Kaleyra’s Wrapper API for Tech Provider Framework

Kaleyra’s Wrapper API

POST Method

Kaleyra's Wrapper Messaging API is a proxy or lightweight API. This API follows Meta's messaging payload format and passes requests directly to Meta for each new template created via Business Manager (BM). Using this API, Kaleyra provides messaging support only by receiving customer requests, sending them to Meta, and then passing Meta's responses directly to the customer.

1. Sign up for free

Before you get started, sign up for a Kaleyra.io account for free and Create an API Key. To view the API Key and the SID, see View API Key and SID.

2. Pre-requisites

Following are the pre-requisites for using Kalyera WhatsApp API’s

- A WhatsApp business number.
- An associated profile with the business number.
- A WhatsApp verified and approved profile.
- A working API key and SID key

For more details, read what WhatsApp says about creating a business account.

3. Send Wrapper Message

You can use this API to send a wrapper message through your WhatsApp account.

Base URL

```
<https://cloud-api.in.kaleyra.io/v1/><SID>
```

End Point

```
<API_DOMAIN>/v1/<sid>/wrapper/<PHONE_NUMBER_ID>/messages
```

Data

The same payload is defined in **Meta Cloud API Documentation**.

Response

Same response as defined in META cloud API documentation for <**Success Responses**> and <**Error Responses**>.

API request to send Wrapper Message

To send a Wrapper Message, use the **/wrapper/<PHONE_NUMBER_ID>/messages** endpoint.

Request Format

The following is the request format for the wrapper message:

```
curl --location
'https://cloud-api.in.kaleyra.io/v1/<SID>/wrapper/<PHONE_NUMBER_ID>/messages'\
--header 'api-key: <api_key>' \
--header 'Content-Type: <content_type>' \ --data
'{
      "messaging_product": "<messaging_product>", \
      "recipient_type": "<recipient_type>", \
      "to": "<to_number>", \
      "type": "<type>", \
      "text": {
        "preview_url": <preview_url>, \
        "body": "<body>"
        }
    }'
```

Sample Request

The following is the sample request for the wrapper message:

```
curl --location
'https://cloud-api.in.kaleyra.io/v1/HXIN1XXXXXXXX/wrapper/<PHONE_NUMBER_ID>/mes
sages'\
--header 'api-key: Axxxxxxxxxxxxxxxxxxxxxxxxxxxx3' \
--header 'Content-Type: application/json' \
--data '{
      "messaging_product": "whatsapp", \
      "recipient_type": "individual", \
 "to": "+9181XXXXXX81", \
 "type": "text", \
 "text": {
  "preview_url": false, \
  "body": "This is a wrapper test message."
  }
}'
```

Parameters and Data Types

Following is the list of parameters and data types supported:

Parameter	Data Type	Description	Example
`sid`	String	Account SID (Security Identifier).	HXIN178XXXXXXXIN
`PHONE_NUMBER_ID`	Numeric	For more information, see Messages.	12772361041857
`api-key`	String	API key generated from Kaleyra.io account.	Axxxxxxxxxxxxxxxxxxxxxxxxxxxx3
`content-type`	String	The format in which the data is sent.	application/json

`messaging_product`	string	Messaging service used for the request. For ending WA messages, use “whatsapp”	whatsapp

Mandatory?

Yes

Yes

Yes

Yes

Yes

Parameter	Data Type	Description	Example
`recipient_type`	string	Currently, you can only send messages to individuals. Set this as individual.	individual

`to`	string	WhatsApp ID or phone number of the customer you want to send a message to. See Phone Number Formats. If needed, On-Premises API users can get this number by calling the `contacts` endpoint.	+9181XXXXXX81
`type`	string	The type of message you want to send. If omitted, defaults to text.	text
`text`	object	A text object	
`preview_url`	boolean	Allows for URL previews in text messages, see the **Sending URLs in Text Messages**. This field is optional	Values: `false` (default), `true`.

Parameter	Data Type	Description	Example

Mandatory?

Optional

yes

optional

Required for text messages

Required if type=text.

Mandatory?

		if not include a URL in your message. Values: `false` (default), `true`.	
`body`	String	Required for text messages. The text of the message which may contain URLs that begin with http:// or https:// and formatting. See available formatting options **here**. If you include URLs in your text and want to include a preview box in text messages (`preview_url: true`), make sure the URL starts with `http://` or `https://` — `https://` URLs are preferred. You must include a hostname, since IP addresses will not be matched. Maximum length: 4096 characters.	This is a wrapper test message.

Required for text messages

Sample Success Response

The following success message appears with status 200 Accepted:

JSON

```
{
    "messaging_product": "whatsapp",
  "contacts": [
    {
      "input": "1650xxxxx55",
      "wa_id": "1650xxxxx55"
    }
  ],
  "messages": [
    {
        "id": "wamid.HBgLMxxxMDUwxxxxxjAVAgARGxxxxxDQTVCM0QxxxxxRTY3RTcA"
    }
  ]
}
```

Note: All responses will be in line with responses given by Meta apart from Unauthorised request responses.

Sample Failure Response

1. Unauthorized Error

The following unauthorized message appears with status 401 error:

JSON

```
{
    "code": "RBC201",
    "message": "Incorrect SID or API key.",
    "data": [],
    "error": {
        "body": "Incorrect SID or API key."
    }
}
```

2. Bad Request Error

The following bad request message appears with status 400 error:

JSON

```
{
    "error": {
        "code": 100,
        "error_subcode": 33,
        "fbtrace_id": "Awnxxxxxi_R3fxxxxxhiIL5",
        "message": "Unsupported post request. Object with ID 'messages' does
not exist, cannot be loaded due to missing permissions, or does not support this
operation. Please read the Graph API documentation at
https://developers.facebook.com/docs/graph-api",
        "type": "GraphMethodException"
    }
}
```

Notes:

- For more information on success response and failure response from Meta, see Success Response and Failure Response pages.
- All responses are aligned with those given by Meta apart from Unauthorised request responses.

Limitation and Assumptions:

- Webhook endpoint configuration will be done by the tech provider during the configuration phase either with the use of webhook override or changing webhook URL directly at the app level.
- The Tech Provider is responsible for any input level validations, throttling, and balance checks.

Error Codes

For information related to the WhatsApp error code, see WhatsApp Error Codes.