

Google Pay™

overview

Google Pay™ is a digital wallet and online payment system that streamline card payments, enabling users with fast and easy transactions without the need to enter the card details each time you purchase. It allows users to make secure and convenient payments online, and within apps using their mobile devices or web browsers. Google Pay combines the functionalities of a payment platform and a digital wallet, enabling seamless transactions for both consumers and merchants. The Google Pay API enables fast checkout on your website and gives you convenient access to hundreds of millions of cards saved to Google Accounts worldwide.

How It works

The following section describes how a merchant (an e-store owner) can implement google pay and showcase the Google pay Payment sheet on their web site.

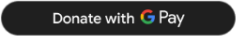
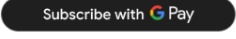
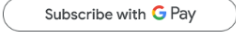
***Note:** If you wish to use BluskysvcPay hosted page. No additional integration efforts will be required.

Branding guideline

if you provide Google Pay as a payment method to your customers, you must use the official Google Pay logo and button assets in compliance with [Google Pay Android brand guidelines](#) and [Google Pay web brand guidelines](#), without modifications to the Google Pay asset colours, proportions, or appearance.

All Google Pay payment buttons exist in two styles: dark and light. You can see examples of both styles in the [Assets section](#). The Google Pay brand name must not be translated. Don't create buttons with your own localized text.

Example: -

Button Type	Dark	Light
book		
buy		
checkout		
donate		
order		
pay		
plain		
subscribe		
Use the button type verbiage that best suits your checkout page.	Use dark buttons on light backgrounds to provide contrast.	Use light buttons on dark or colorful backgrounds.

Google Pay web Integration

You can complete Google Pay integration and start accepting payments using Google Pay just in two steps.

- Integration with Google Pay.
- Send Payment request to the BlyskysvcPay.

* Integration with Google Pay.

***Note:** Before going live as a merchant, you are required to [register with Google Pay & Wallet Console](#). Complete your business profile their and submit it for Google Pay approval.

To get started with Google pay integration for your website, please follow the below steps.

- Follow the [Web developer documentation](#).
- Start with [Setup](#) in the Guides tab.
- Follow the Google pay integration [Tutorial](#). To complete the Google pay integration in the Test environment.
- Follow the Google Pay [Web Integration Checklist](#) and [Web Brand Guidelines](#) to pass the integration review.

- Follow the steps to [Publish your integration](#).

Construct the payment request base configuration.

For more information on the payment request please follow the [Web 101:Basic](#) for a basic request construction and [Web 201:Advanced](#) for advanced request construction under the Guides tab.

```
// Update the `merchantId` and `merchantName` properties with your own values.
// Your real info is required when the environment is `PRODUCTION`.
const merchantInfo = {
  merchantId: '12345678901234567890',
  merchantName: 'Example Merchant'
};

// This is the base configuration for all Google Pay payment data requests.
const baseGooglePayRequest = {
  apiVersion: 2,
  apiVersionMinor: 0,
  allowedPaymentMethods: [
    {
      type: 'CARD',
      parameters: {
        allowedAuthMethods: [
          "PAN_ONLY", "CRYPTOGRAM_3DS"
        ],
        allowedCardNetworks: [
          "AMEX", "DISCOVER", "INTERAC", "JCB", "MASTERCARD", "VISA"
        ]
      },
      tokenizationSpecification: {
        type: 'PAYMENT_GATEWAY',
        parameters: {
          gateway: 'example',
          gatewayMerchantId: 'exampleGatewayMerchantId'
        }
      }
    }
  ],
  merchantInfo
};
```

Please see the below definitions of each object of baseGooglePayRequest

Define the allowedAuthMethods object: -

Google Pay offers two authentication methods.

- **PAN_ONLY:** This authentication method is associated with payment cards stored on file with the user's Google Account. Returned payment data includes personal account number (PAN) with the expiration month and the expiration year.

3Ds required: yes

- **CRYPTOGRAM_3DS:** This authentication method is associated with cards stored as Android device tokens. Returned payment data includes a 3-D Secure (3DS) cryptogram generated on the device.

3Ds required: No

Define allowedCardNetworks object: -

we support VISA, Mastercard, American Express, Discover, Interac, and JCB card networks with the Google Pay API. You can define these values in the allowedCardNetworks property and find the appropriate values in [Google Pay's web developer documentation](#) and [Google Pay's Android developer documentation](#).

