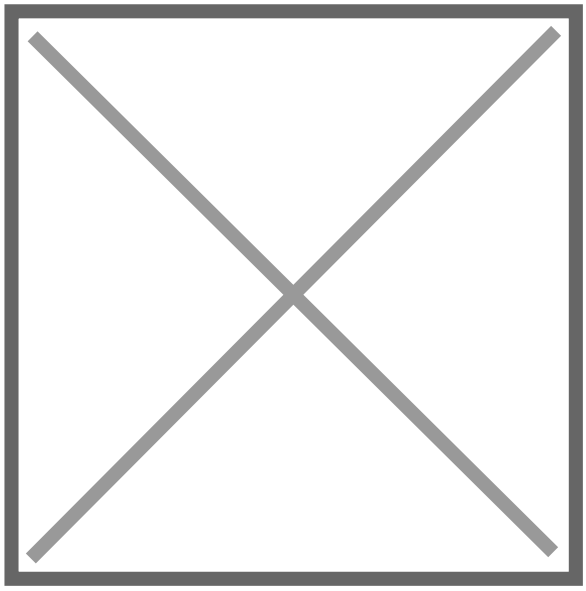
Checkout button templates | Developer Documentation

Checkout button templates

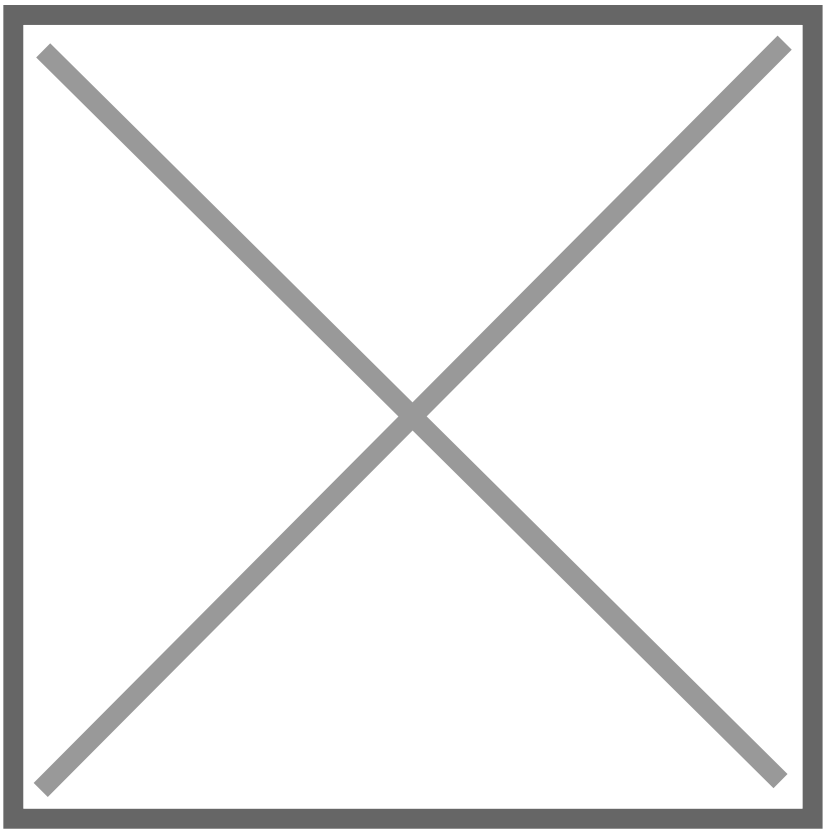
Updated: Dec 12, 2025

Checkout button templates are marketing templates that can showcase one or more products along with corresponding checkout buttons that WhatsApp users can use to make purchases without leaving the WhatsApp client.

