



# API Testing Documentation

**Fadhaa Al-Afkar** for Information Technology,  
Financial and Banking Technologies, Human Resources Development, and  
General Trade Ltd., Karrada, District 903, Street 38, House 7 – 31.

**M.** +964 780 002 0999  
**E.** ali@wayl.io

**A.** Baghdad, Karrada, Mahalla  
903, Z38, D7-31

## → Introduction

### Introduction

### Wayl makes it easy to get paid by connecting your business

Wayl is a payment facilitator that makes it easy for your customers to pay you for goods and services through many payment gateways.

### Wayl API

Wayl has developed its own RESTful API to make it easy for merchants who build their own websites to integrate with the Wayl payment page.



[ 01 ]



Fadhaa Al-Afkar for Information Technology,  
Financial and Banking Technologies, Human Resources  
Development, and General Trade Ltd., Karrada, District  
903, Street 38, House 7 – 31.

M. +964 780 002 0999  
E. ali@wayl.io  
A. Baghdad, Karrada, Mahalla  
909, Z38, D7-31

## → Start using Wayl API

### Start Using Wayl API

**Register and get your Merchant Token to start using Wayl API for seamless payments.**

To use Wayl API for your app or a website and benefit from its various payment options, you first have to get a Merchant Token. If you don't have one already, please head to <https://wayl.io>, register your account, and contact us to get your Merchant Token.

After you get your **Merchant Token**, you can either use it to directly interact with the Wayl API from your server, or you can use our API platform to test the API. For Demo purposes, we will be using the API Platform <https://api.thewayl.com>

### 1. Using Wayl API

#### Verifying your Merchant Token (see if it works)

GET https://api.thewayl.com/api/v1/verify-auth-key



To verify that your merchant token is working properly, go to <https://api.thewayl.com>, then from the navigation menu press Authentication – Verify Authentication Key. This page will appear:

The screenshot shows the Wayl API documentation interface. On the left, there is a navigation sidebar with a search bar at the top. Below the search bar is a list of categories: Introduction, Channels, Links, Refunds, Authentication (which is expanded), Models, and Verify Authentication Key (which is selected and highlighted in blue). To the right of the sidebar, the main content area has a title "Verify Authentication Key". Under this title, there is a "Responses" section showing a 200 status code with the message "Authentication key is valid" and an "application/json" content type. To the right of the responses, there is a "Required Parameters" section and a "Response Structure" section. The "Response Structure" section shows a JSON schema: { "data": {}, "message": "string" }. At the bottom right of the main content area, there is a "Testing Area" with a "Test Request" button. Above the testing area, there is a curl command: "curl https://api.thewayl.com/api/v1/verify-auth-key". The entire interface is styled with a dark background and red highlights for the selected navigation items.

[ 02 ]

## → Using Wayl API

Now in **Testing Area**, Press **Test Request**, and the API testing screen will appear:

The screenshot shows a POST request to `https://api.thewayl.com/api/v1/verify-auth-key`. The request body contains the following JSON:

```
1 {  
2   "data": {},  
3   "message": "Authentication key is valid",  
4   "success": true  
5 }
```

Annotations with red numbers:

- 1: Points to the `merchant` dropdown in the Headers section.
- 2: Points to the `X-WAYL-AUTHENTICATION` header value.
- 3: Points to the **Send** button.
- 4: Points to the JSON response body.

1. Select **Authentication** as **Merchant**
2. Enter your **Merchant Token** in **Value** of as **X-WAYL-AUTHENTICATION**
3. Press **Send**
4. You will get a response. If your merchant token is **valid**, the message will be:  
**"Authentication key is valid"**

Note: In each session, your token will be saved in the testing platform, so you can perform other requests without the need to reenter the token

## 2. Channels

### Getting The Available Payment Methods

GET https://api.thewayl.com/api/v1/channels



Sometimes, you want to display for your customer the available payment options on your app or a website.

