Introduction

---

yalidine provides a Webhook service to notify your application (website) when events occur. These notifications are almost realtime and provides an alternative to polling the REST API. The purpose is to be able to trigger event-based behavior as soon as an event occurs (like when a parcel's delivery status has changed, a parcel has been created, edited, deleted, etc). This way you will be able to build gamification mechanism, live dashboard, synchronizing contents, etc.

Using the webhooks prevents you from sending repetitive query to the API and helps you preserve your quotas. Webhooks will notify you as soon as something happen in your account.

Why Use Webhooks

Let's say you've registered to receive the parcel_status_updated event. When the yalidine delivery agent takes your parcel to delivery, a webhook from yalidine will tell your app that a specific parcel (tracking) has got a new status Sorti en livraison. After your webhook endpoint receives the parcel_status_updated event, your app can then run a specific action (Ex. sending a message to your parcel recipient to tell him that his parcel will be delivered today).

Steps to receive webhooks

---

to receiving event notifications in your app, you can follow these guidelines :

1. Create a webhook endpoint as an HTTPS endpoint (url) on your server.
2. Your endpoint must returns a 200 response status and a valid crc_token.
3. Create a webhook in your yalidine Webhooks Dashboard.
4. Develop and then test that your webhook endpoint is working properly with the webhooks test page.
5. When your webhook is ready, change its status to 'active'
6. Then, You will receive the notification event as soon as it happens

Best Practices

To ensure that your webhooks remains secure and functions seamlessly, we strongly recommend implementing these best practices.

---

crc_token Validation

Make sure that your endpoint always contains the crc_token validation code. Yalidine will try to validate your webhook from time to time. If your endpoint does not return a valid crc_token your webhook will be disabled. Read more on Validate A Webhook section.

Event Delivery

Yalidine deliver the event notification request reasonably fast.
Each webhook delivery contains one or many event of the same event type (We group the events of the same type and send them in the same notification).
Read more on Events Format section.

Event Fields

Expect new fields to appear without notice in event structure, and make sure that this can be handled by your code. We will preserve existing field though for backward compatibility as long as we can.

Response

Your endpoint must return a 200 response code in less than 10 seconds. When you receive a payload we strongly recommend the following :

1. Do not run a long time-consuming script.
2. Store the payload in a queue for background processing
3. Return the response code immediately.
4. Handle the received events later with another script.

Otherwise, Your webhooks can be disabled. Read more on The Retry Policy section.

Retry Logic

Be aware of the retry logic on The Retry Policy section.

Disable Webhook Logic

Yalidine will attempt to notify you via email if an endpoint has not responded with a 200 HTTP status code for multiple days in a row. The email also states when the endpoint will be automatically disabled

---

Duplicate Events

Your webhook endpoints might receive the same event more than once. Each event has a unique event_id so you can prevent processing the duplicate events by logging the events you've processed, and then not processing already-logged events.

Order of events

Yalidine does not guarantee delivery of events in the order in which they are generated. However, We provide in every event the exact point in time the event was occurred_at.

Security

To keep your endpoint secure, You will receive a signature in every webhook delivery to verify events are coming from Yalidine. Read more on Secure Your Webhook section.

Secret Key

The webhook delivery's signature is generated using the payload and your Secret Key. You can find or regenerate your secret key in your Webhooks Dashboard.

Events Format

The payload is always sent to your endpoint in `JSON` format with `POST` request.

We also send a signature in the header, this will help you secure your webhooks. Read more on Secure Your Webhook

Every payload consist of two top-levels elements:

- type: One of the type you subscribed to. You can build your endpoint logic depending on the value if this element
  For example, if the value of type is parcel_created then do this. if the valus is [another event type you subscribed to], do this.
  Please note we always send one type per request.
- events: This is An array of generated events.
  Each element of this array is an event and always contains three mid-levels elements:
  - event_id: A unique identifier of the event. You can use it to prevent deduplication.
  - occurred_at: The point of time when the event was generated. You can use it to preserve the order of the events you received.
  - data: The resource that contains the concerned object and if applicable any new value of a specified parameter
    Please not that the content of this element changes depending on the type of the event. please see the example belows for each event type.

