# Receive UPI Payments Through WhatsApp | Developer Documentation

# Receive UPI Payments Through WhatsApp

Updated: Nov 14, 2025

This is NOT the recommended UPI Intent integration. Migrate to dynamic VPA. With dynamic VPA, you can rotate VPAs on UPI intents without requiring any configuration changes on your end - simplifying integration and ongoing maintenance.
Your business can enable customers to pay for their orders using all the UPI Apps installed on their devices via WhatsApp. Businesses can send customers invoice(`order_details`) messages, then get notified about payment status updates via webhook notifications from Payment Gateway.

## Overview

Currently, customers browse business catalogs, add products to cart, and send orders in with our set of commerce messaging solutions, which includes Single Product Message, Multi Product Message, and Product Detail Page.
With the WhatsApp Payments API, businesses can send customers a bill so the customer can complete their order with all the UPI Apps.

## How It Works

The business must send an `order_details` message for the consumer to initiate payment. This type of 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 number is used throughout the flow to track the order. This `reference_id` is fetched from UPI intent link(**tr** value from the UPI payment intent) generated for a business order from Payment Gateway.
Once the message is sent, the business waits for a payment or transaction status updates directly from Payment Gateway. Upon receiving payment signal for an order, Business should relay this payment signal to consumer through interactive order status(`order_status`) message.
Updating user about the payment signal is important as this message updates the order details message and order details view for the consumer reflecting the order confirmation from merchant. This is shown with an example in subsequent sections.

# Purchase Flow in App

In the WhatsApp customer app, the purchase flow has the following steps:
Customer sends an order with selected products to the business or Business identifies the products that the customer has shown interest to purchase.
After receiving the order/identifying the product, if a merchant accepts payment methods other than UPI, such as credit cards and payment wallets, then merchant will send a message to the user to get their preferred payment method for the order.
Image
When consumers want to pay using UPI payment method, then merchants should retrieve the UPI payment intent by calling Payment Gateway. Merchant needs to use UPI intent to construct order details messages and send it to the consumer.
ImageImage
When the consumer taps the Pay now/continue button, they will be given the option to choose UPI payment Apps - WhatsApp or any other UPI payments apps. Consumers may choose any UPI option to pay for the order.
ImageImage
Consumer pays for the order and the payment method is saved for the future and automatically selected for the next payment transaction. Also, one quick note is the order details screen order-status will continue to show “Order pending” until merchant sends order status interactive message.
ImageImage
Once the payment is complete, the business receives a notification from Payment Gateway and the merchant needs to send order status updates to the consumer client notifying consumers about the progress to the order, this will update the order details message CTAs and order details screen - order status description.
ImageImageImage

# Before You Start

To receive UPI payments on WhatsApp merchants have to set up their VPA on WhatsApp Business Manager using the direct payment methods option.
Each configuration must have a unique name, merchant categorization code (mcc), purpose code, and VPA handles as shown below. A business can have multiple payment configurations, but for

each order merchant must specify a specific configuration to be used for payment. See `payment_configuration` field in `order_details` message.
`name` for each payment type must be unique. `name` will be used to reference the specific configurations for each payment type. If WA is unable to find a payment configuration name, the user will not be able to make the payment upon receiving the `order_details` message. `mcc` refers to a merchant categorization code for the items in the order. `upi_pc` refers to the purpose of the transaction. Some sample codes are below:

| Code | Title |
|---|---|
| 00 | Default |
| 01 | SEMI |
| 02 | AMC |
| 03 | Travel |
| 04 | Hospitality |
| 05 | Hospital |
| 06 | Telecom |
| 07 | Insurance |
| 08 | Education |
| 09 | Gifting |
| 10 | Others |

If merchant/partner is not sure of the MCC and purpose code, then they may contact Payment Gateway to get this information as Payment Gateway sets up these values based on the type of business while generating business VPAs.

# Manage Your Payment Methods (Beta Feature)

Self-service Payment Configuration allows you to add multiple payment configurations to your WhatsApp Manager profile. Each payment configuration will have its own UPI handle(VPA), MCC, and purpose code (for physical goods) so that merchants can accept payments for different categories with different UPI accounts. After setup, merchant can send invoice(`order_details`) messages to users with the corresponding payment configuration to collect payments.