#### Press Channels – Retrieve Channels – Test Request

Then send a GET request, you will receive a response as following:

The screenshot shows the Postman interface with a successful API call. The URL is `https://api.thewayl.com/api/v1/channels`. The response status is `7.5s 200 OK`. The response body is a JSON object:

```
1 < {  
2 <   "data": [  
3 <     {  
4 <       "name": "البطاقات",  
5 <       "icon": "https://icon.com/cards.png",  
6 <       "id": "Cards"  
7 <     },  
8 <     {  
9 <       "name": "البنك الأول العراقي",  
10 <      "icon": "https://icon.com/fib.png",  
11 <      "id": "FIB"  
12 <     },  
13 <     {  
14 <       "name": "خدمات كي",  
15 <       "icon": "https://icon.com/qi-service.png",  
16 <       "id": "QiCard"  
17 <     },  
18 <     {  
19 <       "name": "سويفت",  
20 <       "icon": "https://icon.com/switch.png",  
21 <       "id": "Switch"  
22 <     },  
23 <   }  
24 <
```

# → Links

## 3. Links

### What You Will Need The Most

Links are the order you will create on the Wayl platform and send it link to the customer to complete the payment

#### 3.1. Create a Link

POST https://api.thewayl.com/api/v1/links



Creating A link (in addition to **Merchant Token**) requires some **req.body** parameters that you have to send to our server. The required parameter is explained in the image below:

The screenshot shows the Wayl API documentation. On the left, there's a sidebar with various endpoints. One endpoint, 'Create a Link', is highlighted with a red arrow pointing to it from the sidebar. The main content area shows the 'Create a Link' endpoint details. It includes a 'Body' section with required fields: 'referenceId', 'total', 'currency', 'lineItem', 'label', 'amount', 'type', 'webhookUrl', 'webhookSecret', and 'redirectionUrl'. Each field has a description and examples. To the right of the body, there's a 'curl' command example and a 'Test Request' button.

Press Test Request and send a POST request with the required Body.

#### Note:

- **referenceId**: should be unique in your database;
- **total**: should equal the summation of **lineItem: increase/decrease**;
- **webhookUrl**: a service endpoint that will receive the order and payment;
- **details**: when the payment is completed;
- **redirectionUrl**: a url that the customer will be redirected to when the payment is completed.

[ 05 ]

## → Links

After sending the request, you will get a response :

The screenshot shows a POST request to `https://api.thewayl.com/api/v1/links`. The request body is a JSON object representing a payment link. The response body is a JSON object containing details about the created payment link, including its URL, webhook URL, and redirection URL.

**Request Body:**

```
1 {  
2   "referenceId": "id942",  
3   "total": 4000,  
4   "currency": "IQD",  
5   "lineItem": [  
6     {  
7       "label": "A Flower",  
8       "amount": 5000,  
9       "type": "increase"  
10    },  
11    {  
12      "label": "Discount",  
13      "amount": 1000,  
14      "type": "decrease"  
15    }  
16  ],  
17  "webhookUrl": "https://webhook.site/#/view/897e5f20-e1ed-4629-99cb-7e176abf276",  
18  "webhookSecret": "1234567890",  
19  "redirectionUrl": "https://www.mysite.com/paymentCompleted"  
}
```

**Response Body:**

```
1 {  
2   "data": {  
3     "referenceId": "id942",  
4     "id": "cmay47zv3005hmw08l7wrtxjp",  
5     "total": "4000",  
6     "currency": "IQD",  
7     "code": "DE1082DD",  
8     "paymentMethod": null,  
9     "type": "FullPaymentLink",  
10    "status": "FullPaymentLink",  
11    "completedAt": null,  
12    "createdAt": "2025-05-21T15:46:14.991Z",  
13    "updatedAt": "2025-05-21T15:46:14.991Z",  
14    "url": "https://link.thewayl.com/pay?id=cmay47zv3005hmw08l7wrtxjp",  
15    "webhookUrl": "https://webhook.site/#/view/897e5f20-e1ed-4629-99cb-7e176abf276",  
16    "redirectionUrl": "https://www.mysite.com/paymentCompleted"  
17  },  
18  "message": "Done",  
19  "success": true  
}
```

At this point, you have created a payment link that you should redirect your customer to.

["url": "https://link.thewayl.com/pay?id=cmay47zv3005hmw08l7wrtxjp",](https://link.thewayl.com/pay?id=cmay47zv3005hmw08l7wrtxjp)

After the customer completes the payment, he will be redirected to the earlier specified `redirectionUrl`, and the order and payment details will be sent to the earlier specified `webhookUrl`.