Define the tokenizationSpecification object: -

The **gateway** parameter in the request script should be set to the value of “bluskysvcpay”, the value of PSP you are integrating with.

The **gatewayMerchantId** parameter in the request script should be set to the value of the identifier you received from the PSP while on boarding. E.g. MerchantId provided by bluskysvcpay.

Please see the example below to construct your object accordingly.

```
tokenizationSpecification: {
  type: 'PAYMENT_GATEWAY',
  parameters: {
    gateway: 'bluskysvcpay',
    gatewayMerchantId: 'MerchantId provided by bluskysvcpay'
  }
}
```

Define the merchantInfo object

merchantId: - A Google merchant identifier issued after registration with the [Google Pay & Wallet Console](#). Required when PaymentsClient is initialized with an environment property of PRODUCTION. See [Request production access](#) for more information about the approval process and how to obtain a Google merchant identifier. The merchantID can have 12-18 characters.

merchantName: - Merchant name encoded as UTF-8. Merchant name is rendered in the payment sheet. In TEST environment, or if a merchant isn't recognized, a "Pay Unverified Merchant" message is displayed in the payment sheet.

Define the Billing Address properties.

if a billing address is required for Google Pay, ensure that you include required parameters. For more details, see [BillingAddressParameters](#)

These are the optional properties from the Google Pay to be provided in the request script but in some cases, these are conditional for payment provider for AVS verification.

billingAddressRequired:- Set to **true** if you require a billing address. A billing address should only be requested if it's required to process the transaction. Additional data requests can increase friction in the checkout process and lead to a lower conversion rate.

billingAddressParameters: - The expected fields returned if billingAddressRequired is set to true. Please see the [billingAddressParameters](#) for more information.

Handling the Google Pay Response

Once the request is submitted to Google pay. As a response, Google Pay will provide **PaymentData** item. wherein the **paymentMethodData.tokenizationData.token** field will include encrypted **Google Pay Token** (a string of characters). Please see the example below.

Sample Google Pay Token

```
{
```

```

    "signature":
    "MEYCIQCRNIV8yTkhsBCr53C4E1J6xm2eEVJZuwP91EMpeSUtwlhAlclQ0UqDF
    X62lAn2VY1dLbeS2Hy7gbiMOwTlAyhedQ1",
    "intermediateSigningKey": {
        "signedKey":
        "{ \"keyValue\": \"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE+BdYqiyoqri/ZFC0
        U2jFFeIJn6Gn2F7LBcrw60GpgdI5d1wFlY0xlrJ1sp6b2EMxjyYa2+D69UzerZnmo58
        3gg\\u003d\\u003d\", \"keyExpiration\": \"1738234606916\" }",
        "signatures": [

            "MEUCICjTg86r3o/+u6ih6Cb/jp2wH95iFVrGDu+i0Mo/ug6hAiEAppJZYp+8
            en1r6gYmQORpjCuylyXmNVbN3vj3l9lpaYQ="

        ]
    },
    "protocolVersion": "ECv2",
    "signedMessage":
    "{ \"encryptedMessage\": \"OTJNQJDkxaKKpgzliiLZK+kE5ntDL1ejPVf6RoautG7NJh
    6qnwCK6z7W0uQLB6GPdU+hne9VcMrDbEpUICNepEULxiz+2sEOn1qwNtO+5n
    6W6pX7qJ+zoQaVuxRv6hX5++PFVt4tCtFzmjG+3+usOV2G8mVp6u90p3Nkojmbf
    1DK+Qx6YTgRIOPdGoy0X56vxku4mMTTymd8BsHeO3JAKcleF0eV8XK1k1bXqXZ/
    XkKdkG/nf+gvCYVqnSWoQwdCZcvU3YCNu2ilwQd3+ml0xaW3W61K7KkQ3P
    R67njmsqog4+1TACpxNGLSxDorFrA+1/lfjBv2uQm8GY9AcwGn/D1mEH7V1X0x1
    6Avlg+JWmBHgAEaSV42Auk7EqYpe4BOpelWAgJruE0BZO7F/TQ+8wfAPE5TeGKy
    ybDubgxP7bMc1UBcbhLCfyudmVeT/xpmn4ow43VOUfgP4viVVfapuUTQ3boOZdl
    N6370yXQsXO0PXfO1yArlQd3N0etn3lGyAgyF6C5p16+hYnQvTKJglXxPlbIFK\", \"e
    phemeralPublicKey\": \"BAvL9MWm2XfMtJ8vpSQXVbmdR6js9HMKkwHKc/N5nw
    T3YJBxPVdQFG+NbnFAklel/cwKfSe8n2Bq2xVqM/yE0hE\\u003d\", \"tag\": \"7Byjjm
    lTZMxsX6W2vOVvprQVR270PtITM7noMdnu2FM\\u003d\" }"
}

