Receive payments via payment gateways on WhatsApp | Developer Documentation

Receive payments via payment gateways on WhatsApp

Updated: Dec 12, 2025
Your business can enable customers to pay for their orders through our partner payment gateways without leaving WhatsApp. Businesses can send customers order_details messages, then get notified about payment status updates via webhook notifications.

Overview

Currently, customers browse business catalogs, add products to cart, and send orders with our set of commerce messaging solutions, which includes Single Product Message, Multi Product Message, and Product Detail Page. Now, with the Payments API, businesses can send customers a *bill*, so the customer can complete their order by paying the business without having to leave WhatsApp.
Our payments solution is currently enabled by BillDesk, Razorpay, PayU and Zaakpay, a third-party payments service provider. You must have a BillDesk, Razorpay, PayU or Zaakpay account in order to receive payments on WhatsApp.
We expect more payment providers to be added in the future.

How it works

First, the business composes and sends an `order_details` message. An `order_details` message is a new type of `interactive` message, which always contains the same 4 main components: **header**,

body, **footer**, and **action**. Inside the `action` component, the business includes all the information needed for the customer to complete their payment.
Each `order_details` message contains a unique `reference_id` provided by the business, and that unique ID is used throughout the flow to track the order.
Once the message is sent, the business waits for a payment status update via webhooks.
Businesses get notified when the payment status changes, but they must not solely rely on these webhooks notifications due to security reasons. WhatsApp also provides a payment lookup API that can be used to retrieve the payment statuses directly anytime.

Purchase flow in app

In the WhatsApp Messenger App, the purchase flow has the following steps:
Customers send an order with selected products to the business either through simple text messages or using other interactive messages such as Single Product Message, Multi Product Message, and Product Detail.
Once the business receives the order, they send an `order_details` message to the user. When the user taps on **Review and Pay**, they will see details about the order and total amount to be paid. When the user taps the **Continue** button, they are able to choose to pay natively on WhatsApp or any other UPI app.

Checkout with WhatsApp Pay:
[image removed - too large for import]

Checkout on other UPI Apps:
[image removed - too large for import]
Once the payment has been confirmed by your payment gateway (PG) or payment service provider, the business can start processing the order.
Businesses can then send an `order_status` message to the consumer informing them about the status of the order. Each message will result in a message bubble (as shown below) that refers to the original order details message and also updates the status displayed on the order details page.

Link your payment account

To receive payments on WhatsApp, you must add a *payment configuration* to the corresponding WhatsApp Business Account. A payment configuration allows you to link a payment gateway account to WhatsApp. Each payment configuration is associated with a *unique name*. As part of the `order_details` message, you can specify the payment configuration to use for a specific checkout. WhatsApp will then generate a checkout flow using the associated payment gateway account.
[image removed - too large for import]
After linking your payment partner account, you must integrate with the Payments APIs below. This will allow you to send an `order_details` message to customers with the payment configuration to receive payments.

Steps to unlink Payment Configuration

Note: Make sure no new order messages requesting payment from consumer are sent with the payment config your are trying to remove before you perform the unlink action.

Integration Steps

The steps outlined below assume that the business already knows what the user is interested in through earlier chat threads. The Payments API is a standalone API and hence can work with various messages such as List Messages, Reply Buttons, Single or Multi-Product Messages.

Sequence Diagram

The following sequence diagram demonstrates the typical integration flow for Payments API. The steps highlighted in green are the key integration steps.
[image removed - too large for import]

Step 1: Send Order Details Interactive Message

To send an `order_details` message, businesses must assemble an interactive object of type `order_details` with the following components:

Object	Description
`type` object	**Required.** Must be “order_details”.
`header` object	**Optional.** Header content displayed on top of a message. If a header is not provided, the API uses an image of the first available product as the header
`body` object	**Required.** An object with the body of the message. The object contains the following field: `text` string **Required** if `body` is present. The content of the message. Emojis and markdown are supported. Maximum length is 1024 characters
`footer` object	**Optional.** An object with the footer of the message. The object contains the following fields: `text` string **Required** if `footer` is present. The footer content. Emojis, markdown, and links are supported. Maximum length is 60 characters

