Pay

G Pay

INTEGRATION

Table of Contents

Introduction

PSiGate specializes in the integration and deployment of e-commerce payment service solutions. PSiGate's payment solutions enable automated and secure authorization and fulfillment of credit card transactions. PSiGate communicates directly with major Canadian and U.S. credit card financial institutions and supports businesses that wish to deploy an online storefront.

This document provides merchants and their affiliates with the tools to integrate PSiGate's Google/ Apple Pay Interface so that PSiGate may process their transaction requests.

Test Account Information

To test the transactions in PSiGate's environment, you can follow these steps for Apple Pay and Google Pay™.

1. For **Apple Pay**:
    a. Use the Apple Pay Sandbox environment to test transactions with test payment cards. PSiGate will provide you with the sandbox tester account credentials.
    b. Sign out of iCloud on a valid test device, then sign back in with your sandbox tester account.
    c. In the Wallet app, manually add a new test card. Once added, you can start testing.
    For test card details, refer to this link and check the "Test Cards for Apps and the Web" section.

    Note: Logging in and out of iCloud removes your cards. Test cards work only within the Sandbox environment.
    If you prefer not to use the Apple sandbox testing environment, you can test with your real cards without incurring charges by setting the environmentVariable as "TEST".
2. For **Google Pay**:
    a. You can use the test account details given below in **General Testing Information** but testing for Google Pay™ does not require the sandbox environment. Simply add the card details in Google Wallet.
    b. Real cards can be used in this test environment as Google automatically converts them to test card details within the test environment, ensuring easy setup without sharing real card data with PSiGate.
        i. TEST CARD Details:
            1. Visa: 4111111111111111
            2. Mastercard: 5454545454545454
            3. EXPIRY DATE: Any future date
            4. CVV: any 3 digit number
    c. For 3DS it can be enabled once you are registered with PSiGate. Once you have your own StoreID and Passphrase you can log on to our merchant portal and you can enable 3DS from there.

General Testing Information:

- Log into the test portal at https://staging.psigate.com/MerchantTools/Login/login using:
    - **CID**: 1000001
    - **User**: teststore
    - **Pass**: Testpass1234

Ensure you use the same credentials (i.e. **StoreID**, and **passphrase**) as specified below for passing a Test transaction. For example, passing “teststore" and "Testpass1234," in variables StoreID and Passphrase respectively and sending them as shown further in the document for processing the transaction for google and apple pay individually. Also, don't forget to set environmentVariable as "TEST".

IMPORTANT:

- Do **not** use real card numbers within the test environment (Only for Apple Pay)
- For Production transactions, the Welcome Kit will be provided upon your PSiGate’s account setup. This will have your personal STOREID and PASSPHRASE to process the production transactions and see it in the PSiGate’s Merchant Tools portal.

To send a test transaction you need to follow the below integration steps for Google or Apple pay and use the value of StoreId and Passphrase as mentioned below:

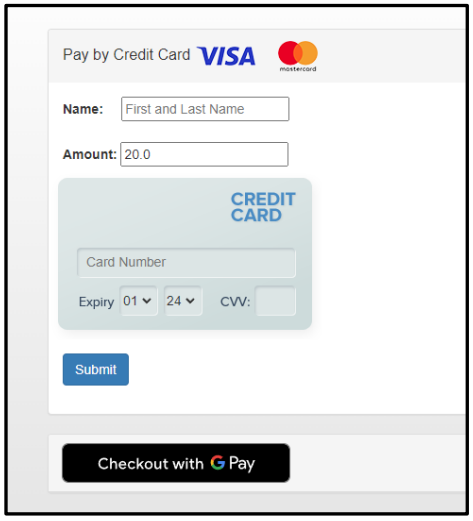
- **StoreID**: teststore
- **Passphrase**: psigate1234

If you need an unshared test account, ***email support@psigate.com***, and we will provide one within a week of your request.

PSiGate merchants can accept payments using Apple Pay and Google Pay™ using one of the two methods elaborated below:

Using HTML Checkout Page

Merchants looking for a simple way to accept payments can use PSiGate’s built-in checkout page, which requires no programming and is convenient for seamless transactions. This method is the default payment option displayed when using a QR code from PSiGate, allowing customers to pay effortlessly. The checkout page automatically displays the appropriate payment icons, such as Apple Pay when accessed on compatible Apple devices, and Google Pay™ as well. Since the entire transaction is processed through PSiGate’s PCI-compliant servers, no additional actions or configurations are required, ensuring a secure, hassle-free experience for both new and existing clients.



Using Realtime-XML APIs

Merchants with customized shopping carts and checkout pages, or those already integrated with PSiGate's Realtime-XML APIs, can easily accept payments via Apple Pay and Google Pay™ using this approach. To simplify the process and reduce dependencies, PSiGate handles the authentication of your website's Domain with Apple, it's registration and manages your merchant account with PSiGate. For Google Pay™, we also manage your account setup and request production access on your behalf to Google, allowing for a smooth integration. This solution offers flexibility and convenience for merchants who prefer a tailored payment experience while ensuring seamless support for both Apple Pay and Google Pay™.

Apple Pay

Apple Pay provides the ability for merchants to accept payments from their customers using AppleWallet. This has the added feature that associated transactions are 3D secure by default.

The minimum requirements for this are as follows:

Incorporating Apple Pay on your website

Admin Level

1. **Requirements**
    a. All pages that include Apple Pay must be served over HTTPS.
    b. Your domain must have a valid SSL certificate.
    c. Your server must support the Transport Layer Security (TLS) protocol version 1.2 or later, and one of the cipher suites listed on the apple documentation.
        i. Setting Up Your Server | Apple Developer Documentation
    d. Allow Apple IP Addresses for Domain Verification from the fallowing IP address ranges from
        i. 17.32.139.128/27
        ii. 17.32.139.160/27
        iii. 17.140.126.0/27
        iv. 17.140.126.32/27
        v. 17.179.144.128/27
        vi. 17.179.144.160/27
        vii. 17.179.144.192/27
        viii. 17.179.144.224/27
        ix. 17.253.0.0/16
    e. Register and verify your domain.
2. **Domain verification process**

    To enable Apple Pay on your domain, you'll need to host the domain verification file provided by PSiGate at a specific location. This file should be accessible at /.well-known/apple-developer-merchantid-domain-association on your website. Domain should be the place where you will be displaying apple pay button.

    For example: if your domain is https://yourdomain.com, ensure the file available at https://yourdomain.com/.well-known/apple-developer-merchantid-domain-association.

