Zatpatmail Webhook

Webhooks in a Mail Agent

What is a Webhook?

Webhooks are user-defined HTTP callbacks that are triggered by an event. In Zatpatmail, Webhooks can be used to relay data based on recipient activity on the sent emails, to your application. The events for which the recipient data is relayed by the webhook are Opens, Clicks, and Bounces of the email. A Webhook instantly relays data to the configured URL and your application is instantly notified.

When the specific event occurs, the Webhook is triggered. That is, an HTTP POST request is made to the URL (or URI) configured by the developer to receive the webhook. The most useful feature of email webhooks is getting the user details when any or all of these events occur. The Webhook URL has to be configured in your https://bulk.zatpatmail.com/ account by the application developer to receive recipient details when these events occur. You can preview your webhook data within a Mail Agent and test it.

Types of Webhooks

We can classify webhooks under 3 types:

- Bounce Webhook
- Email Opens Webhook
- Link Clicks Webhook

Bounce Webhook

Using a bounce webhook you can receive the recipient details when your transactional emails bounce. There are multiple reasons for your emails to bounce. For example, if your recipient's mailbox is full, your email will bounce temporarily. Such temporary bounces are called as soft bounces. If the bounce is permanent, due to an invalid recipient email address, it is called a hard bounce.

Email Opens Webhook

Email Opens webhooks are used to retrieve the recipient details when they open your email. The list of user details sent by Zatpatmail can be seen here. This event is triggered the first time the recipient opens the email. Even if the recipient opens the email multiple times only the recipient details pertaining to the first instance are stored and transmitted by Zatpatmail.