Object	Description
`action` object	**Required.** An action object you want the user to perform after reading the message. This action object contains the following fields: `name` string **Required**. Must be “review_and_pay” `parameters` object See Parameters Object for information

Parameters Object

Object	Description
`reference_id` string	**Required.** Unique identifier for the order or invoice provided by the business. It is case sensitive and cannot be an empty string and can only contain English letters, numbers, underscores, dashes, or dots, and should not exceed 35 characters. The reference_id must be unique for each order_details message for a given business. If there is a need to send multiple order_details messages for the same order, it is recommended to include a sequence number in the reference_id (for example, “BM345A-12”) to ensure reference_id uniqueness.
`type` object	**Required.** The type of goods being paid for in this order. Current supported options are `digital-goods` and `physical-goods`.
`beneficiaries` array	**Required for shipped physical-goods.** An array of beneficiaries for this order. A beneficiary is an intended recipient for shipping the physical goods in the order. It contains the following fields: Note: Beneficiary information isn’t shown to users but is needed for legal and compliance reasons. `name` string **Required.** Name of the individual or business receiving the physical goods. Cannot exceed 200 characters `address_line1` string **Required.** Shipping address (Door/Tower Number, Street Name etc.). Cannot exceed 100 characters `address_line2` string **Optional.** Shipping address (Landmark, Area, etc.). Cannot exceed 100 characters `city` string **Required.** Name of the city. `state` string **Required.** Name of the state. `country` string **Required.** Must be “India”. `postal_code` string **Required.** 6-digit zipcode of shipping address.

Object	Description
`currency`	**Required.** The currency for this order. Currently the only supported value is `INR`.
`total_amount` object	**Required.** The `total_amount` object contains the following fields: `offset` integer **Required.** Must be `100` for `INR`. `value` integer **Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234. `total_amount.value` must be equal to `order.subtotal.value` + `order.tax.value` + `order.shipping.value` - `order.discount.value`. UPI transactions are limited to ₹5,00,000. For higher amounts, set `enabled_payment_options` to `["web"]`. See Restrict Available Payment Options.
`payment_settings` object	**Required.** See Payment Settings object for more information.
`order` object	**Required.** See order object for more information.

Payment settings object

Object	Description
`type` string	**Required.** Must be set to **“payment_gateway”**

Object	Description
`payment_gateway` object	**Required.** An object that describes payment account information: `type` string **Required.** Unique identifier for an item in the order. You must set this to **“billdesk”** or **“razorpay”** or **“payu”** or **zaakpay**, if you have linked your BillDesk or Razorpay or PayU or Zaakpay payment gateway to accept payments `configuration_name` string **Required.** The name of the pre-configured payment configuration to use for this order and must not exceed 60 characters. This value must match with a payment configuration set up on the WhatsApp Business Manager. When `configuration_name` is invalid, the customer will be unable to pay for their order. We strongly advise businesses to conduct extensive testing of this setup during the integration phase. `billdesk/razorpay/payu/zaakpay` object **Optional.** For merchants/partners that want to use additional_info1/7(for BillDesk), notes and receipt(for Razorpay) and UDF fields(for PayU) and extra1/2(for Zaakpay), they can now pass these values in Order Details message and we would use these to create transaction/order at respective PGs. Please refer Payment Gateway specific UDF object for more information.

BillDesk, RazorPay, PayU and Zaakpay fields

We now have support for partners and merchants to pass `notes`, `receipt` and `udf` fields in Order Details message and receive this data back in payment signals. Here we will take a look at merchants can pass additional_info for BillDesk, notes and receipt fields for Razorpay, udf for PayU, extra for Zaakpay PGs.