Once you've placed the file in the correct location and it's publicly accessible, please notify PSiGate. PSiGate will then proceed to verify and activate your domain for Apple Pay integration.

Developer Level

1. **Add the Apple Pay button**

   PSiGate facilitates seamless integration by providing essential code snippets with step-by-step instructions for effortless addition of the button to your existing checkout page.

   To integrate into your existing html page:

   a. Ensure you load the provided JavaScript file and any associated stylesheet links in the HTML header of your checkout page.

      i. `<script src="https://stagingmobilepay.psigate.com/paymentAction.js" integrity="sha256-810104c05eaea924e7ebeec2ab2bfac1edbb59e24c0937ad34af6ae6828bef3e"></script>`

      ii. `<link rel="stylesheet" type="text/css" href="https://stagingmobilepay.psigate.com/styles.css" integrity="sha256-61b0a0b2382a3da8c5a024606d2b792df36b40170065dcfb5d6896a0cc420f30">`

   b. Place the following ** ** element at the location where you want the Apple Pay button to appear:

```
<div class="apple-pay">
   <div class="apple-pay-button" onclick="applePayButtonClicked()"> </div>
</div>
```

   Please note that if the customer device does not support Apple Pay, this button will be invisible, and you will have to adjust the spacing.

2. **You can also refer to the documentation from Apple**

   a. Displaying Apple Pay Buttons Using CSS | Apple Developer Documentation

3. Please provide the following parameters needed to process the transaction. Please do not change the ID names, just pass the values.

```
<div class="apple-pay">

  <input type="hidden" id="storeid" value="teststore">
  <input type="hidden" id="merchantName" value="businessname">
  <input type="hidden" id="passPhrase" value="psigate1234">
  <input type="hidden" id="subtotal" value="1.0">
  <input type="hidden" id="bname" value="John Smith">
  <input type="hidden" id="baddress1" value="123 Main St.">
  <input type="hidden" id="bCity" value="Toronto">
  <input type="hidden" id="bprovince" value="ontario">
  <input type="hidden" id="bPostalCode" value="L5N2B3">
  <input type="hidden" id="email" value="hello@gmail.com">
  <input type="hidden" id="userID" value="jsmith">
  <input type="hidden" id="environmentVariable" value="TEST">
  <div class="apple-pay-button" onclick="applePayButtonClicked()"></div>
```

```
</div>
<script>
  function finalResponse(transResult){
    console.log("response date recived after processing transaction", transResult);
  }
  function errorCase(error){
    console.log("received error ", error );
  }
</script>
```

4. **Mandatory Fields (Can not be empty)**
    a. **StoreID** - Unique provided to you by PSiGate at the time of onboarding.
    b. **MerchantName** - Name of your store or Company.
    c. **Passphrase** - Unique provided to you by PSiGate at the time of onboarding.
    d. **Subtotal** – Total of the amount to be charged on the card.
    e. **EnvironmentVariable**: TEST or PRODUCTION (it should be passed in capital letters and should not be empty
5. **Additional order detail (Optional fields)**
    a. **Bname** – Billing name
    b. **Baddress1** –Billing address1
    c. **Bpostalcode** – Billing Postal/Zip code (Baddress1 and Bpostalcode elements verify that the inputted street number and postal code match that on record with the issuing bank)
    d. **Email** – Email address to receive E-mail receipt.
    e. **Bcity**- Billing city
    f. **Bprovince** – Billing province. (Please use ISO 3166-2 Province and State codes. Use FIPS 10-4 Region codes for outside Canada and US. Use of these codes will increase IP fraud checking accuracy)
    g. **UserId** - You may use the UserID to track your users or for some other purpose

    Note: You may sort based on the UserID within the Reports.

6. **System Requirements**
    a. Apple Pay: Supported in iOS 10 and later, and macOS 10.12 and later.
7. **Internal working details**
    a. The provided JavaScript function verifies that the device can make Apple Pay Payments.

```
document.addEventListener('DOMContentLoaded', () => {

    if (window.ApplePaySession) {
        if (ApplePaySession.canMakePayments) {
            showApplePayButton();
            console.log("can make paymnets block");
        }
    }
});
```

    b. Once the verification process is complete, the Apple Pay button will be displayed. PSiGate payment gateway supports major credit card brands including American Express, Visa, and Mastercard

```
supportedNetworks: ['amex', 'masterCard', 'visa'],
merchantCapabilities: ['supports3DS'],
```

c. When the user clicks on the Apple Pay button, an ApplePaySession will be initiated. Subsequently, a payment sheet will be displayed according to the payment request.

d. If any of the values, such as amount, store ID, or passphrase, are empty, an error message will be sent to the errorCase() function along with the corresponding message, and an alert prompt will also be displayed in the browser. This ensures that any missing or incomplete information is handled appropriately.


```
if (storeId.trim() === '') {
    console.log("store id value is null");
    session.abort();
    errorCase("store id is null")
    alert("please pass valid storeId")
} else if (passPhrase.trim() === '') {
    errorCase("passphrase is null")
    session.abort();
    alert("please pass passphrase")

} else if (subtotal.trim() === '') {
    errorCase("Amount should be greater than zero")
    session.abort();
    alert("amount should not be equal to zero")
} else {
    getApplePaySession(requestData).then(function (response) {
        console.log(response);
        session.completeMerchantValidation(response);
    });
}
```

e. If all values are present, the system initiates a call to the server, transmitting the validation URL, domain, and business name. Subsequently, the server requests a session from the Apple Pay Server for further processing.

   i. Request: Example

```
const requestData = {
    domain: domain,
    validationURL: validationURL,
    displayName: mechantname
};
```

```
getApplePaySession(requestData).then(function (response) {
    console.log(response);
```

Sample Response: (Just for reference)

- In response, you receive a merchant session object, and it will be passed to complete merchant validation process. You can use the merchant session object a single time. It expiresfive minutes after it is created.

```
session.completeMerchantValidation(response);
```

- After successful merchant validation, you will receive a token which should be passed to the server along with your and the customer's details.
- If any of the steps outlined above encounter an issue, the event will be directed to the onApplePayCancel() method. This results in the cancellation of the payment process.
- The token will then be decrypted, and the transaction will be processed accordingly.
- If everything proceeds smoothly, the token will be decrypted, and the transaction will be processed accordingly.
- Subsequently, upon receiving a success case response from the server, it will be sent to the finalResponse() function.
- Any errors received from the PSiGate server will also be directed to this finalResponse() function.
- In the event of a failed server request, the response error message will be sent to the errorCase() function.
- Please refrain from altering the method names in the HTML code and utilize the received responses as per your application's requirements.

```
<script>
    function finalResponse(transResult){
        console.log("response date recived after processing transaction", transResult);
    }
    function errorCase(error){
        console.log("received error ", error );
    }
</script>
```

