTZ
+rbI0toveNchZdakUhMLyFFLqkgnc\/g3WJnLpujRy3jgOe6\/+ymb1gaiOnQQA\\\\u003d\\\\u003d\\
\",\\\"ephemeralPublicKey\\\":\\\"BKB3qFuCrmNI0bOxVg\/kVCcCtswHrwabcrm18JtPTB+w5L\/
d39dnrgaGajdYvdLrFRQc5RyQ52Ug3e151yDNE6E\\\\u003d\\\",\\\"tag\\\":\\\"rJ5SapWOGc\/f4eqzif+P9D\/
wSg\/yvfOJ63yVbRc1lGY\\\\u003d\\\"}\"}"
  },
  "type":"CARD",
  "info":{
   "cardNetwork":"VISA",
   "cardDetails":"3513",
   "billingAddress":{
    "address3":"",
    "sortingCode":"",
    "address2":"",
    "countryCode":"FR",
    "address1":"109 Rue de l'Innovation",
    "postalCode":"31670",
    "name":"My Network",
    "locality":"Labège",
    "administrativeArea":""
   }
  }
 },
 "shippingAddress":{
  "address3":"",
  "sortingCode":"",
  "address2":"",
  "countryCode":"FR",
  "address1":"109 Rue de l'Innovation",
```

```
   "postalCode":"31670",
   "name":"My Network",
   "locality":"Labège",
   "administrativeArea":""
  },
  "email":"network.gpay@gmail.com"
}
```

11. TRANSMETTRE LES DONNÉES À LA PLATEFORME DE PAIEMENT

Maintenant que l 'application mobile a récupéré les données du paiement, vous devez les transmettre à votre serveur.

Une fois les données reçues coté serveur, vous devez appeler le service web de création de paiement (`createPayment`) de l'API Webservices SOAP v5

Votre requête devra être constituée des éléments suivants:

Les entêtes SOAP	Obligatoire	Permet d'identifier le marchand et de sécurisé les échanges avec la plateforme de paiement.
Un objet **commonRequest**	Facultatif	Permet de forcer le numéro de contrat à utiliser pour réaliser le paiement.
un objet **cardRequest**	Obligatoire	Permet de transmettre la payload à la plateforme de paiement. Une fois déchiffrée, les données seront utilisées pour réaliser le paiement.
Un objet **paymentRequest**	Obligatoire	Permet de transmettres les informations du paiement (montant, devise etc..)
Un objet **customerRequest**	Facultatif	Permet de transmettre les données de l'acheteur.
Un sous-objet **billingRequest**	Facultatif	Permet de transmettre l'adresse de facturation de l'acheteur.
Un sous-objet **shippingRequest**	Facultatif	Permet de transmettre l'adresse de livraison de l'acheteur.
Un objet **shoppingCartRequest**	Facultatif	Permet de transmettre des informations sur le contenu du panier

Remarque:

L'implémentation actuelle de Google Pay™ n'est pas compatible avec le 3D Secure. Il est donc inutile de transmettre l'objet **threeDSRequest** dans votre requête.

Etape 1: Générez le `SOAP-HEADER` comme défini dans la documentation.

Exemple de header:

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Enveloppe xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="http://v5.ws.vads.lyra.com/"
    xmlns:ns2="http://v5.ws.vads.lyra.com/Header/">
  <SOAP-ENV:Header>
    <ns2:shopId>12345678</ns2:shopId>
    <ns2:requestId>70c08d51-214b-4a9b-9e74-eadcdde6710c</ns2:requestId>
    <ns2:timestamp>2018-07-06T13:51:32Z</ns2:timestamp>
    <ns2:mode>TEST</ns2:mode>
    <ns2:authToken>o53KmUXgjqhiyTooIfRjh6wF+NJwVQg790mpbrjTg18=</ns2:authToken>
  </SOAP-ENV:Header>
...
```

Etape 2: Créez un objet **commonRequest** comme défini dans la documentation si vous souhaitez forcer le numéro de contrat à utiliser pour réaliser le paiement.