Events

Those are the event you can subscribe to:

Event type
parcel_created
parcel_edited
parcel_deleted
parcel_status_updated
parcel_payment_updated

Payload example

parcel_created

JSON


```
{
    "type": "parcel_created",
    "events": [
        {
            "event_id": "sf4qRk6WA395IcBYiQU8PdJn2vOSTgEV",
            "occurred_at": "2022-04-28 00:01:26",
            "data": {
                "order_id": "myfirstorder",
                "tracking": "yal-111AAA",
                "label":
"https:\/\/yalidine.app\/app\/bordereau.php?tracking=yal-111AAA&token=TEptSkZ2d
Xc4clhhQ3A1ODNXQ3VqQT09",
```

```
                "import_id": 654
            }
        },
        {
            "event_id": "PcKlsmzvoNQAgJ4Y8np2xryRVIwOMDGU",
            "occurred_at": "2022-04-28 00:01:26",
            "data": {
                "order_id": "mysecondorder",
                "tracking": "yal-222BBB",
                "label":
"https:\/\/yalidine.app\/app\/bordereau.php?tracking=yal-222BBB&token=TEptSkZ2d
Xc4clhhQ3A1ODNXQ3VqQT09",
                "import_id": 964
            }
        }
    ]
}
```


---

parcel_edited

JSON


```
{
    "type": "parcel_edited",
    "events": [
        {
            "event_id": "Oja36BwdCIiShQoxupWXUy9fVz7GYenv",
            "occurred_at": "2022-04-28 00:01:26",
            "data": {
                "tracking": "yal-111AAA",
```

```
                "label":
"https:\/\/yalidine.app\/app\/bordereau.php?tracking=yal-111AAA&token=TEptSkZ2d
Xc4clhhQ3A1ODNXQ3VqQT09"

            }

        },

        {

            "event_id": "ZCQJwd8Xvx92hpEtyRI0bcS3azrLg7Wf",

            "occurred_at": "2022-04-28 00:01:26",

            "data": {

                "tracking": "yal-222BBB",

                "label":
"https:\/\/yalidine.app\/app\/bordereau.php?tracking=yal-222BBB&token=TEptSkZ2d
Xc4clhhQ3A1ODNXQ3VqQT09"

            }

        }

    ]

}
```

---

parcel_deleted JSON

```
{

    "type": "parcel_deleted",

    "events": [

        {

            "event_id": "KT2fF0MhpHr7tUlc4BxPLDi3uAjdk6Q5",

            "occurred_at": "2022-04-28 00:01:26",

            "data": {

                "tracking": "yal-111AAA"

            }
```

```
        },
        {
            "event_id": "EbM1YSPKOT3pGQLsw4jvry5mzntelohF",
            "occurred_at": "2022-04-28 00:01:26",
            "data": {
                "tracking": "yal-222BBB"
            }
        }
    ]
}
```

---

parcel_status_updated

JSON

```
{
    "type": "parcel_status_updated",
    "events": [
        {
            "event_id": "AWFRSYDvqpfyVnP8hCb65NGlTMrzJ1ou",
            "occurred_at": "2022-04-28 00:01:26",
            "data": {
                "tracking": "yal-111AAA",
                "status": "Sorti en livraison",
                "reason": null
            }
        },
        {
            "event_id": "2UpjH48hBQ0dozy3XmaVb6uk71CGRxWs",
```

```
            "occurred_at": "2022-04-28 12:01:26",
            "data": {
                "tracking": "yal-222BBB",
                "status": "Tentative \u00e9chou\u00e9e",
                "reason": "Client ne r\u00e9pond pas"
            }
        }
    ]
}
```