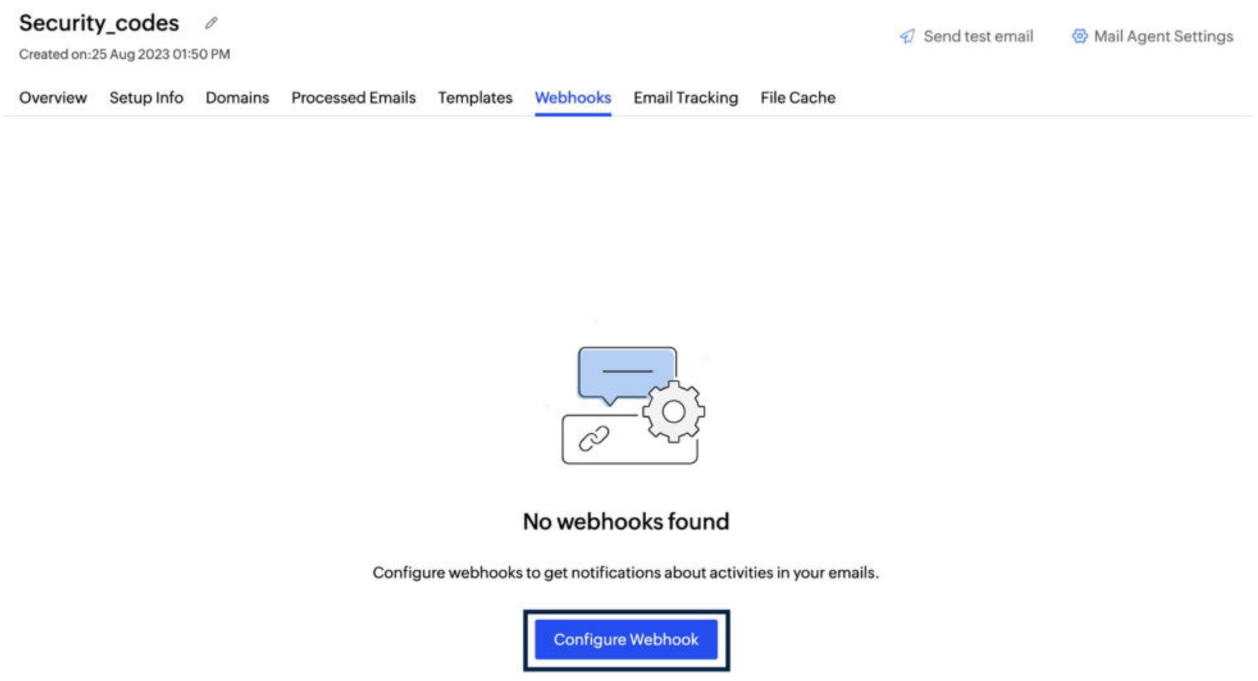
Link Clicks Webhook

Sometimes, your transactional emails will have clickable links as a part of its HTML body. Link Click webhook transmits the recipient details of those who clicked on these links. The list of user details sent by Zatpatmail can be seen here. Zatpatmail relays unique clicks information across all the tracked links in the email.

If a recipient clicks on the same tracked link multiple times, only the unique clicks are recorded and relayed by Zatpatmail to the application via the webhook URL.

Configure your Webhook URL

1. From the *left panel*, select Mail Agents and select a Mail Agent for which you want to add webhooks.
2. Navigate to Webhooks tab.
3. Click on Configure Webhook.



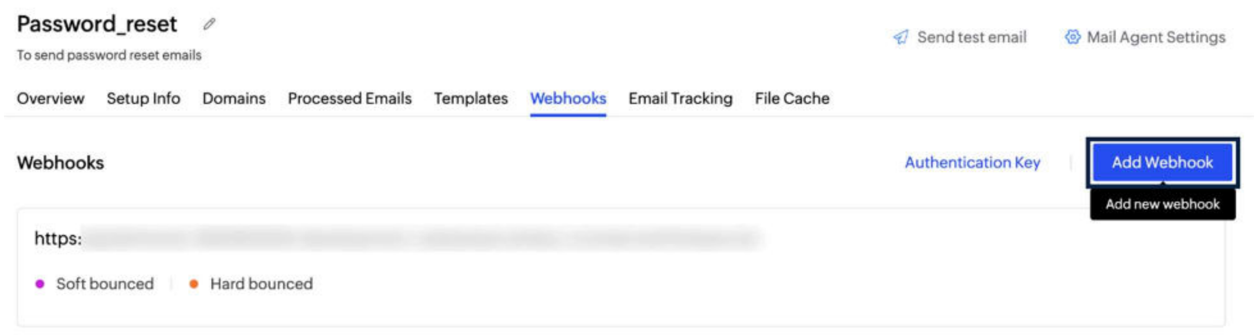
4. Add Webhooks box will appear. Here you can add your webhook and test it.

5. Enter Webhook URL, Description and choose the notifications (from Soft bounced, Hard bounced, Open and Click) for which you want the webhook to relay this data to your application.

6. Click on Save after the required details are filled.

7. Webhook is configured successfully.

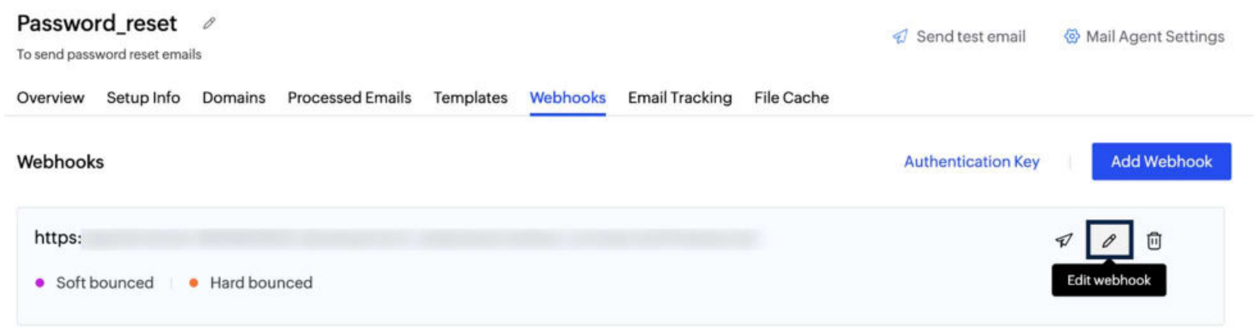
8. If you wish to add more webhooks to your Zatpatmail Mail Agent, then click on Add Webhook button on the right-hand side.