## Pre-requisites

Manage Payment Methods feature is in Beta, so please contact Business Engineering team to allow merchants/partners access the page in WhatsApp Business Manager portal.

## Steps to link Payment Configuration for a Business Service Provider

You can create a payment configuration for a WhatsApp Business Account using the 'Payment configurations' page and under 'India' in WhatsApp Business Manager.
After linking your payment configuration, 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 is about to send order details message to consumer client.
The following sequence diagram demonstrates the typical integration flow for WA Payments API:
Image

## Step 1: Get UPI Intent from Payment Gateway

Once the consumer has expressed their interest to purchase an item using UPI payment method. Merchant needs to call payment gateway to create a UPI intent, the following is the sample UPI intent link:

```
upi://pay?pa=cfsukoonaa@yesbank&pn=Sukoon&tr=877376394&
  am=10.00&cu=INR&mode=00&purpose=00&mc=5399&tn=877376394
```

Merchant needs to parse the `tr` value from the above URI and use this as the reference id in invoice(`order_details`) interactive message.

## Step 2: Assemble the Interactive Object

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

| Object | Description |
|---|---|
| `type`<br>object | **Required.**<br>Must be "order_details". |

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

## Parameters Object

| Object | Description |
| --- | --- |
| `reference_id`<br>string | **Required.**<br>Unique identifier for the order or invoice provided by the business. This cannot be an empty string and can only contain English letters, numbers, underscores, dashes, or dots, and should not exceed 35 characters.<br>The `reference_id` must be unique for each `order_details` message for the same business. If the partner would like to send multiple order_details messages for the same order, invoice, etc. it is recommended to include a sequence number in the `reference_id` (for example, <order-or-invoice-id>-<sequence-number>) to ensure `reference_id` uniqueness. |
| `type`<br>object | **Required.**<br>The type of goods being paid for in this order. Current supported options are `digital-goods` and `physical-goods` |

| Object | Description |
| --- | --- |
| `beneficiaries`<br>array | **Required for shipped physical-goods.**<br>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:<br>Beneficiary information isn't shown to users but is needed for legal and compliance reasons.<br>`name` string<br>**Required.** Name of the individual or business receiving the physical goods. Cannot exceed 200 characters<br>`address_line1` string<br>**Required.** Shipping address (Door/Tower Number, Street Name etc.). Cannot exceed 100 characters<br>`address_line2` string<br>**Optional.** Shipping address (Landmark, Area, etc.). Cannot exceed 100 characters<br>`city` string<br>**Required.** Name of the city.<br>`state` string<br>**Required.** Name of the state.<br>`country` string<br>**Required.** Must be "India".<br>`postal_code` string<br>**Required.** 6-digit zipcode of shipping address. |
| `payment_type` | **Required.**<br>Must be "upi". |
| `payment_configuration` | **Required.**<br>The name of the pre-configured payment configuration to use for this order and must not exceed 60 characters. |
| `currency` | **Required.**<br>The currency for this order. Currently the only supported value is `INR`. |
| `total_amount`<br>object | **Required.**<br>The `total_amount` object contains the following fields:<br>`offset` integer<br>**Required.** Must be `100` for `INR`.<br>`value` integer<br>**Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234.<br>`total_amount.value` must be equal to `order.subtotal.value` + `order.tax.value` + `order.shipping.value` - `order.discount.value`. |
| `order`<br>object | **Required.**<br>See order object for more information. |

