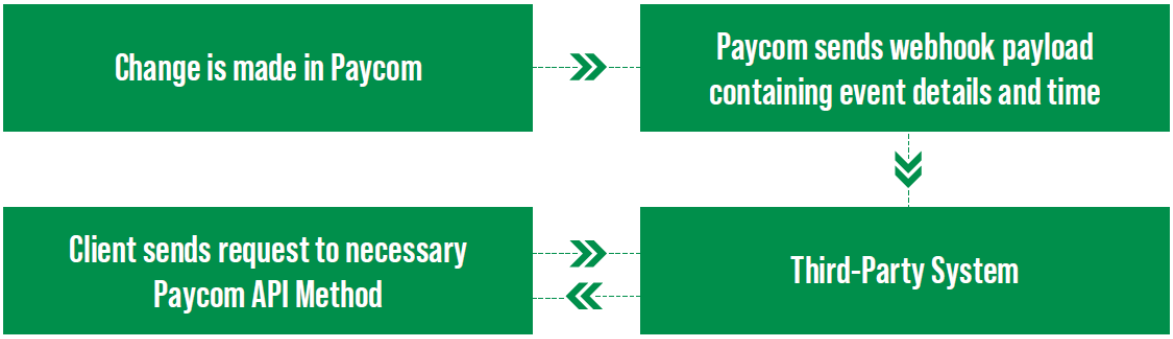
Webhooks

Introduction

Paycom's webhooks are like a reverse API that Paycom kicks off any time a certain event occurs. This means event information, like an employee status change, will be transmitted directly to a third party application or middleware outside of Paycom right after it happens. The webhook will contain details about the event and then the receiving party can do an API call to get the full details. This allows clients to avoid constantly polling the API to retrieve this information greatly reducing the number of API requests needed.



How does this differ from API usage without webhooks?

Example: A client with 500 employees wanted to query Paycom for employee changes.

On one day this client had changes for 100 employees. Without webhooks, this client would set up a process to continually call the API to look for changes. The client decides to set this up on a 15 minute interval. That means every 15 minutes, the integration would need to poll Paycom 500 times to get all employee changes during the last 15 minute window. 500 X 4 X 24 = 48,000 calls a day to capture 100 changes. With webhooks this will reduce the number of calls to 100 calls that day.

Utilizing webhooks will dramatically reduce the amount of traffic while maintaining the near real-time information flow for the new hire provisioning process or updates to employees.

Available Event Notifications

- New Hire Created
- New Hire Deleted
- New Hire Self Onboarding Completed
- Employee Added
- Employee Photo Added

Available Employee Change Events

Additional Schedule Group	Employee Position Change	Position Family
Alternate Pay Frequency	Employee Re-Hire	Position Level Change
Available Health Insurance	Employee Status	Position Seat Change
Badge Level	Ethnic Background	Position Title Change
Badge Number	Exempt Status	Previous Termination Date
Birth Date	Full Time to Part Time Date	Primary Schedule Group
Clock Sequence Number	Gender	Retirement Plan
Cobra End	Highly Comp Employee	Schedule Time Zone
Cobra Start	Hire Date	Statutory Employee
Commission Only	Hourly Or Salary	Supervisor Learning
Current Key Employee	Hours 401K	Supervisor Primary
DOL Status	Last Pay Change	Supervisor Quaternary
EE Message	Last Position Change Date	Supervisor Secondary
EEOC Class	Last Review	Supervisor Talent
Eligible 401K	Marital Status	Supervisor Tertiary
Email Change	Match Eligible	Supervisor Time-Off Approval
Emergency Contact Information	Next Review	Terminal Access Group
Employee Address Changes	Non Resident Alien	Termination Date
Employee Department Changes	On-Leave Start and End Date	Termination Reason
Employee GL Code	Part Num 401K	Union Code
Employee Labor Allocation Changes	Part Time To Full Time Date	Vets 4212 Emp Category
Employee Location Changes	Pay Class	Vets 4212 Job Category
Employee Name Changes	Pay Frequency	Workers Comp Code
Employee Phone Changes		