9. You can follow the same steps to add multiple webhooks.

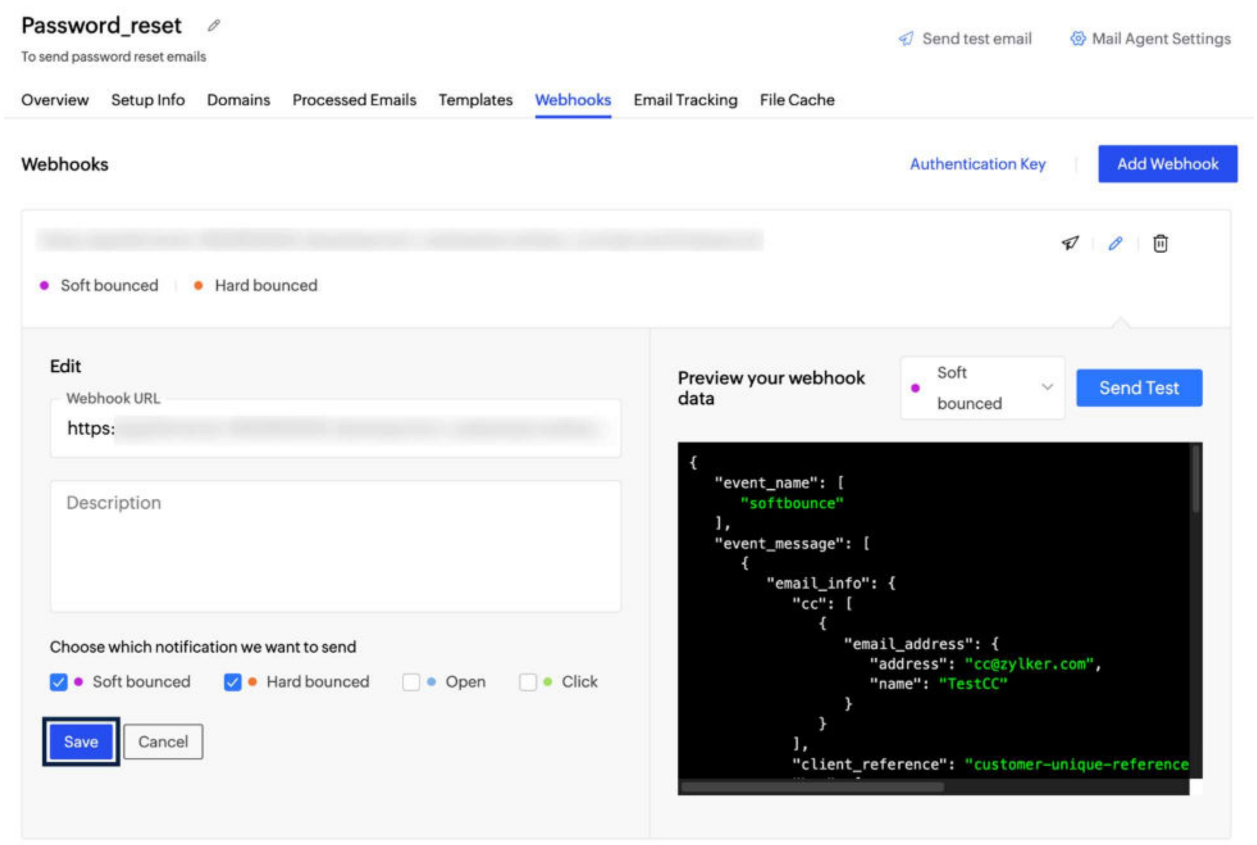
Edit your Webhook URL

1. From the *left panel*, select Mail Agents and select a Mail Agent for which you want to add webhooks.
2. Navigate to Webhooks tab. Here the list of webhooks configured for your Mail Agent will be listed.
3. Hover over Webhook to view Edit webhook icon.
4. Click on Edit webhook to open the Add Webhooks box.



5. You can edit Webhook URL, Description and select/ de-select from these notifications - Soft bounced, Hard bounced, Open and Click.