Object	Description
`notes` object	**Optional.** Only supported for Razorpay payment gateway The object can be key value pairs with maximum 15 keys and each value limits to 256 characters.
`receipt` String	**Optional.** Only supported for Razorpay payment gateway Receipt number that corresponds to this order, set for your internal reference. Maximum length of 40 characters supported with minimum length greater than 0 characters.
`udf1-4` String	**Optional.** Only supported for PayU payment gateway User-defined fields (udf) are used to store any information corresponding to a particular order. Each UDF field has a maximum character limit of 255.

Object	Description
`extra1-2` String	**Optional.** Only supported for Zaakpay payment gateway User-defined fields (extra) are used to store any information corresponding to a particular order. Each extra field has a maximum character limit of 180.
`additional_info1-7` String	**Optional.** Only supported for BillDesk payment gateway User-defined fields (extra) are used to store any information corresponding to a particular order. Each extra field has a maximum character limit of 120.

Order object

Object	Description
`status` string	**Required.** Only supported value in the `order_details` message is `pending`. In an `order_status` message, `status` can be: `pending`, `captured`, or `failed`.
`type` string	**Optional.** Only supported value is `quick_pay`. When this field is passed in we hide the “Review and Pay” button and only show the “Pay Now” button in the order details bubble.

Object	Description
`items` object	**Required.** An object with the list of items for this order, containing the following fields: `retailer_id` string **Optional.** Content ID for an item in the order from your catalog. `name` string **Required.** The item's name to be displayed to the user. Cannot exceed 60 characters `image` object **Optional.** Custom image for the item to be displayed to the user. See item image object for information Using this image field will limit the items array to a maximum of 10 items and this cannot be used with `retailer_id` or `catalog_id`. `amount` amount object with value and offset -- refer total amount field above **Required.** The price per item `sale_amount` amount object **Optional.** The discounted price per item. This should be less than the original amount. If included, this field is used to calculate the subtotal amount. `quantity` integer **Required.** The number of items in this order, this field cannot be decimal has to be integer. `country_of_origin` string **Required** if `catalog_id` is not present. The country of origin of the product `importer_name` string **Required** if `catalog_id` is not present. Name of the importer company `importer_adress` string **Required** if `catalog_id` is not present. Address of importer company
`subtotal` object	**Required.** The value **must be equal** to sum of `order.amount.value` * `order.amount.quantity`. Refer to `total_amount` description for explanation of `offset` and `value` fields The following fields are part of the `subtotal` object: `offset` integer **Required.** Must be `100` for `INR` `value` integer **Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234

Object	Description
`tax` object	**Required.** The tax information for this order which contains the following fields: `offset` integer **Required.** Must be `100` for `INR` `value` integer **Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234 `description` string **Optional.** Max character limit is 60 characters
`shipping` object	**Optional.** The shipping cost of the order. The object contains the following fields: `offset` integer **Required.** Must be `100` for `INR` `value` integer **Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234 `description` string **Optional.** Max character limit is 60 characters
`discount` object	**Optional.** The discount for the order. The object contains the following fields: `offset` integer **Required.** Must be `100` for `INR` `value` integer **Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234 `description` string **Optional.** Max character limit is 60 characters `discount_program_name` string **Optional.** Text used for defining incentivised orders. If order is incentivised, the merchant needs to define this information. Max character limit is 60 characters
`catalog_id` object	**Optional.** Unique identifier of the Facebook catalog being used by the business. If you do not provide this field, you must provide the following fields inside the items object: `country_of_origin`, `importer_name`, and `importer_address`
`expiration` object	**Optional.** Expiration for that order. Business must define the following fields inside this object: `timestamp` string – UTC timestamp in seconds of time when order should expire. Minimum threshold is 300 seconds `description` string – Text explanation for expiration. Max character limit is 120 characters