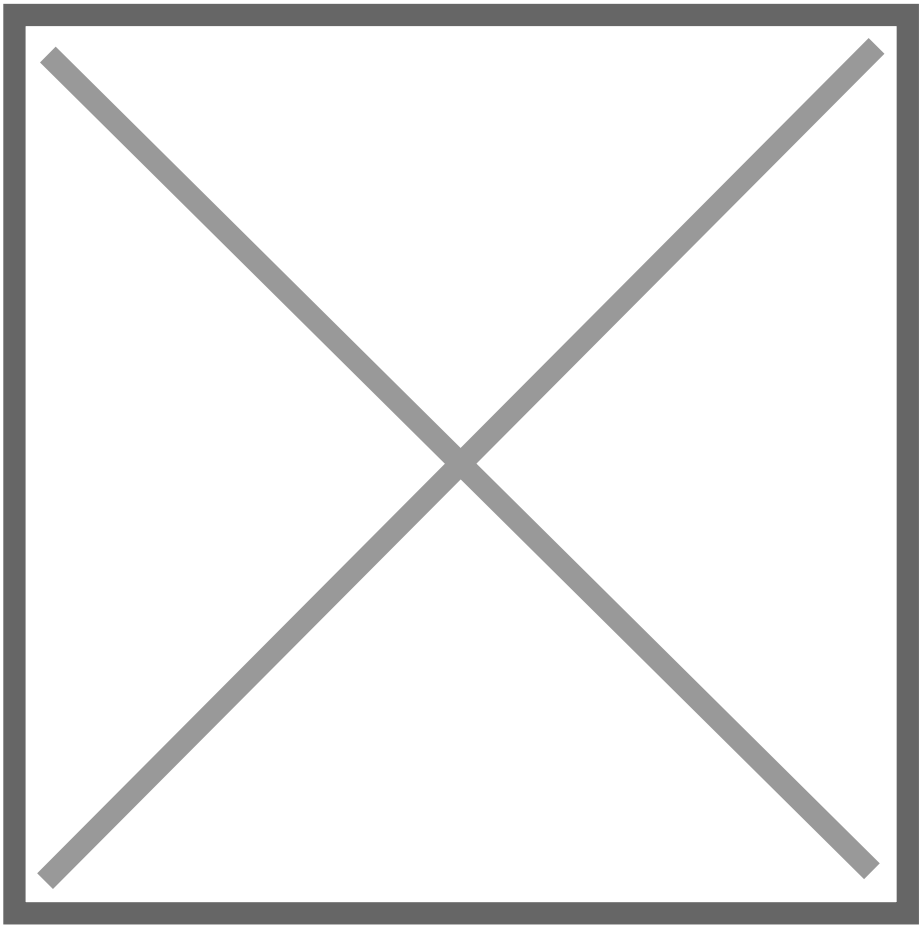


Single products

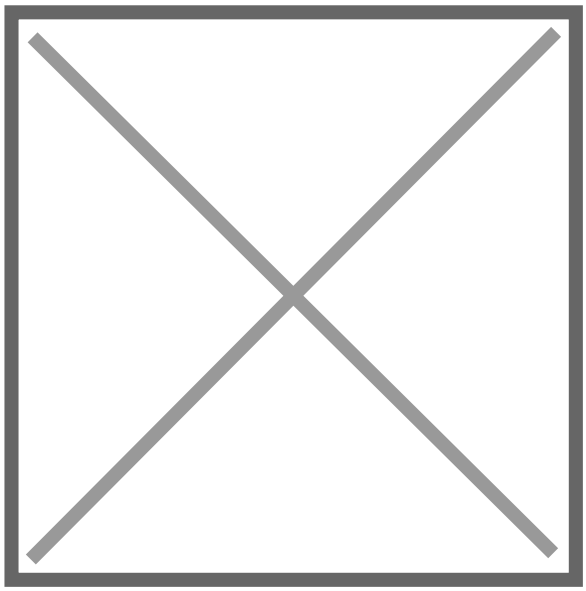
Checkout button templates can show a single product image or video header, along with message body text, message footer, a single checkout button, and up to 9 quick-reply buttons.



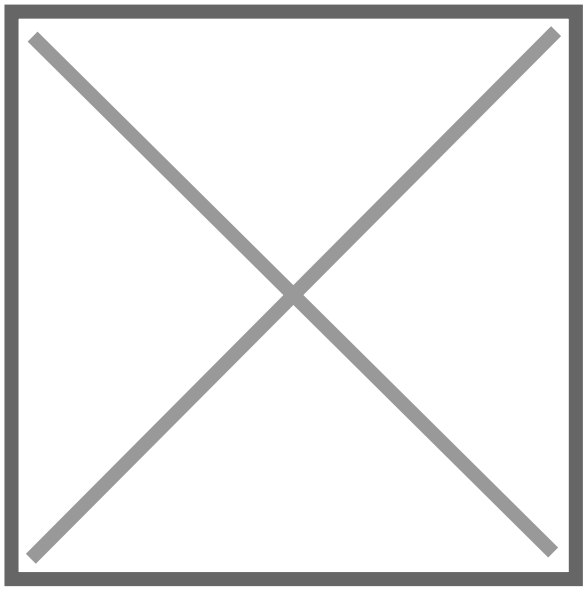
WhatsApp users who tap the button will see details of the order:



Users can proceed by selecting shipping information provided by you (if you know their information and supplied it in the send message payload)...