```

Testing: -

While integrating the Google pay, you can test your integration with Google pay Sandbox/test environments. During the integration process you can set the **environment** parameter as **Test** as show in the example below.

```

let paymentsClient = null;

function getGooglePaymentsClient() {
  if (paymentsClient === null) {
    paymentsClient = new google.payments.api.PaymentsClient({
      environment: 'TEST',
      merchantInfo,
      // todo: paymentDataCallbacks (codelab pay-web-201)
    });
  }

  return paymentsClient;
}

```

* Send Payment request to the bluskysvcpay.

Once you get the payment token from the Google Pay response, send the Payment token to the Bluskysvcpay server Api to decrypt the payment Token. Since the Bluskysvcpay Apis are using secure environment, please use the merchant token provided on the Account activation along with the Bearer scheme for successful Authorization. Please use the below Api and the request and response sample to decrypt the payment Token.

Bluskysvcpay Token decryption Api.

<https://api.bluskysvc.com/googlepay/decryptpaymenttoken>

Request Body: - The Payment Token itself as it received from Google.

```

{
  "signature":
  "MEUCIQDgOx2MBmbRGlzU5S8S1GR2VPml5PJhRIhp7VeDlIQZQAlgGmRoOEg
  mA53d7fXWVB/0A8gaQKzxhs7zrQC+Tx82axk\u003d",
  "intermediateSigningKey": {
    "signedKey":
    "{\n\"keyValue\": \"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEwcvjU0idv2mAXj0
    X0vbgYJ5lxdY8Q6k6EBzQz0lcWHD0vZzm/TNb7NcvtORiJDzJ0SjQM8KMHQAzDC
    zbiLQruA\\u003d\\u003d\", \"keyExpiration\": \"1747357990495\"}",
  }
}

```

```

"signatures": [

"MEYCIQDawFPWHaW46LRJ5TrvQmg0iLpPsnDq4ZCQeGL+wtjyOAIhAIWWxxvB
Qqim8kM0rO/1Ew1P6UIWhKu3EWdZy312/Oj9"

],
"protocolVersion": "ECv2",
"signedMessage":
"{\"encryptedMessage\": \"LUXFRBJSiMHImXMszuUbU5qZ/hJOCTVPem0dZij5ZX/
hVmgMMDKyFStnupCjfxjBLWVAHsjLFnhdtdOA+UOxajkw0b+mBUU12LUx029b6
F3r8NZZohv5q4Q7H/382BLuKlh6LHloQoimoF4RR16ldXnOc166BCg93WjT9GL2
Mc1/0M0wiLeo0rhLZBQuVZCEj4XeXETVB0CuqlcVwPBNC4f1g+Fx02sHPdvasLJ9
ghsXL6yobTsIFR62gbsTEJf2xr1rOSWm8TonFfdlejKsGHuggS2+BEYKeGJtWA5tsm
2AnS+zUjaae0FuYQUZzm+gzC7g+ihg0Ze25jCnelsnwVo4VxzhXbaSismpcfdkSRT
Oq1C26hcOn1Gwn3xpXsFN7mi+ndzhQ1jsLOL8iD+JQBCG3xdHz99llp8e963Mfc
RY2GIQQpRjMeMN8TJ4HvxorWskjKdeC2FQ+T3zcWMNHGShH8vrTkKFh7SjHqK
BkOPoVwjikTXXeE3uyL+XX77ySZb6QDd3m/A7FCIGlvfK4YcrowbmisdsDkiTu+Fq3
p7Axmwt\", \"ephemeralPublicKey\": \"BF9s0WZVjO+M8JptOANxaCqNeCGdUzo3
uZzT/HDFZ5f1dxEmzly0XloiLzWTEPw14bC8d8MCnhHqNooPR3wH1IE\\u003d\",
\"tag\": \"CpPbAehL3yM78q3NzouUqU2e0IHGSJqEaikKhXJ5Yo\\u003d\"}
}