- **Sample token: input to the end point.**


```
{"paymentData":{"data":"rmw0ccbJZSTOxEWi3cc1+VifYt8j0Q/QTTtGkLSZPBoax6N2cVVbq16WCNcQtbNBlRjV+8NIi7g0QCGI/Gm1oXxGN4og1YSho9MNcxwuxZxxaEVnyD4MOJO0fHRy5tBzDN+HfFqwShYtvq/3bIbdCtkqA1/
FU21j6fKkYXQZyKWxkk4lq0fkYDi8WXGmEsGfgbGnOwiK4KDWgjA9v7aDeaemw6ok6S3+47ukRIhaq6nB+PEFP4NucPa7T1SBNGQiIgjHzgYwRCqGv+mKkPZgxK3m/m3QhEySPuMVUbN4NYwYmpwC5vOigOxDJ6CM+aRmjOx8eb3cf1Yudo6dqXjPozjdO0NavSkV89YeXMosDmIL3FmXwVvAy
+WXJpnucWVNIye82CjsjvX3Eqso","signature":"MIAGCSqGSIb3DQEHAqCAMIACAQExDTALBglghkgBZQMEAgEwgAYJKoZIhvcNAQcBAACggDCCA
+MwggOIoAMCAQICCEwwQUlRnVQ2MAoGCCqGSM49BAMCMHoxLjAsBgNVBAMMJUFwcGxlIFFwcGxpY2F0aW9uIEludGVncmF0aW9uIENBIC0gRzMxJjAkBgNVBAsMHUFwcGxlIENlcnRpZmljYXRpb24gQXV0aG9yaXR5MRMwEQYDVQQKDApBcHBsZSBJbmMuMQswCQYDVQQGEwJVUzAeFw0xOTA1MTgwMTMyNTdaF
w0yNDA1MTYwMTMyNTdaMF8xJTAjBgNVBAMMHGVjYy1zbXAtYnJva2VyLXNpZ25fVUM0LVBST0QxFDASBgNVBAsMC2lPUyBTeXN0ZW1zMRMwEQYDVQQKDApBcHBsZSBJbmMuMQswCQYDVQQGEwJVUzBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABMIVd
+3r1seyIY9o3XCQoSGNx7C9bywoPYRgldlK9KVBG4NCDtgR80B+gzMfHFTD9+syINa61dTv9JK3iT68DxOjggIRMIICDTAMBgNVHRMBAf8EAjAAMB8GA1UdIwQYMBaAFCPyScRPk+TvJ+bE9ihsP6K7/
S5LMEUGCCsGAQUFBwEBBDkwNzA1BggrBgEFBQcwAYYpaHR0cDovL29jc3AuYXBwbGUuY29tL29jc3AwNC1hcHBsZWFpY2EzMDIwggEdBgNVHSAEggEUMIIBEDCCAQwGCSqGSIb3Y2QFATCB/
jCBwwYIKwYBBQUHAgIwgbYMgbNSZWxpYW5jZSBvbiB0aGlzIGNlcnRpZmljYXRlIGJ6IGFueSBwYXJ0eSBhc3N1bWVzIGFjY2VwdGFuY2Ugb2YgdGhlIHRoZSB3gYXBwbGljYWJsZSBzdGFuZGFyZCB0ZXJtcyBhbmQgY29uZGl0aW9uczBiC1c2UsIGNlcnRpZmljYXRlIHBvbGljeSBhbmQgY2VydGlmaWNhd
GlvbiBwcmFjdGljZSBzdGF0ZW1lbnRzLjA2BggrBgEFBQcCARYqaHR0cDovL3d3dy5hcHBsZS5jb20vY2VydGlmaWNhdGVhdXRob3JpdHkvMDQGA1UdHwQtMCswKaAnoCWGI2h0dHA6Ly9jcmwuYXBwbGUuY29tL2FwcGxlYWljYTMuY3JsMB0GA1UdDgQWBBSUV9tv1XSBhomJdi9
+V4UH55tYJDAOBgNVHQ8BAf8EBAMCB4AwDwYJKoZIhvdjZAYdBAIFADAKBggqhkjOPQQDAgNJADBGAiEAvglXH+ceHnNbVeWvrLTHL+tEKzAVUiLHJRACth69b1UCIQDRizUKXdbdbrF0YDWxHrLOh8
+j5q9svYOAiQ3ILN2qYzCCAu4wggJioAMCAQICCEltL786mNqXMAoGCCqGSM49BAMCMGcxGzAZBgNVBAMMEkFwcGxlIFJvb3QgQ0EgLSBHMzEmMCQGA1UECwwdQXBwbGUgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkxEzARBgNVBAoMCkFwcGxlIEluYy4xCzAJBgNVBAYTAlVTMB4XDTE0MDUwNjIzNDYzMFoXDT
I5MDUwNjIzNDYzMFowejEuMCwGA1UEAwwlQXBwbGUgQXBwbGljYXRpb24gSW50ZWdyYXRpb24gQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkxEzARBgNVBAoMCkFwcGxlIEluYy4xCzAJBgNVBAYTAlVTMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE8BcRh8nXZIXVGl
4lgQd26ICi7957rk3gjfxLk+EzVtVmWzWuItCXdg8iTnu6CP12F86Iy3a7ZnC+yOghP9URaOB9zCB9DBGBggrBgEFBQcBAQQ6MDgwNgYIKwYBBQUHMAGGKmh0dHA6Ly9vY3NwLmFwcGxlLmNvbS9vY3NwMDQtYXBwbGVyb290Y2FnMzAdBgNVHQ4EFgQUI/JJxE+T6O8n5sT2KGw/orv9LkswDwYDVR0TAQH/
BAUwAwEB/zAfBgNVHSMEGDAWgBS7sN6hWDOImqSKmd6+veuv2sskqzA3BgNVHR8EMDAuMCygKqAohiZodHRwOi8vY3JsLmFwcGxlLmNvbS9hcHBsZXJvb3RjYWczLmNybDAOBgNVHQ8BAf8EBAMCAQYwEAYKKoZIhvdjZAYCDgQCBQAwCgYIKoZIzj0EAwIDZwAwZAIwOs9yg1EWmbGG+zXDVspiv/
QX7dkPdU2ijr7xnIFeQreJ+Jj3m1mfmNVBDY+d6cL+AjAyLdVEIbCjBXdsXfM4O6Bn/Rd8LCFtlk/GcmmCEm9U
+Hp9G5nLmwmJIWEGmQ8Jkh0AADGCAYgwggGEAgEBMIGGMHoxLjAsBgNVBAMMJUFwcGxlIEFwcGxpY2F0aW9uIEludGVncmF0aW9uIENBIC0gRzMxJjAkBgNVBAsMHUFwcGxlIENlcnRpZmljYXRpb24gQXV0aG9yaXR5MRMwEQYDVQQKDApBcHBsZSBJbmMuMQswCQYDVQQGEwJVUwIITDBBSVGdVDYwCwYJYIZI
AWUDBAIBoIGTMBgGCSqGSIb3DQEJAzELBgkqhkiG9w0BBwEwHAYJKoZIhvcNAQkFMQ8XDTI0MDQyNDE0NTIwMFowKAYJKoZIhvcNAQk0MSswGTALBglghkgBZQMEAgGhCgYIKoZIzj0EAwIwLwYJKoZIhvcNAQkEMSIEIMJgwt/KrSlu408P65d9rlHOYTccLqQEFzv97he
+O5nlMAoGCCqGSM49BAMCBEcwRQIgMBl+RcoX+Z66HntattioBoMXc+dFleQZfgsgmgfSpH4CIQCwi/cfq7ISclYPuBe9LZSmTJfKLOBNkrHw8ctx9kPlMwAAAAAAAA==","header":{"publicKeyHash":"4Z2AJ44Y6eK5qHvT6szyHuttmGmaeUh5fhJ8KLIObi0=",
"ephemeralPublicKey":"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEfeObCri+zdh4RLAZk2Ihxio5lkFC8i4MEk+RA779f0zV+a/kw/Nm86+TcXs/EijQog5Y+/XZJF+JJOWlkOdb8w==","transactionId":"e459d59ba399beaf5615da7e51cd9a2fb00bdb1799b3218235638c08cf4c7337"},
"version":"EC_v1"},"paymentMethod":{"displayName":"MasterCard 0839","network":"MasterCard","type":"credit"},"transactionIdentifier":"e459d59ba399beaf5615da7e51cd9a2fb00bdb1799b3218235638c08cf4c7337","StoreID":"your_store_id",
"Passphrase":"your_passphrase","SubTotal":"your_subtotal","Bname":"buyer_name","Bcompany":"buyer_company","Baddress1":"buyer_address1","Bpostalcode":"buyer_postal_code","Email":"buyer_email","InAppType":"APAY"}
```