... or can add their own shipping information:



Enabling coupons, realtime inventory, and pricing updates

Enabling coupons, realtime inventory and pricing updates is currently in beta and only available to India businesses and WhatsApp users with an India country calling code. Please reach out to whatsappindia-bizpayments-support@meta.com to know more.

To enable coupons, realtime inventory and pricing updates, you can set up a checkout endpoint that can exchange data in real time to update the order on the WhatsApp client. It enables businesses to receive the shipping address and offers coupons based on the order and allows users to apply the coupon. It also enables businesses to validate inventory and serviceability on the order before the user completes the checkout.

Setting up the checkout endpoint consists of the following steps and it's the same method that WhatsApp Flows endpoint uses to share the data with WhatsApp clients.

Create a key pair and upload and sign the public key using the Cloud API.Setup the endpoint

Implement Payload Encryption/DecryptionLink the checkout endpoint with payment configuration
Implement checkout endpoint logic

Set up the endpoint

WhatsApp client makes a HTTPS request to exchange the data with the business endpoint. You should make sure the endpoint is configured probably to accept the request and link the endpoint url with the payment configuration:

```
https://business.com/checkout
```

Your server must be enabled to receive and process `POST` requests, use `HTTPS` and have a valid TLS/SSL certificate installed. This certificate does not have to be used in payload encryption/decryption.

Implement Encryption/Decryption

The body of each request contains the encrypted payload and has the following form:

Sample endpoint request syntax

```
{
  encrypted_flow_data: "<ENCRYPTED_FLOW_DATA>",
  encrypted_aes_key: "<ENCRYPTED_AES_KEY>",
  initial_vector: "<INITIAL_VECTOR>"
}
```

Parameter	Description
`encrypted_flow_data` string	**Required.** The encrypted request payload.
`encrypted_aes_key` string	**Required.** The encrypted 128-bit AES key.
`initial_vector` string	**Required.** The 128-bit initialization vector.

After processing the decrypted request, create a response and encrypt it before sending it back to the WhatsApp client. Encrypt the payload using the AES key received in the request and send it back as a Base64 string.

You can refer to examples of how to decrypt and encrypt.
If a request can not be decrypted, the endpoint should return HTTP 421 response status code (see Business Endpoint Error Codes for more details).

Link the checkout endpoint with payment configuration

The business should have payment gateway based payment configuration and reach out to whatsappindia-bizpayments-support@meta.com to enable the WhatsApp business account for checkout endpoint linking with with payment configuration.

Prior to linking the checkout endpoint, you should create a payment configuration and link with the payment gateway account. We advise you to use the linked payment configuration only with checkout button template integration.

You can achieve the endpoint linking with payment configuration by following Onboarding API's - Link data endpoint

Implement checkout endpoint logic

WhatsApp checkout endpoint integration inherits the 'data_exchange' similar to Flows and supports a set of subactions based on the user interaction and passes the relevant information in each of these actions to allow businesses to provide user specific coupons and enable businesses to update the pricing information accordingly.

Sub Action	Method	Description
`get_coupons`	Request	When users click on a savings offer CTA, WhatsApp passes order parameters excluding the payment settings. It also passes the `user phone number` as an input parameter. Refer get coupons request example to understand the order and input parameters
	Response	Checkout endpoint expected to pass the list of coupon information, such as code, id and description. Refer get coupons response example to understand the expected response.
`apply_coupon`	Request	When users select or enter a coupon, WhatsApp passes order parameters excluding the payment settings. It also passes the `user phone number` and information about the coupon to be applied as an input parameter. Refer apply coupon request example to understand the order and input parameters
	Response	Checkout endpoint expected to update the item and order pricing in order parameters and attach the coupon with the order Refer to apply coupon response example to understand the expected response.

