



QORPAY · GOOGLE PAY

Integration Guide

Accept Google Pay through the QorCommerce Purchase API.



02

Contents

01	How Google Pay works with QorCommerce	3
02	Choose your integration path	4
	Hosted Checkout	5
	The QorCommerce SDK	6
03	Configure the Google Pay request	7
	Authorization methods and card networks	8
	Billing address	9
04	Charge the payload	10
	Request parameters, part 1	11
	Request parameters, part 2	12
	The google object	13
	A complete request	14
	Headers, responses, and getting help	15
	Index	16



03

CHAPTER 01

How Google Pay works with QorCommerce

Google Pay lets your customers pay with the cards stored in their Google Account, so checkout finishes without anyone re-entering card details. QorCommerce is a supported payment processor for the Google Pay API, and this guide covers everything between the button and the charge.

The integration has two halves. On the client, the customer taps the Google Pay button and Google returns an encrypted payment payload. On the server, you pass that payload to the QorCommerce Purchase API as a single field named google, on the same sale endpoint you already use for keyed transactions.

QorCommerce integrates as a Google Pay gateway, which means we decrypt the payment data for you. The card number never appears in your systems, in your logs, or in this API in the clear: it exists only inside the encrypted payload, and only QorCommerce can open it.

- 1

Client: collect the payment
Your checkout shows the Google Pay button. The customer confirms, and Google returns a PaymentData object containing an encrypted token.
- 2

Server: charge it
You parse the token and send it to the Purchase API as the google field. QorCommerce decrypts, charges the card, and answers like any other sale.

BRAND ASSETS

If you display Google Pay marks or buttons in your own UI, use the official, unmodified Google Pay logo and button assets. Review the Google Pay brand guidelines for Android and web before you launch.

GOOGLE POLICIES

Every merchant accepting Google Pay must follow the Google Pay and Wallet APIs Acceptable Use Policy and accept the Google Pay API Terms of Service. Both are linked from the Google Pay developer documentation.



04

CHAPTER 02

Choose your integration path

The server half of the work is the same everywhere: send the google payload to the Purchase API. The client half depends on where your checkout lives, and on how much of the Google Pay request you want QorCommerce to assemble for you.

Follow the Google Pay developer documentation for your platform alongside this guide, and run the matching integration checklist before launch.

PLATFORM	FOLLOW	BEFORE LAUNCH
Android	Google Pay Android developer documentation and the Android brand guidelines.	The Android integration checklist.
Web	Google Pay Web developer documentation and the web brand guidelines.	The web integration checklist.

Two client paths follow, one per page. Hosted Checkout is the shortest: QorCommerce renders the checkout page, and Google Pay appears as a payment option once enabled. The QorCommerce SDK keeps checkout in your own page and builds the Google Pay request objects, the `IsReadyToPayRequest` and the `PaymentDataRequest`, on your behalf.

Whichever path you take, chapter 03 covers the request configuration that identifies QorCommerce as your gateway, and chapter 04 covers the charge.



05

Hosted Checkout

Hosted Checkout carries Google Pay as a built-in payment option. Three things make it live.

1

Confirm your SDK version

Run your integration on an SDK version that supports Google Pay. A Content-Security-Policy update may be required so the Google Pay scripts can run on your checkout page.

2

Enable Google Pay for your account

Turn Google Pay on in your merchant dashboard. Until it is enabled, the button does not render.

3

Accept the Google policies

Follow the Google Pay and Wallet APIs Acceptable Use Policy and accept the Google Pay API Terms of Service.

4

Verify on the page

Open your hosted checkout and confirm the Google Pay button renders and completes a test payment end to end.

IF SOMETHING LOOKS WRONG

A missing button usually means Google Pay is not enabled on the account or the SDK version predates support. A button that renders but fails on tap is most often the Content-Security-Policy blocking the Google Pay scripts; check the browser console first.



06

The QorCommerce SDK

With the SDK, checkout stays in your page and the SDK generates the `IsReadyToPayRequest` and `PaymentDataRequest` objects on your behalf.