Sample success Response

- PSiGate response for the transaction success case **(Received in finalresponse function. Use it to display the confirmation of purchase screen on your end)**

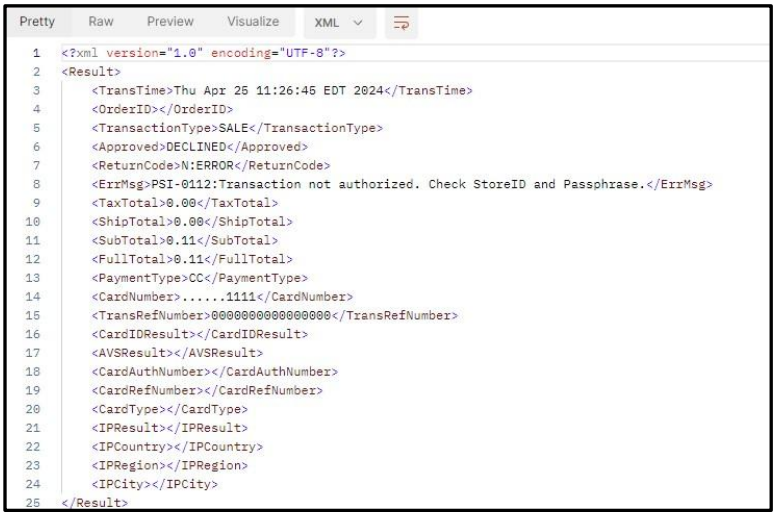

```
Pretty  Raw  Preview  Visualize  XML
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Result>
3      <TransTime>Thu Apr 25 11:25:18 EDT 2024</TransTime>
4      <OrderID>20240425112517097752</OrderID>
5      <TransactionType>SALE</TransactionType>
6      <Approved>APPROVED</Approved>
7      <ReturnCode>Y:123456:0abcdef:::</ReturnCode>
8      <ErrMsg></ErrMsg>
9      <TaxTotal>0.00</TaxTotal>
10     <ShipTotal>0.00</ShipTotal>
11     <SubTotal>0.11</SubTotal>
12     <FullTotal>0.11</FullTotal>
13     <PaymentType>CC</PaymentType>
14     <CardNumber>......1111</CardNumber>
15     <TransRefNumber>1c16d7f0ad729f38</TransRefNumber>
16     <CardIDResult></CardIDResult>
17     <AVSResult></AVSResult>
18     <CardAuthNumber>123456</CardAuthNumber>
19     <CardRefNumber>0abcdef</CardRefNumber>
20     <CardType>VISA</CardType>
21     <IPResult></IPResult>
22     <IPCountry></IPCountry>
23     <IPRegion></IPRegion>
24     <IPCity></IPCity>
25 </Result>
```


Some sample Error Cases

- If the amount value is equal to zero, PSiGate will respond with the fallowing error.


```
Pretty  Raw  Preview  Visualize  XML
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Result>
3      <TransTime>Thu Apr 25 11:27:12 EDT 2024</TransTime>
4      <OrderID></OrderID>
5      <TransactionType>SALE</TransactionType>
6      <Approved>ERROR</Approved>
7      <ReturnCode>N:ERROR</ReturnCode>
8      <ErrMsg>PSI-0106:The total transaction amount is not availabe or is less than or equal to 0.</ErrMsg>
9      <TaxTotal>0.00</TaxTotal>
10     <ShipTotal>0.00</ShipTotal>
11     <SubTotal>0.00</SubTotal>
12     <FullTotal>0.00</FullTotal>
13     <PaymentType>CC</PaymentType>
14     <CardNumber></CardNumber>
15     <TransRefNumber>0000000000000000</TransRefNumber>
16     <CardIDResult></CardIDResult>
17     <AVSResult></AVSResult>
18     <CardAuthNumber></CardAuthNumber>
19     <CardRefNumber></CardRefNumber>
20     <CardType></CardType>
21     <IPResult></IPResult>
22     <IPCountry></IPCountry>
23     <IPRegion></IPRegion>
24     <IPCity></IPCity>
25 </Result>
```