Sub Action	Method	Description
`remove_coupon`	Request	When users try to remove an applied coupon, WhatsApp passes order parameters excluding the payment settings. It also passes the `user phone number` as an input parameter. Refer remove coupon request example to understand the expected response.
	Response	Checkout endpoint expected to update the item and order pricing in order parameters and remove the coupon attached with the order. Refer remove coupon response example to understand the expected response.
`apply_shipping`	Request	When users try to submit a shipping address, WhatsApp passes order parameters excluding the payment settings. It also passes the `user phone number` and shipping information as an input parameter. Refer to the apply shipping request example to understand the expected response.
	Response	Checkout endpoint expected to update the item and shipping pricing in order parameters. Refer to the apply shipping response example to understand the expected response.

We have created a checkout endpoint example? in Node.js that you can clone (remix) on Glitch to create your own endpoint and quickly prototype your checkout logic. Follow the instructions in the README.md? file to get started. Using Glitch is entirely optional. You can clone the example code from Glitch and run it in any environment you prefer.

Upon completing the above steps, when business sends the checkout template with the linked payment configuration, WhatsApp enables the coupons, realtime inventory and pricing updates and allows users to apply coupons and share shipping addresses.

When enabled the `Apply a savings offer` will appear in the order summary screen

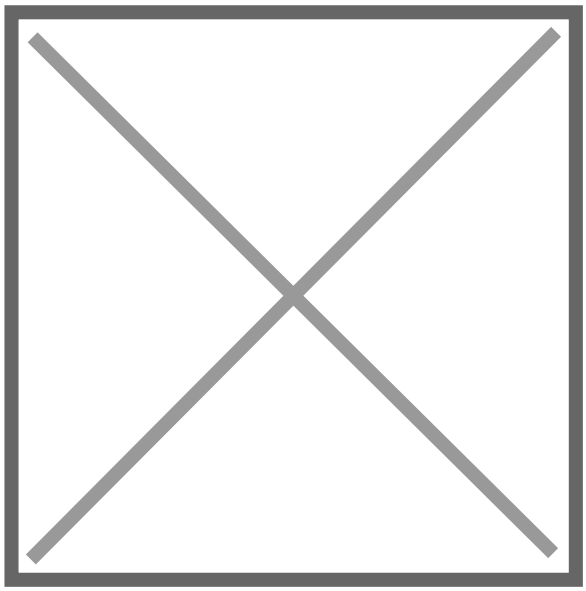
Image

User can click on `Apply a savings offer` to explore the coupons, at this point WhatsApp makes `get_coupons` request to fetch the list coupons based on the passed order and `user phone number` information.

Image

When the user tries to apply a coupon, WhatsApp makes `apply_coupon` and allow businesses to update the order or item pricing based on the selected coupon.

Image
Similar to coupons, user can share the shipping address by clicking on `Add shipping address` and select the addresses saved with the businesses or add new address. WhatsApp makes `apply_shipping` request when user tries to submit the address and allow businesses to check inventory and logistics based on the address provided.
ImageImage
Users can then continue to place the order using their preferred payment method set up in the WhatsApp client:



Once the order is processed, a payment webhook is triggered.

Multiple products

You can create a media card carousel template that showcases up to 10 products in a card carousel, each with their own checkout button. To do this, simply create a media card carousel template as you normally would, but replace one of the buttons with a checkout button, and make sure that it is the first button in the card.
Checkout buttons in media card carousel templates trigger the same order and payment flow as checkout buttons in templates that showcase a single product.

Checkout buttons

Each checkout button in a template must correspond to a single product. Checkout buttons, when creating a template, must have the following non-customizable syntax:
Note that this is simply a button definition. The actual details about the product that maps to this button are included when you send the template in a template message. For example:

If you are sending a media card carousel template (which can have two or more products), each checkout button must be defined in the template, and the item details that map to each button must be included when sending the template.

Send a checkout button template

Once your checkout button template or carousel template has been approved, you can send it in a template message.

Post body

This post body syntax is for a checkout button template. See Sending Media Card Carousel Templates for media card carousel template post body payload syntax.