Item Image Object

Object	Description
`link` string	**Required.** A link to the image that will be shown to the user. Must be an `image/jpeg` or `image/png` and 8-bit, RGB or RGBA. Follows same requirements as image in media

The `parameters` value is a stringified JSON object.
By the end, the interactive object should look something like this for a BillDesk catalog-based integration:
The `parameters` value is a stringified JSON object.
By the end, the interactive object should look something like this for a RazorPay catalog-based integration:
The `parameters` value is a stringified JSON object.
For a PayU non-catalog based integration i.e. when catalog-id is not present, an example payload looks as follows:
For a Zaakpay non-catalog based integration i.e. when catalog-id is not present, an example payload looks as follows:

Step 2: Add Common Message Parameters

Once the interactive object is complete, append the other parameters that make a message: `recipient_type`, `to`, and `type`. Remember to set the `type` to `interactive`.

These are parameters common to all message types.

Step 3:Make a POST Call to Messages Endpoint

Make a POST call to the `/[PHONE_NUMBER_ID]/messages` endpoint with the `JSON` object you have assembled. If your message is sent successfully, you get the following response:

For all errors that can be returned and guidance on how to handle them, see WhatsApp Cloud API, Error Codes.

Product Experience

The customer receives an `order_details` message similar to the one below (left). When they click on “Review and Pay”, it opens up the order details screen as shown below (middle). Customer can then pay for their order using “Continue” button that opens up a bottom sheet with the payment options (right).
[image removed - too large for import]
[image removed - too large for import]
[image removed - too large for import]

Step 4: Receive Webhook about Transaction Status

Businesses receive updates via messages webhooks when the status of the user-initiated transaction changes in a status of type “payment”. It contains the following fields:

Object	Description
`id` string	**Required.** Webhook ID for the notification.
`recipient_id` string	**Required.** WhatsApp ID of the customer.
`type` string	**Required.** For payment status update webhooks, type is “payment”.
`status` string	**Required.** `captured`/`pending`: `captured` - when the payment is successfully completed, `pending` when the user attempted but yet to receive success transactions signal

Object	Description
`payment` object	**Required.** Contains the following field: `reference_id` string Unique reference ID for the order sent in `order_details` message. `amount` object Has value and offset fields corresponding to total amount that user has paid. `currency` string currency is always INR. `transaction` object Transaction attempt for this payment. Transaction object contains the following fields: `id` string **Required.** The alpha-numeric payment gateway order ID. `pg_transaction_id` string **Optional.** The alpha-numeric payment gateway payment ID. `type` string **Required.** The payment type for this transactions. Only, `billdesk` or `razorpay` or `payu` or `zaakpay` are supported. `status` string **Required.** The status of the transaction. Can be one of `pending` or `success` or `failed`. `created_timestamp` integer **Required.** Time when transaction was created in epoch seconds. `updated_timestamp` integer **Required.** Time when transaction was last updated in epoch seconds. `method` object (**Optional.** the payment method information might not be available for failed payments) `type` string **Required.** The describes the type of payment method used by consumer to pay for the order. Can be one of `upi` or `card` or `wallet` or `netbanking`. `error` object (**Optional.** the payment error details might not be available for all payments attempts)`code` string **Required.** The describes the payment failure reason that is generated by payment gateway and Meta returns this to partners. `reason` string **Required.** The describes the payment failure reason in plain text that is generated by payment gateway and Meta returns this to partners. `additional_info1-7` string **Optional.** Only sent for billdesk payment gateway when the value is sent in order details message. Each of the keys additional_info1-4 has string values in them. `notes` object **Optional.** Only sent for razorpay payment gateway when the value is sent in order details message. This contains key-value pair as passed in the Order Details message. `receipt` string **Optional.** Only sent for razorpay payment gateway when the value is sent in order details message. `udf1-4` string **Optional.** Only sent for payu payment gateway when the value is sent in order details message. Each of the keys udf1-4 has string values in them. `extra1-2` string **Optional.** Only sent for zaakpay payment gateway when the value is sent in order details message. Each of the keys extra1-2 has string values in them. `refunds` array **Optional.** The list of refunds for this order. Each refund object

Object	Description
`timestamp` string	**Required.** Timestamp for the webhook.