- If the StoreId or Passphrase is not present or incorrect, PSiGate will return the following error.



```
Pretty  Raw  Preview  Visualize  XML
<?xml version="1.0" encoding="UTF-8"?>
<Result>
    <TransTime>Thu Apr 25 11:26:45 EDT 2024</TransTime>
    <OrderID></OrderID>
    <TransactionType>SALE</TransactionType>
    <Approved>DECLINED</Approved>
    <ReturnCode>N:ERROR</ReturnCode>
    <ErrMsg>PSI-0112:Transaction not authorized. Check StoreID and Passphrase.</ErrMsg>
    <TaxTotal>0.00</TaxTotal>
    <ShipTotal>0.00</ShipTotal>
    <SubTotal>0.11</SubTotal>
    <FullTotal>0.11</FullTotal>
    <PaymentType>CC</PaymentType>
    <CardNumber>......1111</CardNumber>
    <TransRefNumber>0000000000000000</TransRefNumber>
    <CardIDResult></CardIDResult>
    <AVSResult></AVSResult>
    <CardAuthNumber></CardAuthNumber>
    <CardRefNumber></CardRefNumber>
    <CardType></CardType>
    <IPResult></IPResult>
    <IPCountry></IPCountry>
    <IPRegion></IPRegion>
    <IPCity></IPCity>
</Result>
```

Access for Production

Once done with the above step and are able to process the transaction by sending the **environmentVariable as "TEST"**, you would have to contact PSiGate Merchant Services team so that we can register you under PSiGate as merchants and once all the verification is done You will receive a welcome kit where it will include 2 main things that is STOREID and PASSPHRASE which will be used to identify your transaction

Once you get the confirmation email from PSiGate that your domain has been registered and you are able to access the production environment you can do the testing by passing/changing the following mandatory variables:

1. **EnvironmentVariable value to "PRODUCTION"**.
2. **StoreID**: Unique ID provided to you by PSiGate
3. **Passphrase**: unique provided to you by PSiGate
4. Change all the link to
    a. <script src="https://mobilepay.psigate.com/**googlePay.js**" integrity= "sha256-79a75cfd7005e4d70b6944887bbdcf32df06b149360f1d9cd49fb60175b2a406"> </script>
    b. <script src="https://mobilepay.psigate.com/**paymentAction.js**" integrity= "sha256-810104c05eaea924e7ebeec2ab2bfac1edbb59e24c0937ad34af6ae6828bef3e"> </script>
    c. <link rel="stylesheet" type="text/css" href="https://mobilepay.psigate.com/**styles.css**" integrity= "sha256-61b0a0b2382a3da8c5a024606d2b792df36b40170065dcfb5d6896a0cc420f30">
5. Now you can use your Real Cards and test the Production transaction.

Google Pay™ API Web Integration

By using Google Pay™, you agree to all Google Pay™ terms and conditions. See the following links for more information:
Acceptable Use Policy
Google Pay API Terms of Service

Google Pay™ provides the ability for merchants to accept payments from their customers using cards stored in google wallet in their google account. This has an added feature which is associated with transactions that are done on the merchant's website of the using the mobile device's browser which are 3D secure by default. PSiGate's merchants can accept Payments using Google Pay™ using one of the two methods described below.

Payment Process Overview

1. The user clicks the Google Pay™ payment button and sees a payment sheet with a list of supported payment methods.
2. The user selects a payment method and Google Pay™ securely returns a payment token for that method to your website's backend.
3. Your backend submits the payment token, along with details about the purchase, to PSiGate's end point.
4. To execute the payment, the PSiGate processes the transaction and sends the response back to you (i.e Fail or Pass).

Steps to integrate Google Pay™ on your website checkout page

Adding Google Pay™ to your website

To add Google Pay™ button on your checkout page, there are few things that are to be added on your own *"checkout page's html"* file. **PSIGATE** will provide a **Master HTML File** so that you can take a reference from that on how to place all the elements discussed below in your Checkout page.

1. Ensure you load the provided JavaScript file in the HTML header of your checkout page.
   ```
   <script src="https://stagingmobilepay.psigate.com/googlePay.js" integrity=
   "sha256-1a16bce922747ec52399018fd1499a49cc441908d4044e7fbc533fa8dcd667f8">
   </script>
   ```
   This tag connects to the PSiGate's Google server, and at that location it contains a file which will help the google button to load on the Merchant's checkout page.
   The above imported JavaScript follows all the steps outlined here in Google Pay Web documentation.
2. The following script tag is used to load the Google Pay™ API JavaScript Library. Please add the following script:
   ```
   <script async src="https://pay.google.com/gp/p/js/pay.js"
   onload="onGooglePayLoaded()"></script>
   ```
3. The google-pay class div is another line of code that you need to add.
   ```
   <div class="google-pay">
           <div id="container_google_button"/>
           <google-pay-button environment="TEST" button-color="default" button-type="subscribe"
            button-locale="checkout"></google-pay-button>
   </div>
   ```

4. Please provide the following parameters needed to process the transaction. Please do not change the ID names, just pass the values.

```
<input type="hidden" id="storeid" value="teststore">
<input type="hidden" id="gatewayMerchantId" value="googletest">
<input type="hidden" id="merchantName" value="businessname">
<input type="hidden" id="passPhrase" value="psigate1234">
<input type="hidden" id="subtotal" value="0.10">
<input type="hidden" id="bname" value="John Smith">
<input type="hidden" id="baddress1" value="123 Main St.">
<input type="hidden" id="bCity" value="Toronto">
<input type="hidden" id="bprovince" value="ontario">
<input type="hidden" id="bPostalCode" value="L5N2B3">
<input type="hidden" id="email" value="hello@gmail.com">
<input type="hidden" id="userID" value="jsmith">
<input type="hidden" id="environmentVariable" value="TEST">
<script src="https://stagingmobilepay.psigate.com/googlePay.js" integrity="sha256-4ba296c5aadf62f4523c27bdd6fd93f05f4ba27e4af6d6f94ba8922822dc369f"></script>

<script async
   src="https://pay.google.com/gp/p/js/pay.js"
   onload="onGooglePayLoaded()"></script>
  <div class="google-pay">
    <div id="container_google_button"/>
    <google-pay-button
        environment="TEST"
        button-color="default"
        button-type='checkout'
        button-locale="en"
    ></google-pay-button>
  </div>
<script>
  function finalResponse(transResult){
    console.log ("response date recived after processing transaction", transResult);
  }
  function errorCase(error) {
    console.log ("received error ", error);
  }
</script>
```