1

Load the Google Pay client library

Add the Google Pay API JavaScript client library, `pay.js` from `pay.google.com`, to your checkout page.

2

Provision a Google Merchant ID

Register with the Google Pay and Wallet Console and provision a Merchant ID, then enter that value into the QorCommerce SDK configuration.

3

Accept the Google policies

Follow the Google Pay and Wallet APIs Acceptable Use Policy and accept the Google Pay API Terms of Service.

4

Let the SDK do the request work

With the library loaded and the Merchant ID configured, the SDK assembles the Google Pay request objects and returns the encrypted payload for your server to charge, chapter 04.

TIP

Building the request yourself instead of using the SDK is fully supported: chapter 03 documents the values QorCommerce needs inside the `PaymentDataRequest`.



07

CHAPTER 03

Configure the Google Pay request

QorCommerce uses the Google Pay gateway integration type: the payload is encrypted to us, and we decrypt it on your behalf. Two values in the tokenizationSpecification of your PaymentDataRequest make that binding.

FIELD	SET IT TO
gateway	Always qorcommerce, the gateway ID QorCommerce is registered under with Google.
gatewayMerchantId	The identifier you use for this merchant in your own system, typically the same value you use as your QorCommerce mid.

```
● ● API CONFIGURATION EXAMPLE
"tokenizationSpecification": {
  "type": "PAYMENT_GATEWAY",
  "parameters": {
    "gateway": "qorcommerce",
    "gatewayMerchantId": "<Merchant ID assigned by you>"
  }
}
```

NOTE

Everything else in the PaymentDataRequest, allowed methods, networks, and billing address, follows on the next two pages.



08

Authorization methods and card networks

Google Pay returns card credentials in one of two formats. Enable the methods in `allowedAuthMethods` that match what you configured during technical onboarding with QorCommerce.

METHOD	WHAT IT IS	3-D SECURE	COUNTRIES
PAN_ONLY	The physical card details stored in Google Pay.	Yes. The standard 3DS flow applies.	US, Canada, Japan
CRYPTOGRAM_3DS	A tokenized virtual card stored on the device.	No. Google Pay performs the authentication.	US only

If you already run 3-D Secure for other payment methods, the same flow applies to `PAN_ONLY` credentials: pass the credential returned in the encrypted payload through your existing 3DS trigger exactly as you would a manually keyed card.

CARD NETWORKS

Set `allowedCardNetworks` to the networks you were approved for during technical onboarding. QorCommerce supports VISA, Mastercard, American Express, Discover, Interac, and JCB. The exact network constant values are in the Google Pay request-object reference for web and Android.

TIP

Approved networks and authorization methods are account configuration, not code. If a method you enable client-side was not set up at onboarding, ask your QorCommerce contact before debugging the integration.



09

Billing address

If you require a billing address for address verification, request it in the `PaymentDataRequest` with `billingAddressRequired` and `billingAddressParameters`. The full option set is in the `BillingAddressParameters` reference in the Google Pay documentation.

```
● ● BILLING ADDRESS REQUEST EXAMPLE
"allowedPaymentMethods": [
  {
    "type": "CARD",
    "parameters": {
      "billingAddressRequired": true,
      "billingAddressParameters": {
        "phoneNumberRequired": true,
        "format": "MIN"
      },
      "allowedCardNetworks": [...],
      "allowedAuthMethods": [...]
    },
    "tokenizationSpecification": {...}
  }
]
```

NOTE

The billing fields you send to the Purchase API, `bzip` and the `baddress` family on page 11, are separate from what Google collects here. Requesting the address from Google fills your checkout; sending it on the sale request is what feeds address verification.



CHAPTER 04

Charge the payload

When the customer completes checkout, Google Pay returns a `PaymentData` object to your client. Charging it is a standard Purchase API sale with one difference: the card rides inside a single `google` field.

1

Take the token

Read `paymentMethodData.tokenizationData.token` from the `PaymentData` response. It is a JSON string containing the encrypted payload.

