

# Receive UPI Payments Through WhatsApp(Recommended) | Developer Documentation

## Receive UPI Payments Through WhatsApp(Recommended)

Updated: Mar 11, 2026

For businesses working with Billdesk or Zaakpay payment gateways, use the deeper integration with these PGs. Refer to [Payment Gateway Integration Guide](#)

For businesses working with Razorpay, PayU, or Cashfree payment gateways, use a faster integration path. Refer to [Enhanced Payment Links](#)

This is the recommended UPI Intent integration. Using the older payment configuration method?

[See previous docs](#) and plan to migrate to dynamic VPA.

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.

An `order_details` message contains the following fields that are worth noting:

`upi_intent_link` - Fields that will be supplied by your payment gateway and denote where a payment will be sent. `reference_id` - This is used to track the lifecycle of the order. Payment statuses are published against this ID. This could be order-id or transaction-id used to create the upi-intent at 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 users 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 the 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 the 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 the 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 the 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.

# Integration Steps

The steps outlined below assume that the business is about to send order details message to the 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=abc@psp&pn=ABC&tr=877376394&
am=10.00&cu=INR&mode=00&purpose=00&mc=5399&tn=877376394
```

Merchant/Partner could send the entire UPI intent as it is in the `upi_intent_link` type payload. These will be discussed in detail below.

## 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
<code>type</code> object	<b>Required.</b> Must be "order_details".
<code>header</code> object	<b>Optional.</b> 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
<code>body</code> object	<b>Required.</b> An object with the body of the message. The object contains the following field: <code>text</code> string <b>Required</b> if <code>body</code> is present. The content of the message. Emojis and markdown are supported. Maximum length is 1024 characters
<code>footer</code> object	<b>Optional.</b> An object with the footer of the message. The object contains the following fields: <code>text</code> string <b>Required</b> if <code>footer</code> is present. The footer content. Emojis, markdown, and links are supported. Maximum length is 60 characters

Object	Description
<p><code>action</code> object</p>	<p><b>Required.</b> An action object you want the user to perform after reading the message. This action object contains the following fields:</p> <p><code>name</code> string</p> <p><b>Required.</b> Must be “review_and_pay”</p> <p><code>parameters</code> object</p> <p>See <a href="#">Parameters Object</a> for information</p>

## Parameters Object

Object	Description
<p><code>reference_id</code> string</p>	<p><b>Required.</b> 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.</p> <p>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.</p>
<p><code>type</code> object</p>	<p><b>Required.</b> The type of goods being paid for in this order. Current supported options are <code>digital-goods</code> and <code>physical-goods</code></p>
<p><code>beneficiaries</code> array</p>	<p><b>Required for shipped physical-goods.</b> 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:</p> <p>Beneficiary information isn’t shown to users but is needed for legal and compliance reasons.</p> <p><code>name</code> string</p> <p><b>Required.</b> Name of the individual or business receiving the physical goods. Cannot exceed 200 characters</p> <p><code>address_line1</code> string</p> <p><b>Required.</b> Shipping address (Door/Tower Number, Street Name etc.). Cannot exceed 100 characters</p> <p><code>address_line2</code> string</p> <p><b>Optional.</b> Shipping address (Landmark, Area, etc.). Cannot exceed 100 characters</p> <p><code>city</code> string</p> <p><b>Required.</b> Name of the city.</p> <p><code>state</code> string</p> <p><b>Required.</b> Name of the state.</p> <p><code>country</code> string</p> <p><b>Required.</b> Must be “India”.</p> <p><code>postal_code</code> string</p> <p><b>Required.</b> 6-digit zip code of shipping address.</p>

Object	Description
<code>currency</code>	<b>Required.</b> The <code>currency</code> for this order. Currently the only supported value is <code>INR</code> .
<code>total_amount</code> object	<b>Required.</b> The <code>total_amount</code> object contains the following fields: <code>offset</code> integer <b>Required.</b> Must be <code>100</code> for <code>INR</code> . <code>value</code> integer <b>Required.</b> Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234. <code>total_amount.value</code> must be equal to <code>order.subtotal.value</code> + <code>order.tax.value</code> + <code>order.shipping.value</code> - <code>order.discount.value</code> .