Here is an example status webhook of type `payment`:

For more information about other statuses, see Messages Webhooks.

Step 5: Confirm Payment

After receiving the payment status webhook, or at any time, the business can look up the status of the payment for the order. To do that, businesses must make a GET call to the payments endpoint as shown here:

```
GET <PHONE_NUMBER_ID>/payments/<PAYMENT_CONFIGURATION>/<REFERENCE_ID>
```

where `payment_configuration` and `reference_id` are same as that sent in the `order_details` message.
Businesses should expect a response in the same HTTP session (not in a webhook notification) that contains the following fields:

Field	Description
`reference_id` string	**Required.** The ID sent by the business in the `order_details` message
`status` string	**Required.** Status of the payment for the order. Can be one of `pending` or `captured` Refer the table below for what these statuses mean.
`currency` string	**Required.** The currency for this payment. Currently the only supported value is `INR`.
`amount` object	**Required.** The amount for this payment. It contains the following fields: `offset` integer **Required.** Must be 100. `value` integer **Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234.

Field	Description
`transactions` array	**Optional.** The list of transactions for this payment. This field is only present when at least one payment attempt has been made. If the payment status is `pending` and no payment attempt has occurred, this field will not be returned. Each transaction object contains the following fields: `id` string **Required.** The alpha-numeric payment gateway order ID. `pg_transaction_id` string **Required.** The alpha-numeric payment gateway payment ID. `type` string **Required.** The payment type for this transactions. Only, `billdesk` or `razorpay` or `payu` or `zaakpay` are supported. `status` string **Required.** The status of the transaction. Can be one of `pending` or `success` or `failed`. At most one transaction can have a `success` status. `created_timestamp` integer **Required.** Time when transaction was created in epoch seconds. `updated_timestamp` integer **Required.** Time when transaction was last updated in epoch seconds. `method` object **Optional.** the payment method information might not be available for failed payments `type` string **Required.** The describes the type of payment method used by consumer to pay for the order. Can be one of `upi` or `card` or `wallet` or `netbanking`. `error` object **Optional.** the payment error details might not be available for all payments attempts `code` string **Required.** The describes the payment failure reason that is generated by payment gateway and Meta returns this to partners.`reason` string **Required.** The describes the payment failure reason in plain text that is generated by payment gateway and Meta returns this to partners. `refunds` array **Optional.** The list of refunds for this order. Each refund object contains the following fields: `id` string **Required.** The alpha-numeric ID of the refund. `amount` object **Required.** The total amount of the refund. `speed_processed` string **Required.** Speed by which refund was processed. Can be one of `instant` or `normal`.`status` string **Required.** The status of the refund. Can be one of `pending`, `success` or `failed`.`created_timestamp` integer **Required.** Time when refund was created in epoch seconds.`updated_timestamp` integer **Required.** Time when refund was last updated in epoch seconds.
`additional_info1-7` string	**Optional.** Supported for only BillDesk PG, this contains string values sent as part of Order Details message.

Field	Description
`receipt` string	**Optional.** Supported for only Razorpay PG, this contains the receipt-value sent as part of Order Details message.
`notes` object	**Optional.** Supported for only Razorpay PG, this contains the key-value pairs sent as part of Order Details message.
`udf1-4` string	**Optional.** Supported for only PayU PG, this contains string values sent as part of Order Details message.
`extra1-2` string	**Optional.** Supported for only Zaakpay PG, this contains string values sent as part of Order Details message.

Payment Status

Status	Description
`pending`	The order has been created but payment has not yet been captured. This status covers two scenarios: **No payment attempt yet:** The `transactions` array will not be present in the response.**Payment attempted but failed:** The `transactions` array will contain one or more entries with `status` set to `failed`.
`captured`	The payment was successfully captured. The `transactions` array will contain an entry with `status` set to `success`.