5. **Mandatory fields (cannot be empty)**
    a. **StoreID**: Unique ID provided to you by PSiGate (not in Test Environment)
    b. **Passphrase**: unique provided to you by PSiGate (not in Test Environment)
    c. **SubTotal**: Total of the amount to be charged on the card.
    d. **GatewayMerchantId**: Unique Store ID provided to you by PSiGate
    e. **MerhantName**: Name of your store or Company
    f. **EnvironmentVariable**: TEST or PRODUCTION (it should be passed in capital letters and should not be empty

6. **Optional fields (Can be empty)**
    a. **Bname**: Customer's name buying the product
    b. **Baddress1**: Customer's address buying the product
    c. **Bpostalcode**: Customer's Postal Code buying the product (Baddress1 and Bpostalcode elements verify that the inputted street number and postal code match that on record with the issuing bank)
    d. **Bcity**: Customer's City buying the product
    e. **Bprovince**: Customer's Province buying the product (Please use ISO 3166-2 Province and State codes. Use FIPS 10-4 Region codes for outside Canada and US. Use of these codes will increase IP fraud checking accuracy)
    f. **Email**: Customer's email buying the product
    g. **UserID**: You may use the UserID to track your users or for some other purpose

    Note: You may sort based on the UserID within the Reports.

Doing the above procedure there will be no need for you to handle the encrypted payload or send the request for processing the transaction. Everything will be taken care by PSiGate.

Please check the below points to see how you receive the response from PSiGate and how to check if it is approved transaction or not.

1. After clicking the Google Pay™ button and making a payment, PSiGate processes the transaction request and sends you the confirmation.
2. If everything proceeds smoothly, the token will be decrypted, and the transaction will be processed accordingly.
3. Subsequently, upon receiving a success case response from the server, it will be sent to the finalResponse() function If the transaction was approved or not.
4. Any errors received from the PSiGate server will also be directed to this finalResponse() function.
5. In the event of a failed server request, the response error message will be sent to the errorCase() function
6. **Do not change the name of the functions**. The following are the functions that you need to add in the checkout page's html file:

```
<script>
  function finalResponse(transResult){
    console.log("response date recived after processing transaction", transResult);
  }
  function errorCase(error){ console.log("received
    error ", error );
  }
</script>
```

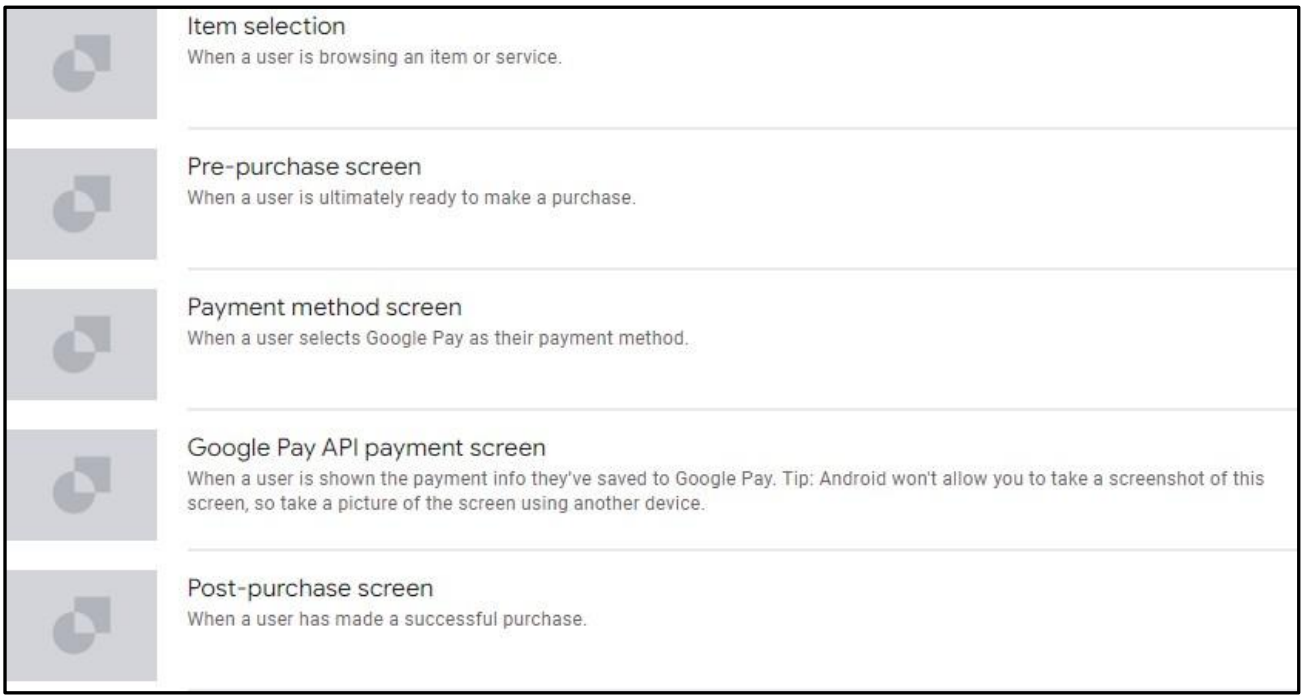
The finalResponse function gives you the responses that are shown on the pages below.

It is a xml response if the transaction was approved or if there was an error because of wrong parameters that were passed to the server.

Access for Production/Domain Registration

Once done with the above step and you are able to process the transaction by sending the **environmentVariable as "TEST"**, you would have to contact PSiGate Merchant Services team so that we can register you under PSiGate as merchants and once all the verification is done You will receive a welcome kit where it will include 2 main things that is STOREID and PASSPHRASE which will be used to identify your transaction and also will apply for the Production access from our end to google for your website's domain and in that email, you would have to add the following images and details:

1. These are the images that we require for the production access
    a. Item selection: When a user is browsing an item or service.
    b. Pre-purchase screen: When a user is ultimately ready to make a purchase
    c. Payment method screen: When a user selects Google Pay™ as their payment method.
    d. Google Pay™ Api Payment screen: When a user is shown the payment info, they've saved to Google Pay™. Tip: Android won't allow you to take a screenshot of this screen, so take a picture of the screen using another device.
    e. Post purchase screen: When a user has made a successful purchase.