Etape 3: Créez un objet **cardRequest**:

Attributs de l'objet cardRequest	Valeur
scheme	**GOOGLEPAY**
walletPayload	Objet `PaymentData` au format json tel que reçu par le serveur marchand.

Les attributs **number**, **expiryYear** et **expiryMonth** deviennent facultatifs lorsque l'attribut **scheme** est valorisé à **GOOGLEPAY**.

Etape 4: Créez un objet **paymentRequest** :

Attributs de l'objet cardRequest	Description
amount	Montant à payer. Peut être différent de la valeur transmise dans l'objet `PaymentDataRequest`.
currency	Code numérique de la devise à utiliser pour le paiement. Ex: Ex : 978 pour l'euro (EUR)

Etape 5: Créez un objet **orderRequest**:

Attribut de l'objet orderRequest	Description
orderId	Référence de la commande. Il est recommandé d'utiliser des références de commande unique. En cas de timeout cette référence sera utilisée pour retrouver la transaction correspondante.

Etape 6: **Gestion des adresses de livraison et de facturation**.

Si vous avez opté pour un checkout rapide, les données sont automatiquement transmises à la plateforme de paiement. Vous pouvez passer à l'étape 7.

Si vous gérez la sélection d'adresses depuis votre application mobile, vous pouvez transmettre ces informations à la plateforme de paiement afin qu'elles soient visibles dans le Back Office. Pour cela, créer un objet `customerRequest`.

Etape 6.1: Créez un objet **billingRequest**:

Attributs de l'objet billingRequest	Description
email	Adresse e-mail de l'acheteur. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**email**" de la payload.
address	Première ligne d'adresse de facturation. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**paymentMethodData.Info.billingAddress.address1**" de la payload.
zipCode	Code postal de l'adresse de facturation. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**paymentMethodData.Info.billingAddress.postalCode**" de la payload.
city	Ville de l'adresse de facturation. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**paymentMethodData.Info.billingAddress.locality**" de la payload.
country	Pays de l'adresse de facturation. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**paymentMethodData.Info.billingAddress.countryCode**" de la payload.
firstName	Nom de l'adresse de facturation. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**paymentMethodData.Info.billingAddress.name**" de la payload.
cellPhoneNumber	Numéro de téléphone mobile. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**paymentMethodData.Info.billingAddress.phoneNumber**" de la payload.

Etape 6.2: Créez un objet **shippingRequest**:

Attributs de l'objet shippingRequest	Description
email	Adresse e-mail de l'écheteur. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**email**" de la payload.
address	Première ligne d'adresse de facturation. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**shippingAddress.address1**" de la payload.
zipCode	Code postal de l'adresse de facturation. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**shippingAddress.postalCode**" de la payload.
city	Ville de l'adresse de facturation. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**shippingAddress.locality**" de la payload.
country	Pays de l'adresse de facturation. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**shippingAddress.countryCode**" de la payload.
firstName	Nom de l'adresse de facturation. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**shippingAddress.name**" de la payload.
cellPhoneNumber	Numéro de téléphone mobile. Si vous laissez ce champ vide, la plateforme de paiement récupèrera automatiquement la valeur de la propriété "**shippingAddress.phoneNumber**" de la payload.

Etape 7: Créez un objet **shoppingCartRequest** si vous souhaitez transmettre des informations sur le contenu du panier.

Consultez le Guide d'implémentation - API Webservices SOAP v5 pour plus de détails sur l'intégration de la méthode createPayment().

Exemple de message XML généré lors de l'appel à la méthode createPayment():