6. Click on Save to apply the changes to your Webhook URL.



[redacted]

Webhook URL: https: [redacted]

Choose which notification we want to send

☑ Soft bounced ☑ Hard bounced ☐ Open ☐ Click

```
{
   "event_name": [
      "softbounce"
   ],
   "event_message": [
      {
         "email_info": {
            "cc": [
               {
                  "email_address": {
                     "address": "cc@zylker.com",
                     "name": "TestCC"
                  }
               }
            ],
            "client_reference": "customer-unique-reference
```

Test your Webhook URL

You can test your webhook URL while creating a new webhook URL, or for an existing webhook URL. You can separately test your webhook URL for Soft bounce, Hard bounce, Open, and Click.

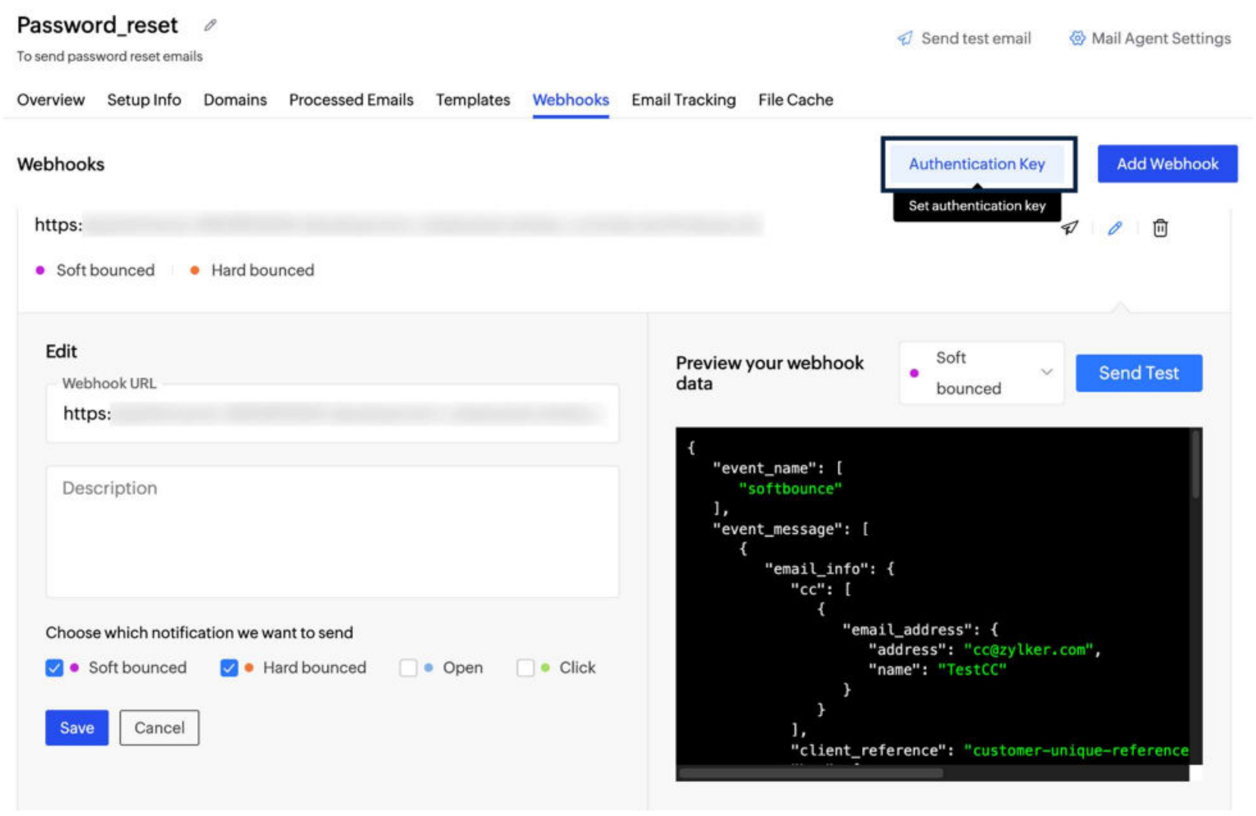
1. From the *left panel*, select Mail Agents and select a Mail Agent to which you want to add webhooks.
2. Navigate to Webhooks tab. You will see the Webhooks page where you can view the list of webhooks for that Mail Agent.
3. If you are adding a new Webhook, click on Add Webhook. If it is for an existing webhook URL, click on the *edit icon* located on the right side.
4. Now choose the notification from Soft bounced, Hard bounced, Open, and Click, you want to test by clicking on the drop-down.
5. Click on Send Test and check the successful reception of your data.