```

Response Body: - The Decrypted payment Details from the Payment Token

** Decrypted card details for the Pan_only payment Token*

```

{
  "isDecrypted": true,
  "paymentDetails": {
    "messageExpiration": "1747271590506",
    "messageId":
"AH2EjtdB5k3g2e94RwZu1XmIBTKfq9tVUycg5qipp9kNYINJFyUzyKDDbGHTkf7SE
F0fBuNFM7j9hjNpXR6Zlg3Kykjayn7nJ7qtKsKmh0bX7naN9Cc48do",
    "paymentMethod": "CARD",
    "paymentMethodDetails": {
      "expirationYear": 2027,
      "expirationMonth": 12,
      "pan": "4111111111111111",
      "authMethod": "PAN_ONLY",
      "assuranceDetails": {

```

```

        "cardHolderAuthenticated": false,
        "accountVerified": true
    }
},
"message": "Decryption Successful"
}

```

*** Decrypted card details for the CRYPTOGRAM payment token.**

```

{
  "isDecrypted": true,
  "paymentDetails": {
    "messageExpiration": "1735930317812",
    "messageId": "AH2EjtfnTZbNrOcit4Va_2T0zndg9QQIMfg_wqp3An-
gpTsXWiihfO65fgNMMPb1_KoVDKFAqcY8xb7TlyC98AgGi9vXvisn4WoSE7NrCqns
Dq8wP0L0ANE",
    "paymentMethod": "CARD",
    "paymentMethodDetails": {
      "expirationYear": 2029,
      "expirationMonth": 12,
      "pan": "5204240250197840",
      "authMethod": "CRYPTOGRAM_3DS",
      "eciIndicator": "02",
      "cryptogram": "ALnt+yWSJdXBACMLLWMNGgADFA==",
      "assuranceDetails": {
        "cardHolderAuthenticated": true,
        "accountVerified": true
      }
    }
  },
  "message": "Decryption Successful"
}

```

*** Send payment transactions to the blyskysvc server.**

3D Payment Transaction PAN_ONLY

If the decrypted card details contain the “**authMethod: PAN_ONLY**” and the merchant is enabled for the 3D secure transactions in the Bluskysvcpay system, then merchants need to perform the 3D secure authentication first and then only

the Payment Authorization. To perform the 3D authentication for the “PAN_ONLY” transactions, merchant needs to send the transaction details to the “Checkenrollment” Api to check the card 3D enrolment status. Please see the below transaction request and response for more details.

Checkenrollment Api endpoint: -

<https://api.bluskysvc.com/merchants/{{bluskysvcpaymerchantid}}/checkenrollment>

Request Body:

```
{
  "siteId": "11",
  "amount": 123.45,
  "currency": "USD",
  "firstName": "John",
  "lastName": "Doe",
  "phone": "555-555-5555",
  "addressLine1": "123 Fake St.",
  "addressLine2": "",
  "city": "Hollywood",
  "state": "CA",
  "countryId": "US",
  "postalCode": "12345",
  "email": "",
  "cardNumber": "4024007118676898",
  "nameOnCard": "John J Doe",
  "expirationMonth": 11,
  "expirationYear": 2019,
  "cVVCode": "402",
  "ipAddress": "127.0.0.0",
  "trackingID": "123",
  "isInitialForRecurring": false
}
```

Response Body: -

```
{
  "state": "Verify",
  "status": "Successful",
  "message": "AUTHORIZATION",
  "reasonCode": "1000 - Unknown",
}
```

```

    result": [
        {
            "responseKey": "EnrollmentStatus",
            "responseValue": "Y"
        },
        {
            "responseKey": "Acsurl",
            "responseValue": " https://someacsurl.com"
        },
        {
            "responseKey": "AuthType",
            "responseValue": "doChallenge"
        }
    ],
    "trackingId": 1234 - internal merchant transaction id - optional
    "postbackURL": https://quickpay.bluskysvc.com/verify - this can be
    ignored
    "version": StreamLine
    "id": "181763" – this is the verify transactionID you will send to
    /ThreeDSaleTransactions
}