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="http://v5.ws.vads.lyra.com/"
    xmlns:ns2="http://v5.ws.vads.lyra.com/Header/">
  <SOAP-ENV:Header>
    <ns2:shopId>12345678</ns2:shopId>
    <ns2:requestId>88c53eb7-6015-4ec6-8045-206e5caa2e2c</ns2:requestId>
    <ns2:timestamp>2018-07-09T10:15:54Z</ns2:timestamp>
    <ns2:mode>TEST</ns2:mode>
    <ns2:authToken>ALxESNJ+mE5htqWccF5OTkYC2noxnDr570DrnI2+5SE=</ns2:authToken>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns1:createPayment>
      <commonRequest>
        <paymentSource>EC</paymentSource>
      </commonRequest>
      <paymentRequest>
        <amount>2990</amount>
        <currency>978</currency>
      </paymentRequest>
      <orderRequest>
        <orderId>myOder</orderId>
      </orderRequest>
      <cardRequest>
        <scheme>GOOGLEPAY</scheme>
        <walletPayload>{"apiVersionMinor":0,"apiVersion":2,"paymentMethodData":{"description":
        "Visa •••• 3513","tokenizationData":{"type":"PAYMENT_GATEWAY","token":"{\"signature\":
        \"MEUCIC1+XSqhtgCWQa\/3k7OFIesJX0SYBtCeOKjD+O2JBWO9AiEAg03FhXVVwD0
        NRhiPnjGWdfTqTXm39fnzr+XvHdbecY4\\u003d\",\"protocolVersion\":\"ECv1\",\"signedMessage
\":\"
        {\\\"encryptedMessage\\\":\\\"qjXoUw7ptqWE\/G+PSOb8QoR+hqzM7h1S7iQ3jH\/mxmjJi\/\/M
        Y5Kem6WmuQxMQbz0pXeX5mxjmUxvbIQzGhtLLELoTWxuAF\/mLSuoIznPbvJNxQ9xGOVM
      QYuo0jTKqUUv6QrjMBTSCcQx2ktkDS3EjHOWBPTw22mI\/kXz0wn5x6jlMLEUTFkNlPZYjeoFR
      vUXpw0XoECNdouiiJwub52zmyebY7pz1JlG5DQOMWRvrCnj5Kgs9sxzXp1VJJ+rIqewoBOgl\/B
      gKsf87fOxVJFYqOmq+LR\/wk8PDKE9VE1XakQBG8kxhSflMQaqC+ObObJfoUC9ruIDK7iFkT8
      EFV3FVWHu8ZVYmoek3HWabZ36NdRPUyGtDCjOR+CmT8tneRHyqRzy8S4Yk5LyCObTZ+rb
      I0toveNchZdakUhMLyFFLqkgnc\/g3WJnLpujRy3jgOe6\/+ymb1gaiOnQQA\\\\u003d\\\\u003d\\\",\\\"e
      phemeralPublicKey\\\":\\\"BKB3qFuCrmNI0bOxVg\/kVCcCtswHrwabcrm18JtPTB+w5L\/d39dnrga
      GajdYvdLrFRQc5RyQ52Ug3e151yDNE6E\\\\u003d\\\",\\\"tag\\\":\\\"rJ5SapWOGc\/f4eqzif+P9D\/wS
      g\/yvfOJ63yVbRc1lGY\\\\u003d\\\"}\"}"},"type":"CARD","info":
{"cardNetwork":"VISA","cardDetails":
      "3513","billingAddress":
{"address3":"","sortingCode":"","address2":"","countryCode":"FR","address1"
      :"109Ruedel'Innovation","postalCode":"31670","name":"Lyranetwork","locality":"Labège",
      "administrativeArea":""}}},"shippingAddress":
{"address3":"","sortingCode":"","address2":"",

 "countryCode":"FR","address1":"109Ruedel'Innovation","postalCode":"31670","name":"MyNetwork",
      "locality":"Labège","administrativeArea":""},"email":"test.gpay@gmail.com"}</
walletPayload>
      </cardRequest>
      <customerRequest>
        <billingDetails>
          <email>test@test.com</email>
        </billingDetails>
        </customerRequest>
        <techRequest/>
      </ns1:createPayment>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


12. TRAITER LE RÉSULTAT DU PAIEMENT

Différents contrôles sont réalisés par la plateforme de paiement lors de la réception de votre requête createPayment.

Ces contrôles peuvent donner lieux à:

- des exceptions (SOAP Fault Exception)
- des erreurs applicatives

En cas d'exception, la réponse contiendra un objet `Fault` détaillant l'erreur rencontrée (erreur de format etc..).