Authenticate your Webhook URL

Webhook URL should be made public for Zatpatmail to send data. Due to its public exposure, the URL is under threat. There is a possibility for hackers to manipulate the URL data by posting irrelevant information to the public webhook URL.

In order to protect the webhook URL, Zatpatmail provides an authentication key. Authentication protects your Webhook URL from irrelevant clicks. You can add an Authentication Key as a part of the data sent to the Webhook URL. Your application can then use this Authentication Key to validate the incoming webhook data.

1. From the *left panel*, select Mail Agents and select the Mail Agent to which you want to add webhooks.
2. Navigate to Webhooks tab. You will see the Webhooks page where you can view the list of webhooks for that Mail Agent.
3. Click on Authentication Key on the top right corner.



Password_reset

To send password reset emails

Send test email | Mail Agent Settings

Overview | Setup Info | Domains | Processed Emails | Templates | **Webhooks** | Email Tracking | File Cache

Webhooks

Authentication Key | Add Webhook

Set authentication key

https: [redacted]

● Soft bounced | ● Hard bounced

Edit

Webhook URL: https: [redacted]

Description

Choose which notification we want to send

[✓ ● Soft bounced] [✓ ● Hard bounced] ● Open ● Click

Save | Cancel

Preview your webhook data

● Soft bounced

Send Test

```
{
   "event_name": [
      "softbounce"
   ],
   "event_message": [
      {
         "email_info": {
            "cc": [
               {
                  "email_address": {
                     "address": "cc@zylker.com",
                     "name": "TestCC"
                  }
               }
            ],
            "client_reference": "customer-unique-reference
```

4. Enter Authentication Key and click on Set.
5. Use Authentication Key to validate the webhook data sent to your application.