An example successful response looks like this:
Shown here is an example for a generic error:

Response by Payment Stage

The response payload varies depending on the payment stage. Below are examples for each stage.
No payment attempted — The user has not yet attempted payment. Only order-level fields are returned; the `transactions` array is not present.
Payment successful — The user completed payment and the payment gateway confirmed capture. The `transactions` array contains the successful transaction with payment method details.
Payment failed — The user attempted payment but it failed. The overall payment status remains `pending` (the order is still awaiting successful payment), but the `transactions` array contains the failed attempt with error details.

Step 6: Update Order Status

Businesses *must* send updates to their order using the `order_status` message instead of text messages since the latest status of an order displayed on the order details page is only based on `order_status` messages.

To notify the customer with updates to an order, you can send an `interactive` message of type `order_status` as shown below.
The following table describes the fields in the `order_status` interactive message:

Object	Description
`type` string	**Required.** Must be “order_status”
`body` object	**Required.** An object with the body of the message. The object contains the following field: `text` string **Required** if `body` is present. The content of the message. Emojis and markdown are supported. Maximum length is 1024 characters.
`footer` object	**Optional.** An object with the footer of the message. The object contains the following field: `text` string **Required** if `footer` is present. The footer content. Emojis, markdown, and links are supported. Maximum length is 60 characters.
`action` object	**Required.** An action object you want the user to perform after reading the message. This action object contains the following fields: `name` string **Required**. Must be “review_order”. `parameters` object See Parameters Object for information.

Parameters object

The `parameters` object contains the following fields:

Value	Description
`reference_id` string	**Required.** The ID sent by the business in the `order_details` message.
`order` object	**Required.** This object contains the following fields: `status` string **Required.** The new order `status`. Must be one of `processing`, `partially_shipped`, `shipped`, `completed`, `canceled`. `description` string **Optional.** Text for sharing status related information in `order_details`. Could be useful while sending cancellation. Max character limit is 120 characters.

`order_status` message introduces two new errors that are summarized below.

Error Code	Description
`2046` - Invalid status transition	The order status transition is not allowed.
`2047` - Cannot cancel order	Cannot cancel the order since the user has already paid for it.

Product Experience

Customers receive each `order_status` update as a separate message in their chat thread, that references their original `order_details` message as shown below (left). The order details page always displays the latest valid status communicated to the customer using the `order_status` message as shown below (right).
[image removed - too large for import]
[image removed - too large for import]

Supported Order Status and Transitions

Currently we support the following order status values:

Value	Description
`pending`	User has not successfully paid yet
`processing`	User payment authorized, merchant/partner is fulfilling the order, performing service, etc.
`partially-shipped`	A portion of the products in the order have been shipped by the merchant
`shipped`	All the products in the order have been shipped by the merchant
`completed`	The order is completed and no further action is expected from the user or the partner/merchant
`canceled`	The partner/merchant would like to cancel the `order_details` message for the order/invoice. The status update will fail if there is already a `successful` or `pending` payment for this `order_details` message

Order status transitions are restricted for consistency of consumer experience. Allowed status transitions are summarized below:
Initial status of an order is always `pending`, which is sent in `order_details` message.`canceled` and `completed` are terminal status and cannot be updated to any other status.`pending` can transition to any of the other statuses including `processing`, `shipped`, `partially-shipped`.`processing`, `shipped` and `partially-shipped` are equivalent statuses and can transition between one another or to one of the terminal statuses.
[image removed - too large for import]
Upon sending an `order_status` message with an invalid transition, you will receive an error webhook with the error code `2046` and message “New order status was not correctly transitioned.”