Post body parameters

Placeholder	Description	Example Value
`<DISCOUNT_AMOUNT>` *Integer*	**Required if using a discount.** Discount amount, multiplied by discount.offset value. For example, to represent a discount of ?2, the value would be `200`. Discount amount applies to the order subtotal.	`15000`
`<DISCOUNT_DESCRIPTION>` *String*	**Optional.** Discount description. Maximum 60 characters.	`Additional 10% off`
`<EXPIRATION_TIMESTAMP>` *String*	**Required if using an order expiration.** UTC timestamp indicating when we should disable the **Buy now** button. The timestamp will be used to generate a text string that appears at the bottom of the **Order details** window. For example: *This order expires on September 30, 2024 at 12:00 PM.* WhatsApp users who view the message after this time will be unable to purchase the item using the checkout button. Values must represent a UTC time at least 300 seconds from when the send message request is sent to us.	`1726692927`
`<IMPORTER_ADDRESS_LINE_1>` *String*	**Required.** Importer address, line 1 (door, tower, number, street, etc.). Maximum 100 characters.	`One BKC`

Placeholder	Description	Example Value
`<IMPORTER_ADDRESS_LINE_2>` *String*	**Optional.** Importer address, line 2 (landmark, area, etc.). Maximum 100 characters.	`Bandra Kurla Complex`
`<IMPORTER_CITY>` *String*	**Required.** Importer city. Maximum 120 characters.	`Mumbai`
`<IMPORTER_NAME>` *String*	**Required.** Importer name. Maximum 200 characters.	`Lucky Shrub Imports and Exports`
`<IMPORTER_POSTAL_CODE>` *String*	**Required.** Importer 6-digit postal index number. Maximum 6 digits.	`400051`
`<IMPORTER_ZONE_CODE>` *String*	**Required.** Importer two-letter zone code.	`MH`
`<ITEM_COUNTRY_OF_ORIGIN>` *String*	**Required.** Item's country of origin. Maximum 100 characters.	`India`
`<ITEM_NAME>` *String*	**Required.** Item name. Maximum 60 characters.	`Blue Elf Aloe`
`<ITEM_PRICE>` *Integer*	**Required.** Individual item price (price per item), multiplied by amount.offset value. For example, to represent an item price of ?12.99, the value would be `1299`.	`200000`
`<ITEM_QUANTITY>` *Integer*	**Required.** Number of items in order, if order is placed. Maximum 100 integers.	`1`
`<MESSAGE_BODY_TEXT_VARIABLE>` *Object*	**Required if template message body text uses variables, otherwise omit.** Object describing a message variable. If the template uses multiple variables, you must define an object for each variable. Supports `text`, `currency`, and `date_time` types. See Messages Parameters. There is no maximum character limit on this value, but it does count against the message body text limit of 1024 characters.	
`<MESSAGE_HEADER_ASSET_ID>` *String*	**Required.**	`1558081531584829`

Placeholder	Description	Example Value
`<MESSAGE_HEADER_FORMAT>` *String*	**Required.** Indicates header type and a matching property name. Note that the `<MESSAGE_HEADER_FORMAT>` placeholder appears twice in the post body example above, as it serves as a placeholder for the type property's value and its matching property name. Value can be `image` or `video`.	`image`
`<PAYMENT_GATEWAY_CONFIGURATION_NAME>` *String*	**Required.** Configuration name of payment gateway you have configured on your WhatsApp Business Account.	`prod-razor-pay-config-05`
`<PAYMENT_GATEWAY_NAME>` *String*	**Required.** Name of payment gateway you have configured on your WhatsApp Business Account. Values can be: `razorpay` `payu` `zaakpay`	`razorpay`
`<PRODUCT_TYPE>` *String*	**Required.** Product type. Value can be `digital-goods` or `physical-goods`.	`digital-goods`
`<QUICK_REPLY_BUTTON_PAYLOAD>` *String*	**Optional.** Value to be included in messages webhooks (`messages.button.payload`) when the button is tapped.	`opt-out`
`<REFERENCE_ID>` *String*	**Required.** Your unique order or invoice reference ID. Case-sensitive. Cannot be empty. Will be preceded by a hash (#) symbol in the checkout flow. Value must be unique for each checkout button template message. If sending a carousel template, each checkout button must have a unique reference ID. If you need to send multiple messages for the same order/invoice, it is recommended to append a sequence number to the value (for example, -1). Values can only contain English letters, numbers, underscores, dashes, or dots. Maximum 35 characters.	`abc.123_xyz-1`