## Order Object

| Object | Description |
| --- | --- |

| | |
|---|---|
| `status`<br>string | **Required.**<br>Only supported value in the `order_details` message is `pending`.<br>In an `order_status` message, `status` can be: `pending`, `captured`, or `failed`. |
| `type`<br>string | **Optional.**<br>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. |
| `items`<br>object | **Required.**<br>An object with the list of items for this order, containing the following fields:<br>`retailer_id` string<br>**Optional.** Content ID for an item in the order from your catalog.<br>`name` string<br>**Required.** The item’s name to be displayed to the user. Cannot exceed 60 characters<br>`image` object<br>**Optional.** Custom image for the item to be displayed to the user. See item image object for information<br>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`.<br>`amount` amount object with value and offset -- refer total amount field above<br>**Required.** The price per item<br>`sale_amount` amount object<br>**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<br>`quantity` integer<br>**Required.** The number of items in this order, this field cannot be decimal has to be integer.<br>`country_of_origin` string<br>**Optional** if `catalog_id` is not present. The country of origin of the product<br>`importer_name` string<br>**Optional** if `catalog_id` is not present. Name of the importer company<br>`importer_adress` string<br>**Optional** if `catalog_id` is not present. Address of importer company |
| `subtotal`<br>object | **Required.**<br>The value **must be equal** to `order.amount.value` multiplied by `order.amount.quantity`.<br>Refer to `total_amount` description for explanation of `offset` and `value` fields<br>The following fields are part of the `subtotal` object:<br>`offset` integer<br>**Required.** Must be `100` for `INR`<br>`value` integer<br>**Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234 |

| | |
|---|---|
| `tax`<br>object | **Required.**<br>The tax information for this order which contains the following fields:<br>`offset` integer<br>**Required.** Must be `100` for `INR`<br>`value` integer<br>**Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234<br>`description` string<br>**Optional.** Max character limit is 60 characters |
| `shipping`<br>object | **Optional.**<br>The shipping cost of the order. The object contains the following fields:<br>`offset` integer<br>**Required.** Must be `100` for `INR`<br>`value` integer<br>**Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234<br>`description` string<br>**Optional.** Max character limit is 60 characters |
| `discount`<br>object | **Optional.**<br>The discount for the order. The object contains the following fields:<br>`offset` integer<br>**Required.** Must be `100` for `INR`<br>`value` integer<br>**Required.** Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234<br>`description` string<br>**Optional.** Max character limit is 60 characters<br>`discount_program_name` string<br>**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`<br>object | **Optional.**<br>Unique identifier of the Facebook catalog being used by the business. |
| `expiration`<br>object | **Optional.**<br>Expiration for that order. Business must define the following fields inside this object:<br>`timestamp` string – UTC timestamp in seconds of time when order should expire. Minimum threshold is 300 seconds<br>`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 |

By the end, the interactive object should look something like this for a catalog-based integration:
For a non-catalog based integration i.e. when catalog-id is not present, an example payload looks as follows:

## Step 3: 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 4: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.

## Step 5: Consumer Pays for the Order

Consumers can pay using WhatsApp payment method or using any UPI supported app that is installed on the device.

## Step 6: Get Notified About Transaction Status Updates from payment gateway

Businesses receive updates to the invoice via payment gateway webhooks, when the status of the user-initiated transaction changes. The unique identifier tr(reference-id passed in `order_details` message) can be used to map the transaction to the consumer invoice or interactive order details message.
Please refer to your PG's documentation for the exact payment signals. If you are not receiving webhooks from your PG, then work with them to enable it.

## Step 7: Update order status

Upon receiving transaction signals from payment gateway through webhook, the business must update the order status to keep the user up to date. Currently we support the following order status values:
Image

| Value | Description |
|---|---|
| `pending` | User has not successfully paid yet |

| Value | Description |
| --- | --- |
| `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 |

Typically businesses update the `order_status` using either the WhatsApp payment status change notifications or their own internal processes. To update `order_status`, the partner sends an `order_status` message to the user.
The following table describes the returned values:

| Value | Description |
| --- | --- |
| `reference_id` | The ID provided by the partner in the `order_details` message |
| `status` | The new order `status` |
| `description` | Optional text for sharing status related information in `order_details`. Could be useful while sending cancellation. Max character limit is 120 characters |

Merchant should always post this order-status message to consumer after receiving transaction updates for an order. As the order_details message and order details screen experience is tied to to the order status updates.

## Step 8: Reconcile Payments

Businesses should use their bank statements to reconcile the payments using the `reference_id` provided in the `order_details` messages.

# 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 - “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 |
| Amazon Pay | amazonpay |
| CRED | cred |
| Mobikwik | mobikwik |

# 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.

---

Revision #5
Created 2026-04-01 15:42:44 UTC by New Admin
Updated 2026-04-06 17:52:46 UTC by New Admin