Exemple de réponse lorsque la boutique ne dispose pas des options nécessaires au paiement Google Pay :

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"/>
   <soap:Body>
      <soap:Fault>
         <faultcode xmlns:ns1="http://www.w3.org/2003/05/soap-envelope">ns1:Sender</faultcode>
         <faultstring>
              bad.shopId: The shop with shopId 12345678 is not allowed to call the Web Service
         </faultstring>
         <detail>
            <requestId>f269ff49-3b8a-4314-b999-24b39ce03287</requestId>
         </detail>
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

Si la requête est valide, alors la réponse contient l'objet `commonReponse`.

Le champ `commonReponse.responseCode` donne une indication sur le traitement de la requête.

Lorsque `responseCode` est valorisé à 0 dans la réponse, la transaction a été créée.

Toute autre valeur de `responseCode` indique que la requête a été rejetée avant le paiement.

Le champ `responseCodeDetail` donne le détail de l'erreur rencontrée.

Pour vérifier le statut du paiement, vous devez analyser la valeur du champ `commonResponse.`**`transactionStatusLabel`**

Les seuls status pouvant garantir un paiement accepté sont les suivants:

- **CAPTURED**
- **AUTHORIZED**
- **AUTHORIZED_TO_VALIDATE**

Lorsque `transactionStatusLabel` est valorisé à `REFUSED`, vous devez vérifier la présence du champ `paymentError` dans l'objet `paymentResponse`.

Ce champ donne des indications sur la raison du refus.

Si `paymentError` n'est pas valorisé, analysez le champ `result` de l'objet `authorizationResponse` pour connaître le motif de refus du paiement.

Consultez le Guide d'implémentation - API Webservices SOAP v5 pour obtenir la liste complète des erreurs possibles.

Exemple de réponse renvoyée en cas de paiement accepté

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:ns1="http://v5.ws.vads.lyra.com/"
   xmlns:ns2="http://v5.ws.vads.lyra.com/Header/" ">
<SOAP-ENV:Header>
   <shopId>12345678</shopId>
   <requestId>8bb621fa-8512-45cc-b695-a30fe104cb06</requestId>
   <timestamp>2018-07-11T09:51:00Z</timestamp>
   <mode>TEST</mode>
   <authToken>Le5sPmTnoE8xh3qtU7F1SxhaieO3rh9uLxQpy/4bEu4=</authToken>
</SOAP-ENV:Header>
<soap:Body>
   <ns2:createPaymentResponse>
      <createPaymentResult>
         <requestId>8bb621fa-8512-45cc-b695-a30fe104cb06</requestId>
         <commonResponse>
            <responseCode>0</responseCode>
            <responseCodeDetail>Action successfully completed</responseCodeDetail>
            <transactionStatusLabel>AUTHORISED</transactionStatusLabel>
            <shopId>91335531</shopId>
            <paymentSource>EC</paymentSource>
            <submissionDate>2018-07-11T11:51:00.749+02:00</submissionDate>
            <contractNumber>5555555</contractNumber>
         </commonResponse>
         <paymentResponse>
            <transactionId>927452</transactionId>
            <amount>2990</amount>
            <currency>978</currency>
            <expectedCaptureDate>2018-07-11T11:51:00.791+02:00</expectedCaptureDate>
            <operationType>0</operationType>
            <creationDate>2018-07-11T11:51:00.791+02:00</creationDate>
            <liabilityShift>NO</liabilityShift>
            <transactionUuid>3257eebe44ab4b409315022735c45c2a</transactionUuid>
            <sequenceNumber>1</sequenceNumber>
            <paymentType>SINGLE</paymentType>
         </paymentResponse>
         <orderResponse>
            <orderId>myOder</orderId>
         </orderResponse>
         <cardResponse>
            <number>411111XXXXXX1111</number>
            <scheme>VISA</scheme>
            <brand>VISA</brand>
            <country>GB</country>
            <bankCode>0169</bankCode>
            <expiryMonth>12</expiryMonth>
            <expiryYear>2023</expiryYear>
         </cardResponse>
         <authorizationResponse>
            <mode>FULL</mode>
            <amount>2990</amount>
            <currency>9788</currency>
            <date>2018-07-11T11:51:00.791+02:00</date>
            <number>3fe3c2</number>
            <result>0</result>
         </authorizationResponse>
         <captureResponse/>
         <customerResponse>
            <billingDetails>
               <language>fr_FR</language>
            </billingDetails>
            <shippingDetails/>
            <extraDetails/>
         </customerResponse>
         <markResponse/>
         <threeDSResponse>
            <authenticationResultData>
            <transactionCondition>COND_SSL</transactionCondition>
            </authenticationResultData>
         </threeDSResponse>
         <extraResponse/>
         <fraudManagementResponse>
            <riskControl>
            <name>CARD_FRAUD</name>
            <result>OK</result>
         </riskControl>
         </fraudManagementResponse>
      </createPaymentResult>
   </ns2:createPaymentResponse>
</soap:Body>
</soap:Envelope>
```

13. PASSER EN MODE PRODUCTION

Si votre intégration est correcte, vous avez réussi à créer une transaction Google Pay dans l'environnement de test.

Vous devez maintenant faire valider votre développement auprès de Google.

1. Google met à disposition la liste de points à vérifier:
   https://developers.google.com/pay/api/android/guides/test-and-deploy/integration-checklist
2. Une fois tous les points validés, vous devez remplir le formulaire suivant pour demander la mise en production:
   https://services.google.com/fb/forms/googlepayAPIenable/
3. Lorsque Google vous demandera votre application de production, modifiez la méthode `onCreate` de votre `Activity` comme suit :