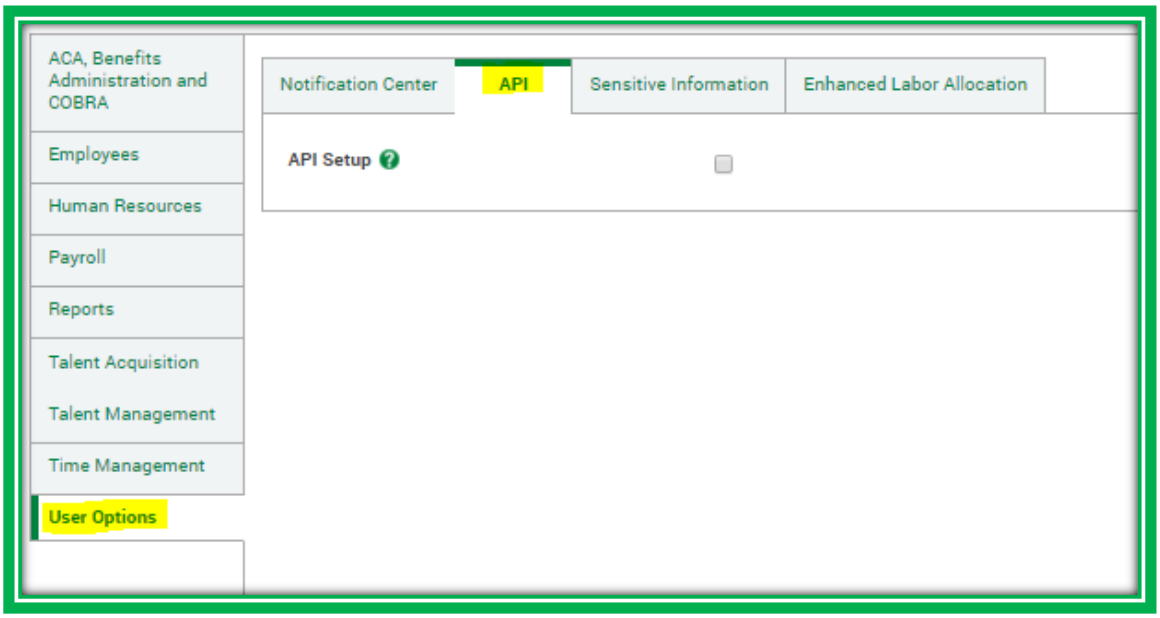
Requirements

- Familiarity with Paycom API – webhooks are an extension of the Paycom API. Webhook notifications will require you to make an API call to retrieve the detailed information about the event.
- You will need to setup a webhook endpoint: usually a web server running .NET, PHP, or similar. This server must be able to accept web requests from Paycom.
- When messages are received at your server, we highly recommend to receive the message, validate it's from Paycom, and insert into a job queue and then promptly respond to Paycom that you received the message.
- You must use an HTTPS URL for your webhook endpoint, Paycom will validate that the connection to your server is secure before sending your webhook data. For this to work, your server must be correctly configured to support HTTPS with a valid server certificate.
- 200 Response Status Code. Paycom expects a 2xx HTTP status code after the payload is posted to the configured endpoint. If there is no response, timeout occurs, or an HTTP status code