2. Along with the images please send us the domain/link where the Google Pay™ button will appear.

    This link should be https enabled and with all the valid certificates as well as secure.

Once you get the confirmation email from PSiGate that your domain has been registered and you are able to access the production environment you can do the testing by passing/changing the following mandatory variables:

6. **EnvironmentVariable value to "PRODUCTION"**.
7. **StoreID**: Unique ID provided to you by PSiGate
8. **Passphrase**: unique provided to you by PSiGate
9. Change all the link to

a. <script src="https://mobilepay.psigate.com/**googlePay.js**" integrity="sha256-79a75cfd7005e4d70b6944887bbdcf32df06b149360f1d9cd49fb60175b2a406"></script>
b. <script src="https://mobilepay.psigate.com/**paymentAction.js**" integrity="sha256-810104c05eaea924e7ebeec2ab2bfac1edbb59e24c0937ad34af6ae6828bef3e"></script>
c. <link rel="stylesheet" type="text/css" href="https://mobilepay.psigate.com/**styles.css**" integrity="sha256-61b0a0b2382a3da8c5a024606d2b792df36b40170065dcfb5d6896a0cc420f30">

Once the above process is done, you'll be able to process the real transactions.

Failed Transaction Scenarios and Error Codes

1. Amount is 0 or null

```
<?xml version="1.0" encoding="UTF-8"?>
<Result>
    <TransTime>Thu Apr 25 11:27:12 EDT 2024</TransTime>
    <OrderID></OrderID>
    <TransactionType>SALE</TransactionType>
    <Approved>ERROR</Approved>
    <ReturnCode>N:ERROR</ReturnCode>
    <ErrMsg>PSI-0106:The total transaction amount is not availabe or is less than or equal to 0.</ErrMsg>
    <TaxTotal>0.00</TaxTotal>
    <ShipTotal>0.00</ShipTotal>
    <SubTotal>0.00</SubTotal>
    <FullTotal>0.00</FullTotal>
    <PaymentType>CC</PaymentType>
    <CardNumber></CardNumber>
    <TransRefNumber>0000000000000000</TransRefNumber>
    <CardIDResult></CardIDResult>
    <AVSResult></AVSResult>
    <CardAuthNumber></CardAuthNumber>
    <CardRefNumber></CardRefNumber>
    <CardType></CardType>
    <IPResult></IPResult>
    <IPCountry></IPCountry>
    <IPRegion></IPRegion>
    <IPCity></IPCity>
</Result>
```

2. StoreID or PassPhrase is not present

```
<?xml version="1.0" encoding="UTF-8"?>
<Result>
    <TransTime>Thu Apr 25 11:26:45 EDT 2024</TransTime>
    <OrderID></OrderID>
    <TransactionType>SALE</TransactionType>
    <Approved>DECLINED</Approved>
    <ReturnCode>N:ERROR</ReturnCode>
    <ErrMsg>PSI-0112:Transaction not authorized. Check StoreID and Passphrase.</ErrMsg>
    <TaxTotal>0.00</TaxTotal>
    <ShipTotal>0.00</ShipTotal>
    <SubTotal>0.11</SubTotal>
    <FullTotal>0.11</FullTotal>
    <PaymentType>CC</PaymentType>
    <CardNumber>......1111</CardNumber>
    <TransRefNumber>0000000000000000</TransRefNumber>
    <CardIDResult></CardIDResult>
    <AVSResult></AVSResult>
    <CardAuthNumber></CardAuthNumber>
    <CardRefNumber></CardRefNumber>
    <CardType></CardType>
    <IPResult></IPResult>
    <IPCountry></IPCountry>
    <IPRegion></IPRegion>
    <IPCity></IPCity>
</Result>
```

Approved Transaction

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Result>
    <TransTime>Thu Apr 25 11:25:18 EDT 2024</TransTime>
    <OrderID>2024042511251709752</OrderID>
    <TransactionType>SALE</TransactionType>
    <Approved>APPROVED</Approved>
    <ReturnCode>Y:123456:0abcdef:::</ReturnCode>
    <ErrMsg></ErrMsg>
    <TaxTotal>0.00</TaxTotal>
    <ShipTotal>0.00</ShipTotal>
    <SubTotal>0.11</SubTotal>
    <FullTotal>0.11</FullTotal>
    <PaymentType>CC</PaymentType>
    <CardNumber>......1111</CardNumber>
    <TransRefNumber>1c16d7f0ad729f38</TransRefNumber>
    <CardIDResult></CardIDResult>
    <AVSResult></AVSResult>
    <CardAuthNumber>123456</CardAuthNumber>
    <CardRefNumber>0abcdef</CardRefNumber>
    <CardType>VISA</CardType>
    <IPResult></IPResult>
    <IPCountry></IPCountry>
    <IPRegion></IPRegion>
    <IPCity></IPCity>
</Result>
```

NOTE: Please check below what and how the Google Pay™ button properties are set, how is it made visible and how is it handling the encrypted payload on PSiGate's side.

Contents in "googlePay.js" file (Just for reference)

This file contains the configuration of the Google Pay™ button and how it processes the transaction. It contains all the basic steps that are shown in the Google Pay Web documentation and it uses all the approved branding provided by Google Pay Web brand guidelines.

1. Define your Google Pay™ API version.
   a.
```
const baseRequest = {
  apiVersion: 2,
  apiVersionMinor: 0
};
```
2. Choose a payment tokenization method.
   a.
```
const tokenizationSpecification = {
  type: 'PAYMENT_GATEWAY',
  parameters: {
    'gateway': 'example',
    'gatewayMerchantId': 'exampleGatewayMerchantId'
  }
};
```
3. Define supported payment card networks
   a. const allowedCardNetworks = ["AMEX", "DISCOVER", "INTERAC", "JCB", "MASTERCARD", "VISA"];

4. Describe your allowed payment methods

a.
```
const allowedCardAuthMethods = ["PAN_ONLY", "CRYPTOGRAM_3DS"];
```
```
const baseCardPaymentMethod = {
  type: 'CARD',
  parameters: {
    allowedAuthMethods: allowedCardAuthMethods,
    allowedCardNetworks: allowedCardNetworks
  }
};
```
```
const cardPaymentMethod = Object.assign(
  {tokenizationSpecification: tokenizationSpecification},
  baseCardPaymentMethod
);
```

5. Load the Google Pay™ API JavaScript library

a.
```
<script
  async
  src="https://pay.google.com/gp/p/js/pay.js"
  onload="console.log('TODO: add onload function')">
</script>
```
(done on Merchant Side)

b.
```
const paymentsClient =
    new google.payments.api.PaymentsClient({environment: 'TEST'});
```

6. Determine readiness to pay with the Google Pay™ API

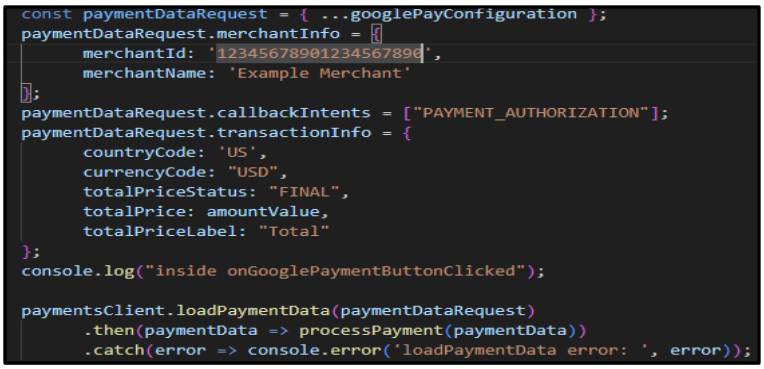
a.
```
const isReadyToPayRequest = Object.assign({}, baseRequest);
isReadyToPayRequest.allowedPaymentMethods = [baseCardPaymentMethod];
```
```
paymentsClient.isReadyToPay(isReadyToPayRequest)
    .then(function(response) {
      if (response.result) {
        // add a Google Pay payment button
      }
    })
    .catch(function(err) {
      // show error in developer console for debugging
      console.error(err);
    });
```

7. Add a Google Pay™ payment button

a.
```
const button =
    paymentsClient.createButton({onClick: () => console.log('TODO: click handler'),
    allowedPaymentMethods: []}); // same payment methods as for the loadPaymentData() API call
document.getElementById('container').appendChild(button);
```

8. Create a PaymentDataRequest object

a.



