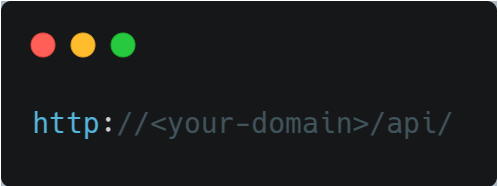
# **WhatsApp Integration API Documentation**

This documentation outlines the available API endpoints for sending and receiving WhatsApp messages using the Django application. The API follows RESTful principles and includes basic async message processing.

## **Base URL**



Replace <your-domain> with your actual domain or localhost URL.

## **Authentication**

No authentication is required for these API endpoints in the current version.

## **Endpoints**

### **1. Send Message**

* **URL**: /send-message/
* **Method**: POST
* **Description**: This endpoint allows you to send a WhatsApp message to a specified receiver.

#### **Request Body**

The request body must be a JSON object containing the following fields:

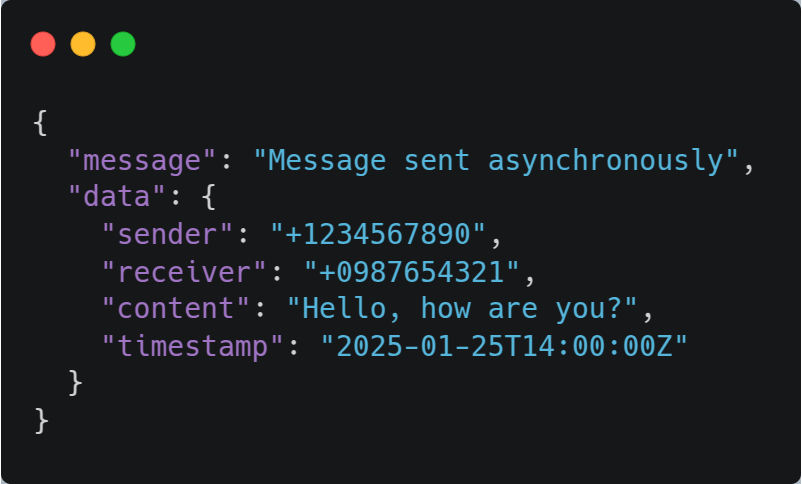
|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Description** | **Example** |
| sender | string | The sender's phone number or ID. | +1234567890 |
| receiver | string | The receiver's phone number or ID. | +0987654321 |
| content | string | The content of the message. | Hello, how are you? |

#### **Response**

On success, the server will respond with the following JSON object:

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| message | string | A success message. |
| data | object | The details of the created message. |
| sender | string | The sender's phone number or ID. |
| receiver | string | The receiver's phone number or ID. |
| content | string | The content of the message. |
| timestamp | string | Timestamp of when the message was created. |

##### **Example Response (Success)**



##### **Example Response (Error)**



### **2. Webhook - Receive Message**

* **URL**: /webhook/
* **Method**: POST
* **Description**: This endpoint receives incoming WhatsApp messages from a third-party service (like WhatsApp Business API or a simulation) and stores them in the database.

#### **Request Body**

The request body must be a JSON object containing the following fields:

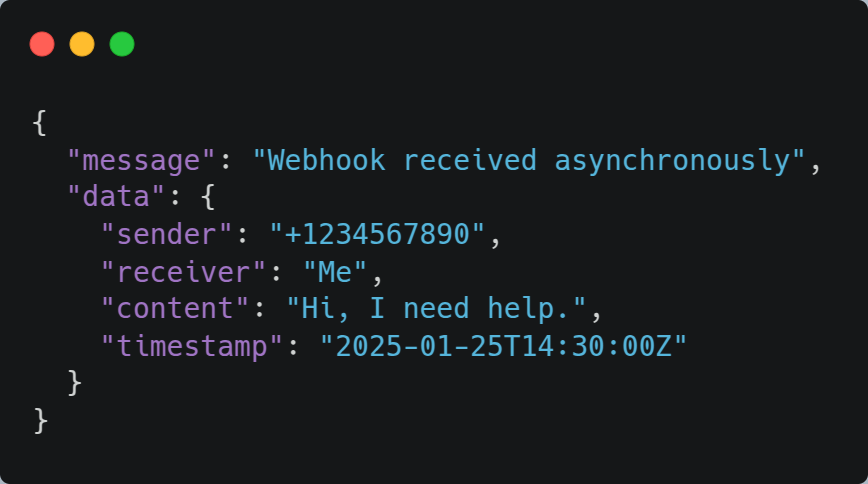
|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Description** | **Example** |
| sender | string | The sender's phone number or ID. | +1234567890 |
| content | string | The content of the received message. | Hi, I need help. |

#### **Response**

On success, the server will respond with a JSON object confirming receipt and storing the message:

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| message | string | A success message. |
| data | object | The details of the received message. |
| sender | string | The sender's phone number or ID. |
| receiver | string | The receiver's phone number or ID. |
| content | string | The content of the received message. |
| timestamp | string | Timestamp of when the message was created. |

##### **Example Response (Success)**



##### **Example Response (Error)**



## **Error Handling**

All error responses will follow the standard format:



The error message will provide details about the issue, such as missing required fields or invalid input.

## **Async Message Processing**

* The API uses async processing for both sending and receiving messages. This ensures that the message creation and processing are non-blocking, improving performance.
* When sending a message, the process is handled asynchronously to prevent blocking the request-response cycle.
* Incoming messages via the webhook are also processed asynchronously to handle multiple requests efficiently.

## **Future Improvements**

* **Authentication**: Implement token-based authentication (e.g., JWT) for better security.
* **Message Status Updates**: Implement endpoints to update message status (e.g., delivered, read).
* **Third-Party Integration**: Integrate with the WhatsApp Business API or another third-party messaging service for sending and receiving messages.
* **Error Handling**: Improve error handling with detailed error codes and custom exceptions.

## **Conclusion**

This API enables basic WhatsApp message sending and receiving functionality within a customer support system. It uses Django and Django REST Framework with asynchronous processing for better scalability. Future improvements can include advanced features like authentication, message status tracking, and third-party integration.