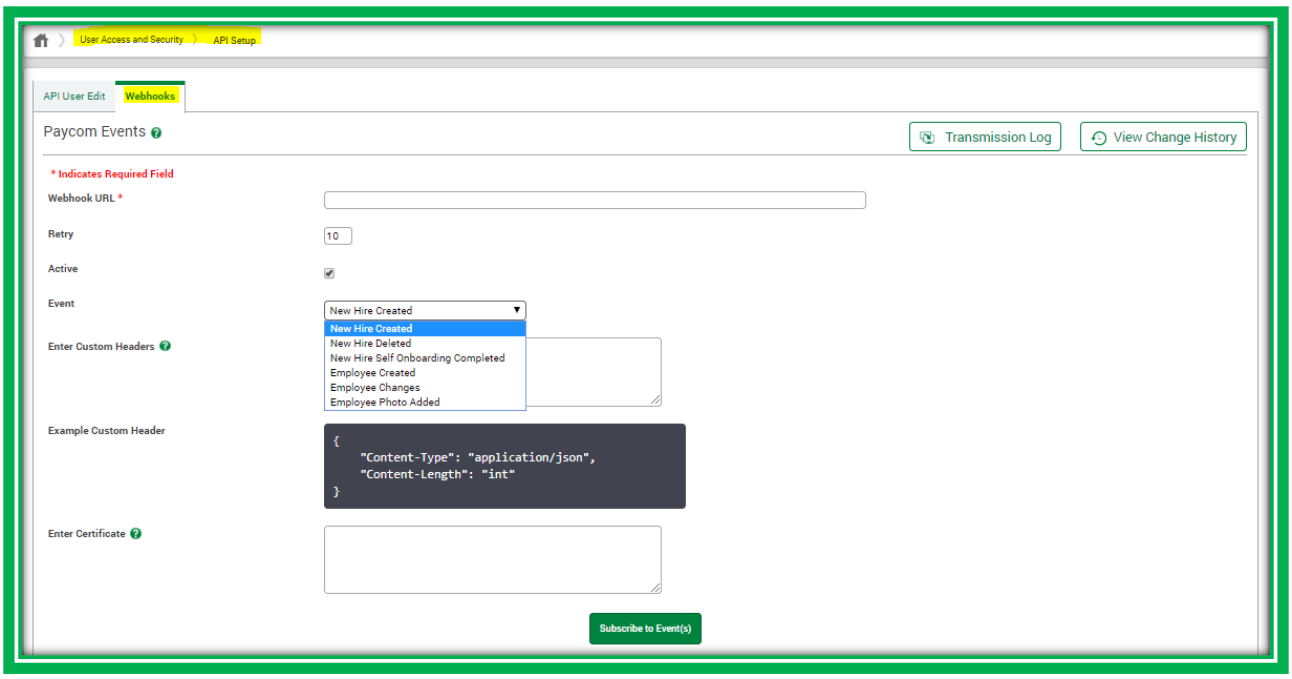
other than 2xx is returned, the webhook payload will be sent again after 60 seconds. This defaults to 10 tries before failing entirely. The number of retries is configurable on the webhook configuration page inside Paycom.

Configuration and Setup

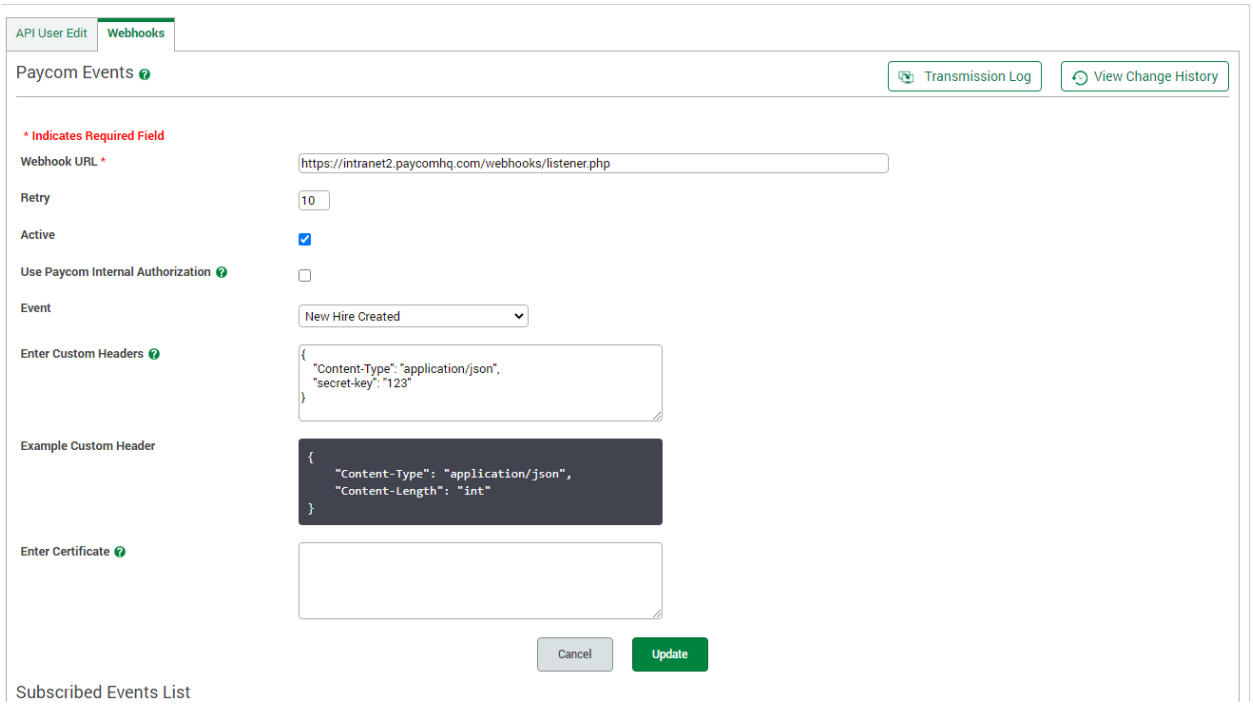
The webhook configuration page can be found in the API Setup menu in Paycom's client portal. You will need client user access and permission profile adjusted to have access to this API setup menu. If you need assistance with updating your permission profile, please contact your dedicated specialist.