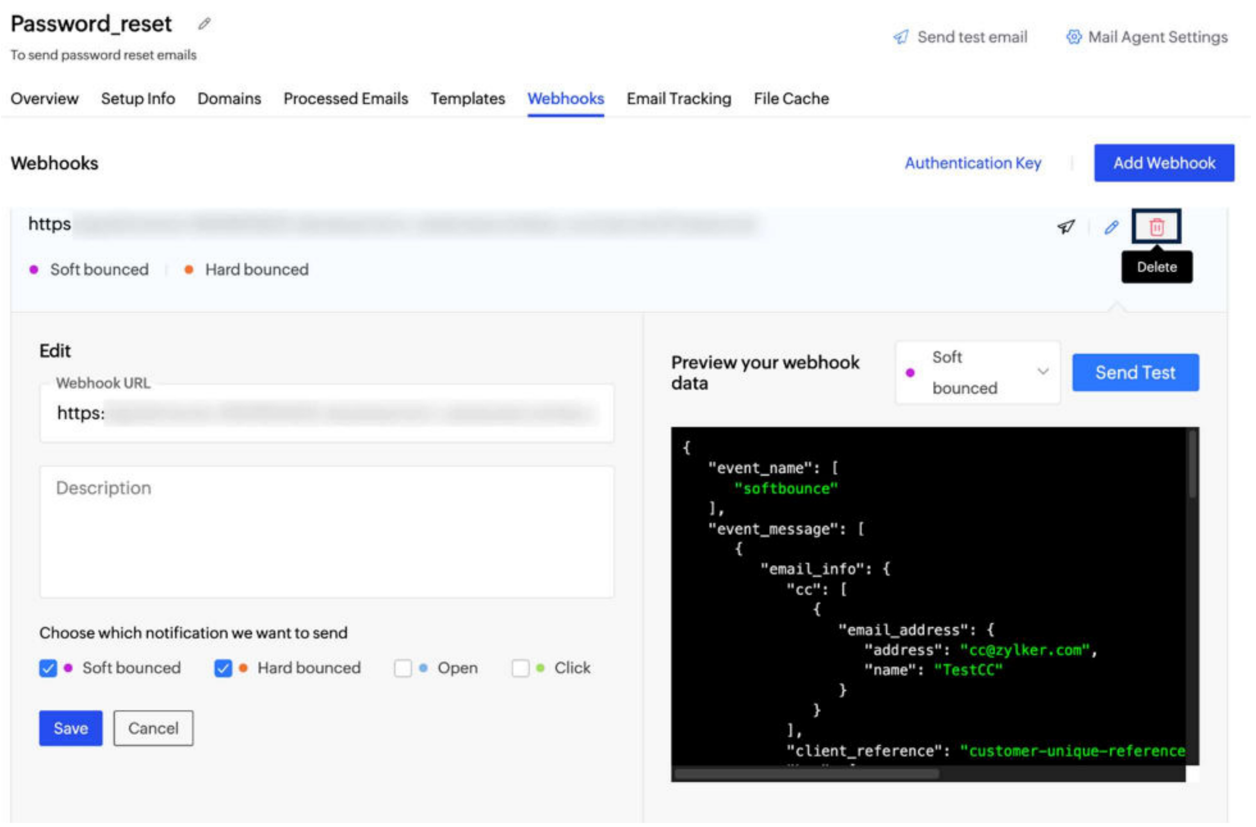
Delete your Webhook URL

You can delete a Webhook URL, however, you will have to add it again to re-use it.

1. From the *left panel*, select Mail Agents and select the Mail Agent to which you want to add webhooks.
2. Navigate to Webhooks tab. You will see the Webhooks page where you can view the list of webhooks for that Mail Agent.

3. Locate the webhook you want to delete and hover your cursor over it.

4. Click on the delete icon on the right corner.



Password_reset

To send password reset emails

Send test email | Mail Agent Settings

Overview | Setup Info | Domains | Processed Emails | Templates | Webhooks | Email Tracking | File Cache

Webhooks

Authentication Key | Add Webhook

https [redacted]

● Soft bounced | ● Hard bounced

Delete

Edit

Webhook URL: https: [redacted]

Description

Choose which notification we want to send

[✓ ● Soft bounced] [✓ ● Hard bounced] ☐ ● Open ☐ ● Click

Save | Cancel

Preview your webhook data: ● Soft bounced | Send Test

```
{
   "event_name": [
      "softbounce"
   ],
   "event_message": [
      {
         "email_info": {
            "cc": [
               {
                  "email_address": {
                     "address": "cc@zylker.com",
                     "name": "TestCC"
                  }
               }
            ],
            "client_reference": "customer-unique-reference
```

5. Click on OK to confirm the webhook deletion.

Standard Reported Data

Bounce webhook

- event_name - the type of event. This will be either hard bounce or soft bounce
- event_message - details about the event
    - email_info - This provides the details of the bounced email, like:
        - email_reference - used to trace the email
        - client_reference - it is the customer's unique reference to identify the mail
        - is_smtp_trigger - true means the webhook is triggered using an SMTP
        - subject - subject of the bounced email
        - bounce_address - the bounce email address configured in the Mail Agent and the name
        - from - the sender email address details

                - address - the sender email address
                - name - the sender name
            - to - email address to which the bounce details are to be sent. You can add any number of "to" addresses here
                - email_address - JSON array to add multiple email addresses
                    - address - the email address to which the bounce details are to be sent
            - processed_time - timestamp of when the email was sent
            - object - event_message data is for the object email
        - event_data - this field provides the reason for the bounce, time of bounce, and the diagnostic message
            - details - details of the bounce like reason, time, and diagnostic_message
                - reason - the reason for the bounce
                - time - the time of bounce
                - diagnostic_message - helps in trouble-shooting
            - object - event_data is for the object bounce. It can be softbounce or hardbounce
        - request_id - unique identifier of your email that was bounced
    - mailagent_key - the unique identifier of the Mail Agent
    - webhook_request_id - this is the unique id for a specific webhook transaction.