2

Parse it

Parse the string as JSON. Do not modify, re-key, or unwrap anything inside it; every field is part of the signed, encrypted payload.

3

Send it as google

POST the sale to `api-sandbox.qorcommerce.io /v3/payment/sale/manual/` with the parsed object as the `google` field, alongside your normal sale fields, page 11. QorCommerce recognizes the transaction as Google Pay from the presence of the `google` object; there is no separate endpoint and no `transaction-type` flag.

4

Read the response

The call answers like any other Purchase request, with the standard payment response codes.

WARNING

Do not include `creditcard`, `month`, `year`, or `cvv` on a Google Pay transaction. That card data is only ever available to QorCommerce inside the encrypted google payload, never in the clear.

NOTE

Like other Purchase requests, the sale is also reachable at your shared sale endpoint, `/v3/payment/sale/`, with the same `google` parameter. Production uses your assigned API host in place of the sandbox host.



Request parameters, part 1

The transaction_data object carries the sale. Core and billing fields first; customer and recurring fields follow on the next page.

PARAMETER	DESCRIPTION	REQUIRED
mid	The merchant id assigned by QorCommerce or your merchant service provider.	Yes
amount	The total amount to process.	Yes
google	The encrypted Google Pay payment token, parsed from paymentMethodData.tokenizationData.token. Page 13.	Yes
orderid	Unique merchant order tracking number.	Yes
bzip	The card billing zip code.	Yes
tid	Terminal id for processing. A reference value, required for MarketPlace transactions.	No
topt	Type of payment transaction. Set a value to tag or route a transaction to a pre-defined provider.	No
reference_id	A reference id echoed back in the message response.	No
service_charge	The portion of the total made up of the service charge.	No
baddress` / `baddress2	The card billing street address, and its second line.	No
bcity` / `bstate	The card billing city and state code.	No
bcountry	The card billing country code. Default: US.	No
currency	The currency to process. Defaults to USD.	No
invoiceid	The merchant invoice id associated with the sale.	No



12

Request parameters, part 2

Customer, risk, and recurring fields. All are optional.

PARAMETER	DESCRIPTION
ipaddress	The IP address of the client or server sending the request.
cfirstname` / `clastname	The customer first and last name.
cemail` / `cphone	The customer email address and phone number.
cwebaddress	The customer website.
risk_score	Risk score associated to this transaction, when you run a risk prevention system.
is_initial	First transaction in a recurring or installment purchase.
is_recurring	Whether this is a recurring transaction. Default 0; set 1 on the next transaction in the installment.
first_trxn	Original transaction id of the first payment. On a recurring transaction, set to the first transaction id of the installment purchase.
metadata	Key-value pairs attached to the payment, for storing additional information in a structured format.

TIP

Recurring on Google Pay follows the same pattern as recurring on keyed cards: mark the first charge with is_initial, then send is_recurring 1 and first_trxn on the installments that follow.



The google object

The google object is the encrypted payment token itself, exactly as Google Pay returns it. Every field is required because every field is part of the signed, encrypted payload; removing or altering any of them makes decryption fail.

FIELD	DESCRIPTION
protocolVersion	The protocol Google Pay used to encrypt the payload, for example ECv2.
signature	The signature over signedMessage, used to verify the payload has not been tampered with.
intermediateSigningKey	Object containing the signed intermediate public key used to verify signature. Present for ECv2.
intermediateSigningKey.signedKey	JSON string containing the intermediate public key value and its expiration timestamp.
intermediateSigningKey.signatures	Array of signatures over signedKey.
signedMessage	JSON string containing the encryptedMessage, ephemeralPublicKey, and tag needed to decrypt the underlying payment credentials.

WARNING

Pass the object through untouched. Parse the token string, attach the result as google, and send it. Any transformation, re-encoding, or field pruning breaks the signature and the charge declines at decryption.

IF SOMETHING LOOKS WRONG

A decryption failure almost always means the payload was altered in transit through your stack: a JSON library re-ordering or re-escaping strings, a proxy trimming fields, or a double parse. Log the exact string you received from Google and compare it to what you sent.