The configuration page is where you will enter your webhook endpoint, custom headers, and configure which events will generate webhooks. A transmission log is provided where past events can be viewed.



For reference, the configuration for all the examples in the next section:



The key item is utilizing the custom header, which contains a secret-key JSON value. In our example, this is the method to validate the incoming webhook is coming from Paycom. Depending on your setup, you can include more values, and this custom header can be set per event you subscribe to.

This example configuration above will generate this header:

```
{
```

```
  "Content-Type":"application/json",
  "Content-Length":"311",
  "Secret-Key":"123",
  "User-Agent":"GuzzleHttp/6.5.3 curl/7.55.0 PHP/7.1.10",
  "Host":"intranet2.paycomhq.com"
}
```

Possible Webhooks

This table contains a list of possible webhooks with information about each one.

Event	Description	Use	Next Step
Subscribed	This event notifies you that a new subscription was setup.	Notification of new event subscription	None, this is a notification
Verification URL	This is sent when verifying your webhook setup. Paycom will send an authorization code with this event, which must be entered into Paycom to finalize setup. Failure to enter code will prevent any event notifications.	Used to validate a new webhook URL (NOTE: Once a URL is validated, further validations are not required)	Use the registration code to verify the webhook URL
Ping	This is a manually triggered event that can be sent from the Paycom Webhook Interface. Usual for testing webhook communication.	Used to test a webhook URL	None, this is a notification
New Hire Created	This event is triggered when a new hire is created. This is part of the New Employee Queue portion of Paycom.	Useful to know when a new hire gets added	Do an API call to get new hire information
New Hire Deleted	This event is triggered when a new hire is archived. This is part of the New Employee Queue portion of Paycom.	Useful to know when a new hire gets archived	Stop any process concerning new hire in other systems.
New Hire Self Onboarding Completed	This event is triggered when a new hire self-onboarding is completed. This is part of the New Employee Queue portion of Paycom.	Useful to know that a new hire completed their self-onboarding. During self-onboarding an employee will enter their personal information.	Do an API call to get new hire information
Employee Added	This event is triggered when an employee is added. This can be triggered a multitude of ways: Employee import is completed, an employee is manually added, or an employee moves from new hire to employee.	Useful to know when a new hire gets converted to an employee. Or an employee gets added manually. At this stage, most HR workflows will need IT resources	Do an API call to get employee information. Trigger an IT onboarding process to create accounts, create service desk tickets for equipment needs, etc…