Email Opens webhook

- event_name - the type of event. The event tracked here is email opens.
- event_message - details about the event
    - email_info - this provides the details of the email that was opened, like:
        - email_reference - used to trace the email
        - client_reference - it is the customer's unique reference to identify the mail
        - is_smtp_trigger - true means the webhook is triggered using an SMTP
        - subject - subject of the email that was opened
        - bounce_address - the bounce address configured
        - from - the sender email address details

        - address - the sender email address
        - name - the sender name
    - to - email address to which the bounce details are to be sent. You can add any number of "to" addresses here
        - email_address - JSON array to add multiple email addresses
            - address - the email address to which the bounce details are to be sent
    - processed_time - timestamp of when the email was sent
    - object - event_message data is for the object email
- event_data - this field provides the details about the event
    - details - details like email_client, modified_time, ip_location_info, browser, operating_system, time, and device
        - email_client - details of the recipient email client
            - name - name of the recipient email client
            - version - version of the recipient email client
        - modified_time - time the open event occurred
        - ip_location_info - details of the recipient ip location
            - zipcode - zipcode of the recipient
            - country_code - country_code of the recipient
            - city - city of the recipient while the event occurred
            - latitude - latitude of the recipient while the event occurred
            - country_name - city of the recipient where the event occurred
            - ip_address - ip address used by the recipient while opening the email
            - time_zone - the recipient time zone
            - region - the region of the recipient
            - longitude - the longitude of the recipient
        - browser - the browser details used by the recipient to open the email
            - name - name of the browser

        - version - the browser version
      - operating_system - the details of the operating system used by the recipient
        - name - name of the recipient operating system
        - version - version of the recipient operating system
      - time - the time of the event occurrence
      - device - the details of the recipient device
        - name - name of the recipient device
    - object - event_data is for the object email_open
  - request_id - unique identifier of your email that was opened
- mailagent_key - the unique identifier of the Mail Agent
- webhook_request_id - this is the unique id for a specific webhook transaction

Link Clicks webhook

- event_name - the type of event. The event tracked here is email clicks.
- event_message - details about the event
  - email_info - this provides the details of the email that was clicked, like:
    - email_reference - used to trace the email
    - client_reference - it is the customer's unique reference to identify the mail
    - is_smtp_trigger - true means the webhook is triggered using an SMTP
    - subject - subject of the email that was clicked
    - bounce_address - the bounce address configured
    - from - the sender email address details
      - address - the sender email address
      - name - the sender name
    - to - email address to which the bounce details are to be sent. You can add any number of "to" addresses here
      - email_address - JSON array to add multiple email addresses
        - address - the email address to which the bounce details are to be sent
    - processed_time - timestamp of when the email was sent

    - object - event_message data is for the object email
- event_data - this field provides the details about the event
    - details - details like email_client, modified_time, ip_location_info, browser, operating_system, time, and device
        - email_client - details of the recipient email client
            - name - name of the recipient email client
            - version - version of the recipient email client
        - modified_time - time the open event occurred
        - ip_location_info - details of the recipient ip location
            - zipcode - zipcode of the recipient
            - country_code - country_code of the recipient
            - city - city of the recipient while the event occurred
            - latitude - latitude of the recipient while the event occurred
            - country_name - city of the recipient where the event occurred
            - ip_address - ip address used by the recipient while opening the email
            - time_zone - the recipient time zone
            - region - the region of the recipient
            - longitude - the longitude of the recipient
        - browser - the browser details used by the recipient to open the email
            - name - name of the browser
            - version - version of the recipient operating system
        - operating_system - the details of the operating system used by the recipient
            - name - name of the recipient operating system
            - version - version of the recipient operating system
        - time - the time of the event occurrence
        - device - the details of the recipient device

            - name - name of the recipient device
        - object - event_data is for the object email_link_click
    - request_id - unique identifier of your email that was clicked
- mailagent_key - the unique identifier of the Mail Agent.
- webhook_request_id - This is the unique id for a specific webhook transaction.