Canceling an Order

An order can be `canceled` by sending an `order_status` message with the status `canceled`. The customer cannot pay for an order that is canceled. The customer receives an `order_status` message and order details page is updated to show that the order is canceled and the “Continue” button removed. The *optional* text shown below “Order canceled” on the order details page can be specified using the `description` field in the `order_status` message.
An order can be canceled only if the user has not already paid for the order. If the user has paid and you send an `order_status` message with `canceled` status, you will receive an error webhook with error code `2047` and message “Could not change order status to ‘canceled’”.

Step 7: Reconcile Payments

WhatsApp does not support payment reconciliations. Businesses should use their payment gateway account to reconcile the payments using the `reference_id` provided in the `order_details` messages and the `id` of the transactions returned as part of the payment lookup query.

Merchant Preferred UPI Payment Method

Now merchants can specify up to one UPI Payment app to showup in checkout flow. Merchant preferred payment app will be shown on top of the list of available UPI apps in the “Choose payment method” screen. To enable this capability we require partners to specify the external app-id in the Order Details or order-invoice message.
Note: This feature is available on consumer apps on and above version: 2.24.21.0

Updates to Order Details Payload

List of supported apps:

UPI Application	Application ID to be passed in Order Details payload
Google Pay	gpay
PhonePe	phonepe
PayTm	paytm
BHIM	bhim
Amazon Pay	amazonpay
CRED	cred
Mobikwik	mobikwik

Restrict Available Payment Options

Merchants can specify which payment options to show in checkout flow between UPI and Web options. This will allow merchants to enable only UPI or credit card(any PG available option) to accept payments for invoices.
UPI transactions are limited to ₹5,00,000. For higher amounts, set `enabled_payment_options` to `["web"]` to use your payment gateway's web checkout. Payments with UPI enabled above this limit will fail.
Note: This feature is available on consumer apps on and above version: 2.24.22.4

Updates to Order Details Payload

List of payment options

Enabled Option	Experience in checkout flow
upi	Only UPI apps are show in checkout flow
web	Payment gateway webpage is loaded and merchant payment gateway account configured payment options will be shown in the checkout flow.

Some Payment Gateways allow customization of payment options that are shown in payment link or web based checkout flow. Please contact Payment Gateway to restricting payment options in payment link or web page.

Third Party Validation with Razorpay and PayU Payment Gateways

We now support TPV for RazorPay and PayU merchants, this allows merchants to specify the consumer accounts from which orders needs to be paid. Since, the consumer bank account information is sensitive, please work with Payment Gateways to procure public encryption key and pass the encryption information as part of Order details message.
To use this feature which is in alpha testing, please reach out to Meta payments team - whatsappindia-bizpayments-support@meta.com

Updates to Order Details Payload to support TPV for Razorpay merchants

The raw value before encryption should look something like the following:

Updates to Order Details Payload to support TPV for PayU merchants

The raw value before encryption should look like the following:
Note please closely work with Meta and Payment Gateway teams(RazorPay or PayU) to unlock this feature as we are still in alpha testing phase.

Security Considerations

Businesses should comply with local security and regulatory requirements in India. They should not rely solely on the status of the transaction provided in the webhook and must use payment lookup API to retrieve the statuses directly from WhatsApp. Businesses must always sanitize/validate the data in the API responses or webhooks to protect against SSRF attacks.

Checklist for Integrated Merchants

Ensure that `order_status` message is send to consumer informing them about updates to an order after receiving transaction updates for an order.
Ensure the merchant is verified and WABA contact is marked with a verified check.
Verify the WABA is mapped to appropriate merchant initiated messaging tier(1k, 10k and 100k per day)
Merchant should list the customer support information in the profile screen incase consumer wants to report any issues.
Migrate to “payment_settings” in place of “payment_type” and “payment_configuration”. This is the recommended way, and gives access to features likes “notes” and “udf” fields. For an example, view the payloads above.

Revision #6
Created 2026-04-02 10:15:30 UTC by New Admin
Updated 2026-04-06 17:52:54 UTC by New Admin