Placeholder	Description	Example Value
`<SALE_PRICE>` *Integer*	**Required if using a sale amount.** Sale price, multiplied by `sale.offset` value. For example, to represent a sale price of ?10, the value would be `1000`.	`150000`
`<SHIPPING_AMOUNT>` *Integer*	**Required.** Order shipping cost, multiplied by `shipping.offset` value. For example, to represent a shipping cost of ?.99, the value would be `99`.	`20000`
`<SHIPPING_INFO_ADDRESS>` *String*	**Required if you know the recipient's shipping information.** Product recipient's address. Maximum 512 characters.	`Bandra Kurla Complex`
`<SHIPPING_INFO_BUILDING_NAME>` *String*	**Optional.** Product recipient's building name. Maximum 128 characters.	`One BKC`
`<SHIPPING_INFO_CITY>` *String*	**Required if you know the recipient's shipping information.** Full name of product recipient's city. Maximum 100 characters.	`Mumbai`
`<SHIPPING_INFO_FLOOR_NUMBER>` *String*	**Optional.** Product recipient's floor number. Maximum 10 characters.	`2`
`<SHIPPING_INFO_HOUSE_NUMBER>` *String*	**Optional.** Product recipient's house number. Maximum 8 characters.	`12`
`<SHIPPING_INFO_INDIA_PIN>` *String*	**Required if you know the recipient's shipping information.** Product recipient's postal index number. Maximum 6 characters.	`400051`
`<SHIPPING_INFO_LANDMARK_AREA>` *String*	**Optional.** Product recipient's landmark area. Maximum 128 characters.	`Near BKC Circle`
`<SHIPPING_INFO_NAME>` *String*	**Required if you know the recipient's shipping information.** Product recipient's full name. Maximum 256 characters.	`Nidhi Tripathi`
`<SHIPPING_INFO_PHONE_NUMBER>` *String*	**Required if you know the recipient's shipping information.** Product recipient's WhatsApp phone number. Maximum 12 characters.	`919000090000`

Placeholder	Description	Example Value
`<SHIPPING_INFO_STATE>` *String*	**Required if you know the recipient's shipping information.** Full name of product recipient's state. Maximum 100 characters.	`Maharastra`
`<SHIPPING_INFO_TOWER_NUMBER>` *String*	**Optional.** Product recipient's tower number. Maximum 8 characters.	`2`
`<SUBTOTAL_AMOUNT>` *Integer*	**Required.** Order subtotal. Calculate by multiplying `<ITEM_PRICE>` by `<ITEM_QUANTITY>` by `subtotal.offset`. For example, if the template is for placing a single order containing 2 items priced at ?12.99, the value would be `2598`.	`150000`
`<TAX_AMOUNT>` *Integer*	**Required.** Tax amount, multiplied by `tax.offset` . For example, to represent a tax amount of ?5, the value would be `500` .	`10000`
`<TAX_DESCRIPTION>` *String*	**Optional.** Tax description. Maximum 60 characters.	`Sales tax`
`<TEMPLATE_LANGUAGE>` *String*	**Required.** Template language and locale code.	`en_US`
`<TEMPLATE_NAME>` *String*	**Required.** Template name. Maximum 512 characters.	`item_back_in_stock_v1`

`<TOTAL_AMOUNT>`
Integer
	Required.
	Total amount of order, multiplied by `total_amount.offset` value.
	For example, to represent a total amount of ?18, value be `1800`.
	Must be a sum of:
	`order.subtotal.value`
	`order.shipping.value` `order.tax.value`
	Minus:
	`order.discount.value`
		`165000`

`<WHATSAPP_USER_PHONE_NUMBER>` *String*	**Required.** WhatsApp user phone number.	`+16505551234`

Example request

The following sample request and responses are only supported with Enabling coupons, realtime inventory and pricing updates feature and it is currently in beta and only available to India businesses and WhatsApp users with an India country calling code. Please reach out to

whatsappindia-bizpayments-support@meta.com to know more.

Revision #6
Created 2026-04-01 14:57:24 UTC by New Admin
Updated 2026-04-06 19:03:12 UTC by New Admin