14

A complete request

A full sale with a Google Pay payload. The long cryptographic strings are shortened here for print; send them exactly as received.

```
● ● ● JSON EXAMPLE · POST /v3/payment/sale/manual/
{
  "transaction_data": {
    "mid": "887728202",
    "amount": "1.5",
    "service_charge": "0.00",
    "google": {
      "signature": "MEUCIFkxQ4KUIzHYMJ4yDnrnW1Zs/qbiMZFXp4BXEDX13wX3AiEA5PRS...",
      "intermediateSigningKey": {
        "signedKey": "{\"keyValue\":\"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE6BQB...\",
          \"keyExpiration\":\"1785346073000\"}",
        "signatures": ["MEYCIQCudrUgQUqDFNwsDu97JxdsDOkORWT0ZdHDWHRHKmpDTwIhAIO2..."]
      },
      "protocolVersion": "ECv2",
      "signedMessage": "{\"encryptedMessage\":\"pebGwzUkQ6UtKVUaLXcQ7KqLvKz3ghT9...\",
        \"ephemeralPublicKey\":\"BJ9CoEjd7APGLHE5Dq3TXhtPZ2rc0Ywh...\",
        \"tag\":\"Al/naHqBCq14W/tGu+mNvdqzFGsjSuG7zqpL+CLbV/w=\"}"
    },
    "bzip": "92382",
    "currency": "USD",
    "invoiceid": "INV-123332",
    "orderid": "193383",
    "ipaddress": "0.0.0.0",
    "baddress": "18312 All View Dr.",
    "baddress2": "Apt 207",
    "bcity": "Running Springs",
    "bstate": "CA",
    "bcountry": "US",
    "cfirstname": "Russell",
    "clastname": "Hanneman",
    "cemail": "russell@example.com",
    "cphone": "1-344-877-8928",
    "cwebaddress": "example.com",
    "risk_score": "0"
  }
}
```



Headers, responses, and getting help

Two headers authenticate every request. Test values are shown; production keys are issued for your account.

HEADER	DESCRIPTION
Qor-App-Key	String, required, length up to 24. The unique key identifying the app making this request. For testing: T6554252567241061980.
Qor-Client-Key	String, required. Represents an authorized merchant account, assigned by QorCommerce or your merchant service account provider. For testing: 01dffeb784c64d098c8c691ea589eb82.

The body is a single required object, transaction_data. A successful call answers 200 OK; result details follow the standard Payment Response Codes, documented at docs.qorcommerce.io.

When something does not behave as this guide describes, write to your QorCommerce contact with the request you sent (with the google payload redacted), the response you received, and the reference_id or orderid. Integration questions about the client side belong with the Google Pay developer documentation for your platform.



16

Index

3	
3-D Secure · 8	
A	
allowedAuthMethods · 8	
allowedCardNetworks · 8	
amount · 11	
B	
Billing address (Google request) · 9	
Brand guidelines · 3	
bzip and baddress family · 9, 11	
C	
Card networks supported · 8	
CRYPTOGRAM_3DS · 8	
CSP (Content-Security-Policy) · 5	
D	
Decryption failures · 13	
G	
gateway / gatewayMerchantId · 7	
Google Merchant ID · 6	
google object · 10, 13, 14	
Google policies · 3, 5, 6	
H	
Headers (Qor-App-Key, Qor-Client-Key) · 15	
H	
Hosted Checkout · 5	
I	
IsReadyToPayRequest · 4, 6	
M	
metadata · 12	
mid · 7, 11	
O	
orderid · 11	
P	
PAN_ONLY · 8	
pay.js client library · 6	
PaymentData token · 10	
PaymentDataRequest · 4, 6, 7, 9	
R	
Recurring fields · 12	
Response codes · 15	
S	
Sale endpoint · 10, 14	
SDK (QorCommerce) · 6	
signedMessage · 13	
T	
tokenizationSpecification · 7	