<code>payment_settings</code> object	<b>Required.</b> See <a href="#">Payment Settings object</a> for more information.
<code>order</code> object	<b>Required.</b> See <a href="#">order object</a> for more information.

## Payment Settings Object

You can pass UPI intent as it is or parse the UPI intent parameters and pass them in a json structure. We support both the formats, so following are the two variants of payments settings objects:

Payment Settings Object for UPI intent link

Object	Description
<code>type</code> string	<b>Required.</b> Must be set to <b>"upi_intent_link"</b>
<code>upi_intent_link</code> object	<b>Required.</b> An object that describes payment account information: <code>link</code> string <b>Required.</b> The UPI intent that is generated from Payment gateway. The UPI intent only supports the following "&" separated attributes- pa, pn, mc, purpose and tr Example: <code>upi://pay?pa=merchant_vpa&amp;pn=Merchant_Name&amp;mc=merchant_category_code&amp;purpose=purpose_code&amp;tr=pg_generated_id</code>

## Order Object

Object	Description
--------	-------------

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

<p><code>tax</code> object</p>	<p><b>Required.</b> The tax information for this order which contains the following fields: <code>offset</code> integer <b>Required.</b> Must be <code>100</code> for <code>INR</code> <code>value</code> integer <b>Required.</b> Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234 <code>description</code> string <b>Optional.</b> Max character limit is 60 characters</p>
<p><code>shipping</code> object</p>	<p><b>Optional.</b> The shipping cost of the order. The object contains the following fields: <code>offset</code> integer <b>Required.</b> Must be <code>100</code> for <code>INR</code> <code>value</code> integer <b>Required.</b> Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234 <code>description</code> string <b>Optional.</b> Max character limit is 60 characters</p>
<p><code>discount</code> object</p>	<p><b>Optional.</b> The discount for the order. The object contains the following fields: <code>offset</code> integer <b>Required.</b> Must be <code>100</code> for <code>INR</code> <code>value</code> integer <b>Required.</b> Positive integer representing the amount value multiplied by offset. For example, ₹12.34 has value 1234 <code>description</code> string <b>Optional.</b> Max character limit is 60 characters <code>discount_program_name</code> string <b>Optional.</b> Text used for defining incentivised orders. If order is incentivised, the merchant needs to define this information. Max character limit is 60 characters</p>
<p><code>catalog_id</code> object</p>	<p><b>Optional.</b> 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: <code>country_of_origin</code>, <code>importer_name</code>, and <code>importer_address</code></p>
<p><code>expiration</code> object</p>	<p><b>Optional.</b> Expiration for that order. Business must define the following fields inside this object: <code>timestamp</code> string – UTC timestamp in seconds of time when order should expire. Minimum threshold is 300 seconds <code>description</code> string – Text explanation for expiration. Max character limit is 120 characters</p>

## Item Image Object

Object	Description
<code>link</code> string	<b>Required.</b> A link to the image that will be shown to the user. Must be an <code>image/jpeg</code> or <code>image/png</code> and 8-bit, RGB or RGBA. Follows same requirements as image in <a href="#">media</a>

By the end, the interactive object should look something like this for a merchant upi intent type catalog-based integration:

## 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:

### Errors

WhatsApp Payments Terms of Service Acceptance Pending

If you see the following error, accept the WhatsApp Payments terms of service using the link provided in the error message before trying again.

For all other 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 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 our PG integration guide for the exact payment signals. [Cashfree](#) and [CCAvenue](#)

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

The following table describes the returned values:

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

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

## 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 an `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 in case the consumer wants to report any issues.

---

Revision #5

Created 2026-04-01 14:49:54 UTC by New Admin

Updated 2026-04-06 17:51:20 UTC by New Admin