---

parcel_payment_updated

JSON

```
{
    "type": "parcel_payment_updated",
    "events": [
        {
            "event_id": "zCMSocNIUG0uTep53v629WtA7xhnDrPZ",
            "occurred_at": "2022-04-28 00:01:26",
            "data": {
                "tracking": "yal-111AAA",
                "status": "ready",
                "payment_id": null
            }
        },
        {
            "event_id": "VSncCO39fF8k2plXmrDAJZKtQ6qsdg5a",
            "occurred_at": "2022-04-28 12:01:26",
```

```
            "data": {
                "tracking": "yal-222BBB",
                "status": "receivable",
                "payment_id": "pmt-123456"
            }
        }
    ]
}
```

Retry Policy

When we send a webhook-delivery to your endpoint, you must respond in less than 10 seconds with a 200 http code.
Otherwise, this retry policy will apply

The retry policy consist of doing a total of 7 tentative with an exponential time interval

You will receive an email after each failed retry

After the last tentative, your webhook is automatically disabled.

Retry policy table

Tentative #	proceed After	Hour example
1	Immediatly	2025-05-04 16:20:45
2	5 minute(s) after the previous tentative	2025-05-04 16:25:45
3	15 minute(s) after the previous tentative	2025-05-04 16:40:45

4	1 hour(s) after the previous tentative	2025-05-04 17:40:45
5	3 hour(s) after the previous tentative	2025-05-04 20:40:45
6	12 hour(s) after the previous tentative	2025-05-05 08:40:45
7	1 day(s) after the previous tentative	2025-05-06 08:40:45

Please note: The interval in the previous table are count from the date of the previous tentative, and not the date the event occurred at.

This mean the retry policy will run for 40 hours after the first try.

Create A Webhook

Before creating the webhook subscription, your need to create a file (endpoint) in your server first.
This endpoint must return a valid crc_token.
Read more on Validate A Webhook section.

When your endpoint is ready for validation, you will be able to create a webhook successfully.

So, to create w webhook:

1. Go to Webhooks Dashboard.
2. Click on the add button.
3. Fill the form with corresponding values
    - **Webhook Name:** A name that let you recognize it afterwards.
    - **URL of reception:** An HTTPS url of your endpoint (This endpoint must return a valid crc_token)
    - **Email:** An email address to receive alert notifications if something goes wrong with the delivery of your events

    - **Event types:** The event you would like to subscribe to. You can choose one or many from the list :
      parcel_created, parcel_edited, parcel_deleted, parcel_status_updated, parcel_payment_updated,
4. Click on the Proceed button

Important: When you click the proceed button, to be sur that you own the chosen url, we will try to validate your endpoint url. Read more on Validate A Webhook section.

Your webhook will only be created if it passes this validation.

When your webhook is created, it is automatically disabled. You can then :

- Test your webhook
- Enable it

Validate A Webhook

To ensure that you own the url of your webhook endpoint, we do a **Challenge-Response Check (CRC)**

This validation occurs in the following situations:

- The creation of a webhook
- The edition of a webhook
- From time to time to validated already created webhooks

This is how we validate the endpoint:

- We will send a `GET` request to your endpoint URL containing two parameters: subscribe and crc_token
- Your endpoint must always check if the subscribe GET parameter is set (see example below)
- When it is set, you have to echo the crc_token value we sent
- Your endpoint will be validated only when :
    - You respond with a 200 code status in less than 10 seconds
    - You echo correct value of crc_token

Example

Endpoint Validation

```php
<?php

    // validation start.


    // This code must always be present to prevent your webhook from being disabled.

    if(isset($_GET["subscribe"], $_GET["crc_token"])) {

        echo $_GET["crc_token"];

        exit();

    }

    // validation end.


    /* rest of the code

        ...

*/
```

Important: The validation code must always be present in your endpoint code. We do webhooks validation from time to time, if the validation fails, your webhook will be disabled

Edit A Webhook

To edit a webhook:

1. Go to Webhooks Dashboard
2. Click on the edit button in the webhook card you wish to edit
3. Make the changes you want
4. Click on Proceed

When you edit a webhook, yalidine will validate the url again.
These actions are added in the log of your webhook.