Securing Webhooks

Securing your webhooks is recommended as it helps you determine if the requests have actually originated from Zatpatmail. To enable you to verify the webhooks, Zatpatmail adds a signature to all its webhook requests. This adds an extra layer of security to your webhooks.

Validating Webhook requests

Each webhook request contains a producer-signature in their request header that is used to verify whether the request is generated by Zatpatmail. The producer signature consists of 3 parts - *timestamp*, *signature*, and *signing algorithm*.

- ts- timestamp is the time when the webhook request was initiated from Zatpatmail server to user-configured URL
- s - the signature is the MAC message generated on encoding the webhook event notification with the authentication key added by you in your Mail Agent
- s-algorithm - signing algorithm is the standard HMAC SHA256 algorithm used to sign the payload of the webhook request

Sample producer-signature format: ts=1596109465823;s=dN0yVozgabP5NPlxMDfP1r5u65bVO9kTGEZMIQlqI2o%3D;s-algorithm=HmacSHA256

You can use the below sample code in your application to validate the webhook request. This algorithm will return a Boolean value of either True or False. True means that the webhook request is generated by Zatpatmail and False means the webhook request is not from Zatpatmail.

Sample code:

```
import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

import javax.servlet.http.HttpServletRequest;

import java.io.BufferedReader;

import java.net.URLDecoder;

import java.security.MessageDigest;
import java.time.Duration;

import org.apache.commons.codec.Charsets;

import org.apache.commons.codec.binary.Base64;

import org.json.JSONObject;



public class WebhookRequestValidator {


/**
*
* @param request
* @param acceptableDuration - The maximum genuine delay that the request can suffer from
the time of Request initiated from Zatpatmail server to that of the configured webhook URL
destination.
* @param secretKey - Secret key / Authentication key configured in the Zatpatmail portal
* @throws Exception
**/
public static void authenticateWebhookContent(HttpServletRequest request, Duration
acceptableDuration, byte[] secretKey)

throws Exception {


String producerSignature = request.getHeader("producer-signature");

String psDecoded = URLDecoder.decode(producerSignature, "UTF-8");

String sign[] = psDecoded.split(";");


JSONObject signJSON = new JSONObject();


for (int i = 0; i < sign.length; i++) {


String field = sign[i];

String fieldArr[] = field.split("=", 2);

signJSON.put(fieldArr[0], fieldArr[1]);
```

```
}

long timeSent = Long.valueOf(signJSON.getString("ts"));
long diff = System.currentTimeMillis() - timeSent;
long acceptableLimit = acceptableDuration.toMillis();

if (diff <= acceptableLimit) {

String signatureAlgorithm = signJSON.getString("s-algorithm");
String signatureReceived = signJSON.getString("s");
StringBuffer requestBody = new StringBuffer();
String value = null;
BufferedReader reader = request.getReader();
while ((value = reader.readLine()) != null) {
requestBody.append(value);
}

String data = requestBody.toString();
String dataDecoded = URLDecoder.decode(data, "UTF-8");
String arr[] = dataDecoded.split("=", 2);

String dataValue = arr[1];
String constructedSignature = getSignature(dataValue, secretKey, signatureAlgorithm);

byte decode1[] = Base64.decodeBase64(signatureReceived);
byte decode2[] = Base64.decodeBase64(constructedSignature);

if (!MessageDigest.isEqual(decode1, decode2)) {
throw new Exception("Message digest isn't equal");
}

}

}


private static String getSignature(String data, byte[] secretKey, String
signatureAlgorithm) throws Exception {

SecretKeySpec signingKey = new SecretKeySpec(secretKey, signatureAlgorithm);


Mac mac = Mac.getInstance(signatureAlgorithm);

mac.init(signingKey);


byte[] rawHmac = mac.doFinal(data.getBytes(Charsets.UTF_8));


String result = Base64.encodeBase64String(rawHmac);

return result;

}


}

;i++)>
```

Note:

Before you get started it is important to know that Zatpatmail is for sending transactional emails like welcome emailers, password resets emails, OTPs. We do not support sending of emails or promotional emails like newsletters or marketing campaign emails.

======================{End Of Document}======================