```
...
      builder.setEnvironment(WalletConstants.ENVIRONMENT_PRODUCTION)
...
```

   Si vous utilisez le code d'exemple, modifiez la valeur de la variable `PAYMENT_MODE` à **PRODUCTION** dans le fichier MainActivity.kt.

4. Modifiez ensuite l'appel `createPayment` réalisé par votre serveur marchand afin d'utiliser la clé de production dans la génération du `SOAP HEADER`.
5. Modifiez la valeur de l'en-tête `mode` dans votre méthode `createPayment` comme suit:

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Enveloppe xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="http://v5.ws.vads.lyra.com/"
    xmlns:ns2="http://v5.ws.vads.lyra.com/Header/">
  <SOAP-ENV:Header>
    <ns2:shopId>70258842</ns2:shopId>
    <ns2:requestId>b2e0beab-371e-4751-9aa7-20d69daac1ac</ns2:requestId>
    <ns2:timestamp>2018-07-09T12:37:51Z</ns2:timestamp>
    <ns2:mode>PRODUCTION</ns2:mode>
    <ns2:authToken>x7JqR7QLDRc4bsM57nOyf5xzKqlalEqPmeai1EAOZDM=</ns2:authToken>
  </SOAP-ENV:Header>
```

6. Envoyez votre application configurée en mode production à Google pour réaliser les derniers tests.
   L'application devra être signée avec la clé de release. La clé de debug ne fonctionnera pas en environnement de production.
7. Lorsque que Google vous l'autorisera, activez les paiements Google Pay dans le Google Pay Developer Profile pour cette application, puis déployez l'application sur le Google Play Store.
   Pour plus de détails, vous pouvez consulter la procédure de déploiement d'une application :
   https://developers.google.com/pay/api/android/guides/test-and-deploy/deploy-your-application?authuser=1

Après avoir réalisé la mise en production, si vous réalisez un appel avec **WalletConstants.ENVIRONMENT_TEST**, la mention ci-dessous apparaitra lors de l'affichage du bottom sheet Google Pay:
"*Application inconnue. Veuillez continuer uniquement si vous faites confiance à cette application.*"

14. GÉRER VOS TRANSACTIONS GOOGLE PAY™ DEPUIS LE BACK OFFICE PAYZEN

Le Back Office permet différentes actions les transactions Google Pay™ en fonction de leur statut:

Actions disponibles depuis l'onglet **Transactions en cours**:

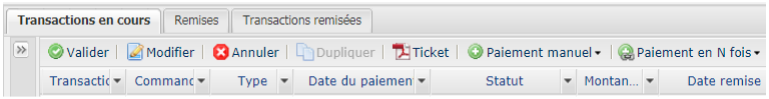
- Afficher le détail de la transaction
- Valider
- Modifier
- Annuler
- Remiser manuellement (uniquement les transactions réalisées dans l'environnement de test)
- Editer la référence de la commande
- Renvoyer l'e-mail de confirmation de la transaction à l'acheteur
- Renvoyer l'e-mail de confirmation de la transaction au marchand.