Delete A Webhook

To delete a webhook:

1. Go to Webhooks Dashboard.
2. Click on the delete button in the webhook card you wish to delete.
3. Confirme the deletion of the selected webhook.

When you delete a webhook all its log datas are deleted too.

Secure Your Webhook

This steps are not mandatory, but we highly recommend securing your webhook.

To secure your webhook you need to be sure that this is yalidine who sent you the webhook.

To do that, we send you in every webhook-delivery a signature X_YALIDINE_SIGNATURE in the header
This signature is generated by hash_hmac using these parameters :

- The sha256 algorithm
- The payload as the data
- Your webhook secret_key as the key

In your endpoint, before handling the payload, you need to do the following

1. Check if the HTTP_X_YALIDINE_SIGNATURE is present in the header
2. If yes, compute a verification_hash using :
    - The sha256 algorithm
    - The payload as the data
    - Your webhook secret_key as the key
3. Compare the value of HTTP_X_YALIDINE_SIGNATURE with the verification_hash you generated

4. If both are the same, the payload came from yalidine and can be handled
5. Otherwise, ignore the payload as it may affect the integrity of your data

You can regenerate your webhook secret_key in the webhook dashboard

Example

Secure Your Webhook

PHP

```php
<?php

    // validation start.

    // This code must always be present to prevent your webhook from being disabled.

    if(isset($_GET["subscribe"], $_GET["crc_token"])) {

        echo $_GET["crc_token"];

        exit();

    }

    // validation end.


    // Signature verification

    // to be sur that the event came from yalidine we need to verify the signature (found in the header)

    if(isset($_SERVER["HTTP_X_YALIDINE_SIGNATURE"])) {

        // the signature is set, so we continue


        // we declare the variables

        $secret_key = "MY_SECRET_KEY"; // you can find it in your webhook dashboard

        $yalidine_signature = $_SERVER["HTTP_X_YALIDINE_SIGNATURE"]; // the yalidine signature for this event

        $payload = file_get_contents("php://input"); // the raw events data (json)
```

```
        // to verify the signature we should compute the payload and the
secret_key with the hash_hmac() function

        $computed_signature = hash_hmac("sha256", $payload, $secret_key);


        // we verify if the $yalidine_signature is the same as the
$computed_signature

        if($yalidine_signature === $computed_signature) {

            // autorisation successful, it's sent by yalidine

            // what next ? save the $payload in the database and return 200
http response immediately.


            /*

             * It is recommended to save the received data ($payload) in the
database

             * and proceed it with another script.

             * a 200 response code should be received immediately within 10
seconds of receiving the data

             * otherwise, if the webhook will be disabled

             */


        } else {

            // autorisation failed. it's not sent by yalidine
header("HTTP/1.1 400 Bad Request");

            exit("invalide signature");

        }



    } else {
```

```
        // the signature is not set, so we exit with a bad request 400
        header("HTTP/1.1 400 Bad Request");
        exit("invalide signature");
    }
```

Test Your Endpoint

To test your webhook endpoint:

1. Go to Webhooks Dashboard
2. Click on the make tests button in the webhook card you wish to test
3. You will see all the event type you subscribed to
4. Click on the Send a test button
5. You will receive in you app, a webhook delivery of some data
6. The result of the test will be shown near the button you clicked

You can use this method to develop or test your webhook endpoints.
These tests are not added to the webhook log.

Enable Disable A Webhook

To change the status of a webhook:

1. Go to Webhooks Dashboard
2. Click on the change status button in the webhook card you wish to edit
3. Make the changes you want
4. Click on Yes, change the status

You will not receive any events if the webhook is disabled.

Important: When you disable a webhook, if some events are still present in the queue, the retry policy will be applied. These events will only be deleted if they have consumed all possible attempts.

If you enable the webhook again before the deletion, you will receive them as normal as possible.

View The Logs

To view the log of a webhook:

1. Go to Webhooks Dashboard
2. Click on the log button in the webhook card you wish to see

The log are kept for 48 hours.
When a webhook is deleted, all its log will be deleted too.