Note:

- `referenceId` is the Id in your database and should be unique;
- `id` is the order id in Wayl system;
- `code` is the order Code in Wayl System.

## → Links

Below is what your customer will see when visiting your created link :  
"url": "https://link.thewayl.com/pay?id=cmay47zv3005hmv08l7wrtxjp"



[ 07 ]

## Webhook

### 3.1.1. Link Webhook Url

Understanding webhooks is an important part of Wayl API integration. A webhook is a request that will be sent when a payment is completed to a service endpoint on your server.

This request will contain data about the order, payment method used, customer info, and address, and most importantly, a header signature to verify that this request has been sent from Wayl servers.

So, to verify that your customer has truly paid for your product, you have to create a service endpoint on your server that will receive the webhook request.

Note: for testing purposes, you can use <https://webhook.site>, which will give you an endpoint that will receive requests.

Below is an example of a received webhook request after a customer completed a payment:

**Request Details & Headers**

POST	https://webhook.site/53a2472c-6a93-4e51-9601-b5	7	host	webhook.site
Host	169.224.11.183	Whois Shodan Netify Censys VirusTotal	accept-encoding	gzip, compress, deflate, br
Date	05/21/2025 5:05:56 PM (2 hours ago)		content-length	482
Size	482 bytes		user-agent	axios/1.7.9
Time	0.000 sec		x-wayl-signature-256	1b69bc002ac6cdf795757b78d8b754f2651cbbc616f6c92fffdca511e89e692
ID	eef55d54-aaee-44f8-96da-f6d002c76650		content-type	text/plain
Note	<a href="#">Add Note</a>		accept	application/json, text/plain, */*

**Query strings**

None

**Form values**

None

**Webhook Signarure**

**Request Content**

**Raw Content**

```
{
  "verb": "POST",
  "event": "order.created",
  "referenceId": "dassasasuaassasmmy-rze12faserasasenceas-id",
  "paymentMethod": "Card",
  "paymentStatus": "Completed",
  "paymentProcessor": "Wayl",
  "total": 1000,
  "commission": 0,
  "code": "I94F590I",
  "customer": {
    "id": "cm8kktqmz0000g00btwo4ill",
    "name": "John Doe",
    "phone": "+964 780 002 0999",
    "city": "iraq_al_basrah",
    "country": "IQ",
    "address": "Baghdad"
  },
  "items": [
    {
      "type": "increase",
      "label": "Basket Value",
      "amount": 1000
    }
  ],
  "id": "cmay0lfl2000ng0194x5r801o"
}
```

**Webhook Data**

## Webhook Signature

The webhook signature is a very important part of the Wayl API that you don't want to ignore; it verifies that the webhook request has been truly sent from Wayl servers after a customer completes a Payment.

### Verifying the Signature on Your Server

On your webhook endpoint, you have to implement a signature verification function like this one :

Using Node-JS // Wayl Signature Verification

```
import crypto from "crypto";
function verifyWebhookSignature(data, signature, secret) {
  //data from webhook req.body
  //signature from req.headers "x-wayl-signature-256"
  //secret that was previously sent when creating the link, should be locally
  //stored in your code or database
  //generate signature from data and secret
  const calculatedSignature = crypto.createHmac("sha256",
    secret).update(data).digest("hex");
  // Convert the signatures to buffers for security
  const signatureBuffer = Buffer.from(signature, "hex");
  const calculatedSignatureBuffer = Buffer.from(calculatedSignature, "hex");
  // Compare the created signature with the signature from the request
  if (signatureBuffer.length !== calculatedSignatureBuffer.length) {
    return false;
  }
  return crypto.timingSafeEqual(signatureBuffer, calculatedSignatureBuffer);
  // true if the signature is valid, false otherwise
}
verifyWebhookSignature(data, signature, secret)
```

## → Links

Different Langauge  
and Currency Display

**Different Langauge and Currency Display**

**Link + &lang=ku&currency=iqd**

**Link + &lang=en&currency=usd**

**Link + &lang=ar&currency=iqd**

**3.2 Retrieve Links**

**3.3 Retrieve a specific Link**

**3.4 Invalidate a specific Link**