Actions disponibles depuis l'onglet **Transactions remisées**:

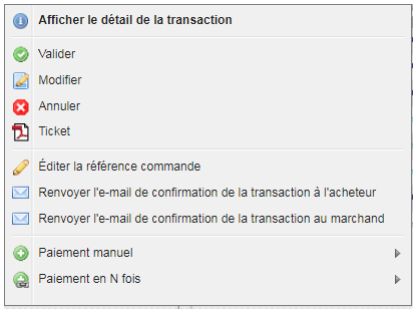
- Rembourser
- Dupliquer
- Editer la référence de la commande
- Renvoyer l'e-mail de confirmation de la transaction à l'acheteur
- Renvoyer l'e-mail de confirmation de la transaction au marchand
- Rapprocher manuellement.

Vous pouvez accéder à ces actions de trois façons différentes:

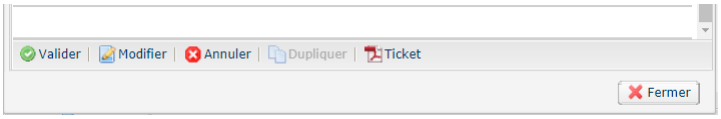
- par la barre de menus.



- par clique droit.



- en bas de la fenêtre de **Détail de la transaction**.



14.1. Afficher le détail de la transaction Google Pay™

Les transactions sont visibles dans le Back Office Marchand depuis le menu **Gestion** > **Transactions**.

Depuis le menu **Gestion**, le marchand a accès aux transactions réelles et aux transactions de TEST.

Remarque :

Suivant ses droits d'accès, les transactions de TEST (exemple : profil développeur) et/ou les transactions réelles (exemple : profil comptable) peuvent s'afficher.

Par défaut l'interface affiche le contenu de l'onglet **Transactions en cours**. Il liste toutes les transactions de la journée.

Spécificités d'un paiement réalisé avec Google Pay™ :

Les paiements réalisés avec succès sont visibles depuis le Back Office Marchand, onglet **Transactions en cours**.

Les paiements en échec sont visibles depuis le Back Office Marchand, onglet **Transactions en cours**.

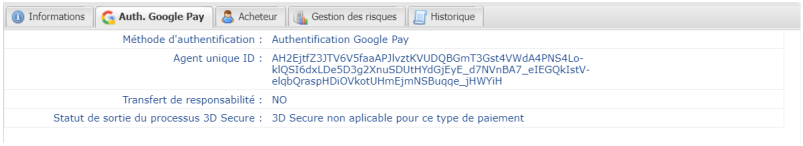
Pour visualiser le détail d'une transaction :

1. Sélectionnez une transaction.
2. Effectuez un clic droit puis sélectionnez **Afficher le détail de la transaction** ou double cliquez sur la transaction à visualiser.

   La boîte de dialogue **Détail d'une transaction en cours** apparaît.

   Parmi les informations présentées, vous trouverez par exemple :

   - le moyen de paiement utilisé
   - la référence de la commande
   - le montant de la transaction
   - la date de création de la transaction
   - le statut de la transaction
   - Le type de portefeuille électronique utilisé : Google Pay™
3. Cliquez sur l'onglet **Auth.Google Pay** pour visualiser les informations relatives à l'authentification Google Pay™.