```

Example form: -

```

<form name="acs" method="post" action="<acsurl_value> ">
<input name="transactionId" type="hidden" value="3605449" /><br />
<input name="AuthType" type="hidden" value="doChallenge" />
<input name="TermUrl" type="hidden"
value="http://merchantssite.com/process3dsale.php" />
<input type="submit" value="Authenticate ACS URL" />
</form>

```

At this point merchant needs to perform an HTML form post to the AcsUrl returned in the “**CheckEnrollment**” Api’s response, to redirect the cardholder’s browser to the issuer’s authentication page. Once the cardholder is redirected to the Issuer’s AcsUrl, the cardholder will be presented with an authentication screen to perform the authentication challenge and once the challenge is successfully performed, merchant will receive the challenge result on the **termUrl** they provided in the HTML form, they will also receive the **Blyskysvcpay**’s transaction Id.

At this point merchant needs to perform the payment Authorization (Sale transaction) and needs to send transaction request to the “**googlePaythreeDtransactions**” Api endpoint.

ThreeDSaleTransaction Api endpoint: -

- <https://api.bluskysvc.com/merchants/{{bluskysvcpaymerchantid}}/googlePaythreeDtransactions>

Request Body: -

```
{
  "VerifyTransactionId": "bluskysvcpay transaction Id",
  "MerchantReferrerUrl": "{{merchantReferrerUrl}}",
  "ConfirmUrl": "{{confirmUrl}}",
  "isPanOnlyThreeD": true
}
```

Response Body: -

```
{
  "state": "Sale",
  "status": "Successful",
  "message": " authorized",
  "resonCode": " 00 - Approved",
  "result": [
    {
      "responseKey": "LiabilityShiftIndicator",
      "responseValue": "Y"
    },
  ],
  "id": "181763"
}
```

CRYPTOGRAM 3D Payment transaction:

If the decrypted card details received from the Blyskysvcpay contains the “**authMethod: CRYPTOGRAM_3DS**”, No additional 3d Authentication needs to be performed. As the decrypted card details already contains the 3D related data. To perform the “**CRYPTOGRAM_3DS**” you need to send the payment request to the below Api endpoint of the Bluskysvcpay with the following payment request.

“cryptogram” and “eciIndicator” values are required to send to the bluskysvcpay request which can be obtained from the decrypted card details

Sale Transaction API: -

- <https://api.bluskysvc.com/merchants/{{bluskysvcpaymerchantid}}/googlePaytransactions>

Request Body: -

```
{
  "ipAddress": "10.210.99.245",
  "siteId": "{{siteId}}",
  "amount": "{{amount}}",
  "currency": "{{currency}}",
  "firstName": "{{firstName}}",
  "lastName": "{{lastName}}",
  "email": "{{email}}",
  "phone": "{{phone}}",
  "addressLine1": "{{addressLine1}}",
  "city": "{{city}}",
  "countryId": "{{countryId}}",
  "postalCode": "{{postalCode}}",
  "state": "{{state}}",
  "nameOnCard": "{{nameOnCard}}",
  "cardNumber": "{{cardNumber}}",
  "expirationMonth": "{{expirationMonth}}",
  "expirationYear": "{{expirationYear}}",
  "cvvCode": "{{cvvCode}}",
  "isInitialForRecurring": "{{isInitialForRecurring}}",
  "confirmUrl": "{{confirmUrl}}",
  "merchantReferrerUrl": "{{merchantReferrerUrl}}",
  "referrerURL": "{{referrerURL}}",
  "eci": “eciIndicator value from the decrypted token”,
  "acsVerificationId": “cryptogram value from the decrypted token”,
  "googlePayOrigin": "web or mobile", (Default is web)
  "isGooglePayTransaction": true
}
```

Response: -

```
{
  "id": "123456",
```

```
"message": "Success",  
"state": "Sale",  
"status": "Successful"  
"trackingid": "123456"  
"reasoncode": "00-approved"  
}
```