```
const paymentDataRequest = { ...googlePayConfiguration };
paymentDataRequest.merchantInfo = {
      merchantId: '12345678901234567896',
      merchantName: 'Example Merchant'
};
paymentDataRequest.callbackIntents = ["PAYMENT_AUTHORIZATION"];
paymentDataRequest.transactionInfo = {
      countryCode: 'US',
      currencyCode: "USD",
      totalPriceStatus: "FINAL",
      totalPrice: amountValue,
      totalPriceLabel: "Total"
};
console.log("inside onGooglePaymentButtonClicked");

paymentsClient.loadPaymentData(paymentDataRequest)
      .then(paymentData => processPayment(paymentData))
      .catch(error => console.error('loadPaymentData error: ', error));
```

Encrypted token Generation.

After the load payment data method is invoked the google creates the Payment token and the API call to process the transaction is called along with that the payload consisting of the payment token and the required and optional fields are send to PSiGate where it processes the transaction and sends back the response.

Encrypted Token:

{

"protocolVersion":"ECv2",

"signature":"MEUCIG39tbaQPwJe28U+UMsJmxUBUWSkwlOv9Ibohacer+CoAiEA8Wuq3lLUCwLQ06D2kErxaMg3b/oLDFbd2gcFze1zDqU\u003d",

"intermediateSigningKey":{ "signedKey":

"{\"keyExpiration\":\"1542394027316\",\"keyValue\":\"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE/1+3HBVSbdv+j7NaArdgMyoSAM43yRydzqdg1TxodSzA96Dj4Mc1EiKroxxunavVIvdxGnJeFViTzFvzFRxyCw\\u003d\\u003d\"}",

"signatures":

["MEYCIQDcXCoB4fYJF3EolxrE2zB+7THZCfKA7cWxSztKceXTCglhAN/d5eBgx/1A6qKBdH0IS7/aQ7dO4MuEt26OrLCUxZnl"]

},

"signedMessage":"{\"tag\":\"TjkIKzIOvCrFvjf7/aeeL8/FZJ3tigaNnerag68hIaw\\u003d\",\"ephemeralPublicKey\":\"BLJoTmxP2z7M2N6JmaN786aJcT/L/OJfuJKQdIXcceuBBZ00sf5nm2+snxAJxeJ4HYFTdNH4MOJrH58GNDJ9Uw\\u003d\",\"encryptedMessage\":\"mleAf23XkKjj\"}"

}

The following image shows how the request is sent and how you will get the response.

```
function processPayment(paymentData) {
      console.log(paymentData);
      const storeDetails = {
            StoreID: storeValue,
            Passphrase: passPhrase,
            SubTotal: subtotal,
            UserID:userID,
            Bname: bName,
            Baddress1: baddress1,
            BCity: bCity,
            BProvince: bprovince,
            Bpostalcode: bPostalCode,
            Email: email,
            InAppType:"GPAY",
            environmentVariable : environmentVariable
        };
        if (storeValue.trim() === '') {
            console.log("Store ID value is null");
            errorCase("Store Id value is not valid")
        }else if(passPhrase.trim() === ''){
            errorCase("passphrase value is not valid")
            console.log("passphrase value is not valid");
        }else if (subtotal.trim() === "0.00"){
            errorCase("total amount is null")
            console.log("total amount is null");
        } else{
      return new Promise(function(resolve, reject) {
      setTimeout(function() {
             paymentToken = paymentData.paymentMethodData.tokenizationData.token;
             const requestBody = {
                 ...storeDetails,
                 paymentToken: paymentToken
             };
             console.log("payment token " + paymentToken);
             if (attempts++ % 2 == 0) {
                 reject(new Error('Every other attempt fails, next one should succeed'));
             } else {
                 resolve({});
             }
              fetch('https://stagingmobilepay.psigate.com/api/process/google/xmlpayment', {

                   method: 'POST',
                   headers: {
                         'Content-Type': 'application/json'
                   },
                   body: JSON.stringify(requestBody)
             })
             .then(response => {
                   if (!response.ok) {
                         throw new Error('Network response was not ok');
                         reject(new Error('Every other attempt fails, next one should succeed'));
                   }

                   return response.text();
                                  })
             .then(xmlString => {
                   finalResponse(xmlString);
              });
```