14.2. Valider une transaction

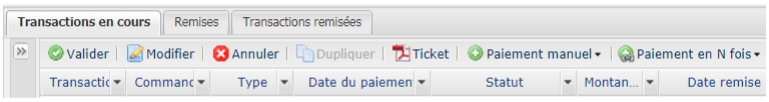
L'action **Valider** est disponible lorsque la transaction possède un des statuts suivant :

- À valider
- À valider et autoriser

Cette action permet de valider la remise en banque de la transaction. Une transaction qui n'a pas été validée avant la date de fin de validité de la demande d'autorisation prendra le statut expiré. Elle ne pourra plus être remise en banque.

Pour valider une transaction:

1. Affichez l'onglet **Transactions en cours**



2. Sélectionnez la transaction.

3. Cliquez sur **Valider**

Une fois la transaction validée, le statut devient:

- "**En attente de remise**" pour les transactions dont le statut était "**À valider**",
- "**En attente d'autorisation**" pour les transctions dont le statut était "**À valider et autoriser**".

14.3. Modifier une transaction

L'action **Modifier** est disponible lorsque la transaction possède un des statuts suivant :

- À valider
- À valider et autoriser
- En attente d'autorisation
- En attente de remise

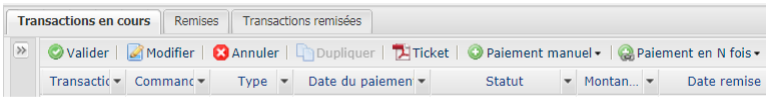
Cette action permet de modifier le montant et la date de remise en banque en respectant les contraintes suivantes:

- le montant modifié ne peut être supérieur au montant initial
- lorsque la transaction n'a pas encore été autorisée, la date de remise peut etre positionnée à n'importe quelle date comprise entre la date du jour et la date de remise spécifiée par le marchand lors du paiement.

  Une demande d'autorisation sera automatiquement déclenchée si la date de remise choisie est comprise entre la date du jour et la date de fin de validité d'une demande d'autorisation (ex: durée de 7 jours pour CB).
- lorsque la transaction a déjà été autorisée, la date de remise en banque ne peut être supérieure à la date de validité de l'autorisation (Ex: 7 jours pour CB).

Pour modifier une transaction :

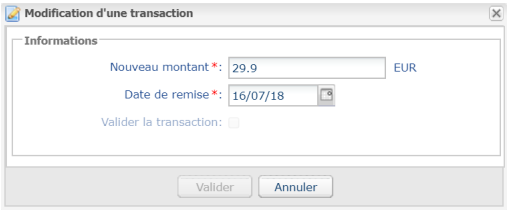
1. Affichez l'onglet **Transactions en cours**



2. Sélectionnez la transaction.

3. Cliquez sur **Modifier**.

   La boîte de dialogue **Modification d'une transaction** s'affiche.



4. Si vous souhaitez modifier le montant de la transaction , renseignez le nouveau montant.

   Pour rappel, le nouveau montant doit être inférieur au montant initial.

5. Si vous souhaitez modifier la date de remise en banque, renseignez la date de remise.

   Le calendrier proposera la plage autorisée pour la date de remise. La plage est calculée en fonction de la durée de validitée de l'autorisation. Cette durée dépend du moyen de paiement et du réseau sur lequel a été réalisée la demande d'autorisation (ex: 7 jours pour CB).

   Il sera possible de valider les transactions ayant un statut **À valider** ou **À valider et autoriser**, en cochant la case **Valider la transaction**.

6. Cliquez sur **Valider**.

Vous pouvez, si vous le souhaitez, afficher le détail de la transaction pour visualiser ces changements (clic droit sur la transaction modifiée > **Afficher le détail de la transaction**).

14.4. Annuler une transaction

L'option **Annuler** est uniquement disponible pour les transactions n'ayant pas été remisées.

1. Effectuez un clic droit sur une transaction.

2. Sélectionnez **Annuler**.

3. Confirmez votre souhait d'annuler définitivement la transaction sélectionnée.

Le statut de la transaction devient **Annulé**.

Remarque

*Il est possible d'**annuler** plusieurs transactions en même temps.*

*Il suffit de sélectionner l'ensemble des transactions à annuler. Vous pouvez vous servir de la **touche Ctrl** et du **clic** pour faire une sélection multiple.*

*Après la sélection, vous pouvez cliquer sur **Annuler** via le clic droit ou à partir de la barre de menu et confirmer votre choix.*

*Les statuts des transactions passeront en **Annulé**.*

14.5. Editer la référence de la commande

Cette opération permet au marchand de changer la référence de commande.

Pour éditer la référence commande d'une transaction :

1. Effectuez un clic droit sur la transaction.
2. Sélectionnez **Editer la référence de la commande**.
3. Saisissez la nouvelle référence de la commande.
4. Cliquez sur **OK**.

14.6. Renvoyer l'e-mail de confirmation de la transaction à l'acheteur

Pour renvoyer l'e-mail de confirmation de la transaction à l'acheteur en cas de non réception ou en cas de correction de l'adresse e-mail.

1. Recherchez la transaction.
2. Effectuez un clic droit sur la transaction.
3. Effectuez un clic droit sur la transaction et cliquez **Renvoyer l'e-mail de confirmation de la transaction à l'acheteur**.
   La boîte de dialogue pour saisir l'adresse e-mail de l'acheteur s'affiche.
4. Saisissez l'adresse e-mail.
5. Cliquez sur **OK**.

14.7. Renvoyer l'e-mail de confirmation de la transaction au marchand

Pour renvoyer l'e-mail de confirmation de la transaction au marchand.

1. Recherchez la transaction.
2. Effectuez un clic droit sur la transaction et cliquez sur **Renvoyer l'e-mail de confirmation de la transaction au marchand**.
   Un message de confirmation d'envoi apparait.
3. Cliquez sur **OK**.

14.8. Remiser une transaction

Cette opération est mis à votre disposition dans les phases de Test. Elle n'est pas disponible en mode Production

L'option **Remiser** est uniquement disponible pour les transactions n'ayant pas atteint la date de présentation.

Pour remiser manuellement une transaction :

1. Affichez l'onglet **Transactions en cours**
2. Effectuez un clic droit sur une transaction.
3. Sélectionnez **Remiser manuellement**.
4. Confirmez que vous souhaitez réellement remiser la transaction sélectionnée.

14.9. Rapprocher manuellement

Cette opération permet de rapprocher manuellement les paiements d'un marchand depuis un extrait de compte.

1. Depuis l'onglet **Transactions remisées**, recherchez la transaction concernée.
2. Effectuez un clic droit sur la transaction.
3. Sélectionnez **Rapprocher manuellement**.
4. Cliquez sur **Oui** pour confirmer le rapprochement manuel de la transaction sélectionnée.
   La boîte de dialogue **Commentaire** s'affiche.
5. Saisissez un commentaire pour ce rapprochement.
6. Cliquez sur **OK**.

Le statut de rapprochement de la transaction devient **Rapproché**.

14.10. Effectuer un remboursement

L'action **Effectuer un remboursement** est disponible lorsque la transaction possède le statut **présenté**.

Cette opération permet de re-créditer le compte d'un acheteur.

Le compte de l'acheteur est crédité du montant remboursé, le compte du marchand est débité de ce même montant.

En fonction de l'acquéreur, il est possible de rembourser une partie ou la totalité du montant de la transaction.

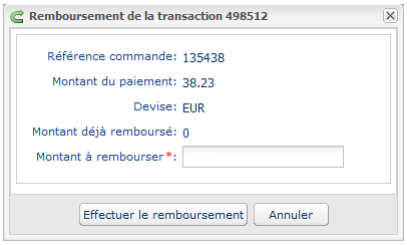
Le délai de remboursement effectif sur les comptes concernés dépend également de l'acquéreur.

Pour effectuer un remboursement :

1. Affichez l'onglet **Transactions remisées**
2. Sélectionnez la transaction.
3. Cliquez sur **Rembourser**.
   La boîte de dialogue **Remboursement de la transaction** s'affiche.

Exemple de remboursement total

Exemple de remboursement partiel



4. Renseignez le montant que vous souhaitez rembourser.
   Le champ de saisie apparaît si le remboursement partiel est possible.
5. Cliquez sur **Effectuer le remboursement**.

Le détail de la transaction de remboursement s'affiche.