		allocated next to prep for the employee.	
Employee Photo Added	This is event is triggered when an employee's photo is updated.	Useful to know when an employee's photo has been updated.	Do an API call to retrieve the photo and update other systems like Active Directory.
Employee Change	This event is when selected subscribed employee changes occur. A callback URL will be provided that gives a direct API link to the change.	Useful to know that some sort of employee information changed. See Paycom Webhook Configuration Screen for available fields that can subscribed to.	Do an API call to retrieve change information and update other systems.
Unsubscribed	This event notifies you that a new subscription was deleted in the Paycom Webhook Configuration Menu.	Useful to know an event subscription was deleted for your webhook URL	Perhaps trigger an alert that a webhook subscription was canceled if not intentional?

Example Payloads

This section will outline examples of payloads. This is not all inclusive and the aim is to provide an example of the events you can subscribe to. do an API call to retrieve the photo and update other systems like Active Directory.

All payloads contain similar fields. Here is a breakdown of those fields:

Field	**Description**
Event_ID	This is the ID of the webhook event. This is the unique identifier of an event. NOTE: It's important to only process an unique ID once in case duplicate messages somehow get received.
Event_Name	Friendly description of the event
Event_DateTime	UTC time of event.
ClientCode	Client code this event pertains to. This is useful if you are listening to events from multiple Paycom Client Instances.
Resource	
Resource_Identifier	
Object	
Object_Identifier	
Data	This changes depending on the webhook. Please review the below examples to get an idea of what this will contain.
Endpoint	Recommended API endpoint to use to gather information
EndpointUrl	Direct API link to data (does not include base API URL)

Note: Only a few employee change webhook samples are provided here to give the reader and idea of what those payloads look like.

Subscribed


```
{
   "Event_Id": "933846ac57b707daa7616239dc6d248d",
   "Event_Name": "Subscribed",
   "Event_DateTime": 1605907265,
   "ClientCode": "05510",
   "Resource_Field": null,
   "Resource_Identifier": null,
   "Object": null,
   "Object_Identifier": null,
   "Data": "{"Event_URL":15,"Subscribed_Event":"new_hire_created"}",
   "Endpoint": "",
   "EndpointUrl": ""
}
```


Verification URL


```
{
   "Event_Id": "72e5f632c3e86fe77d63124743969a8e",
   "Event_Name": "Verify URL",
   "Event_DateTime": 1605910379,
   "ClientCode": "05510",
   "Resource_Field": null,
   "Resource_Identifier": null,
   "Object": null,
   "Object_Identifier": null,
   "Data": "{"verification_code":"318873","expires_on":{"date":"2020-11-21
16:12:59.000000","timezone_type":3,"timezone":"America/Chicago"}}",
   "Endpoint": "",
   "EndpointUrl": ""
}
```


Ping


```
{
   "Event_Id": "d76adfae03b156698f9035cd88f8c4a9",
   "Event_Name": "Ping",
   "Event_DateTime": 1605910510,
   "ClientCode": "05510",
   "Resource_Field": null,
   "Resource_Identifier": null,
   "Object": null,
   "Object_Identifier": null,
   "Data": null,
   "Endpoint": "",
```

```
    "EndpointUrl": ""
}
```


New Hire Created


```
{
    "Event_Id": "ed9db3bf1640c8b6b1c99922ac6056f1",
    "Event_Name": "New Hire Created",
    "Event_DateTime": 1605926665,
    "ClientCode": "05510",
    "Resource_Field": "New Hire",
    "Resource_Identifier": null,
    "Object": null,
    "Object_Identifier": 75067,
    "Data": null,
    "Endpoint": "New Hire",
    "EndpointUrl": "api/v1/newhire/75067"
}
```


New Hire Deleted


```
{
    "Event_Id": "5b2b2db73ec375e806900f6a081e8bf5",
    "Event_Name": "New Hire Deleted",
    "Event_DateTime": 1605926747,
    "ClientCode": "05510",
    "Resource_Field": "New Hire",
    "Resource_Identifier": null,
    "Object": null,
    "Object_Identifier": 75067,
    "Data": null,
    "Endpoint": "New Hire",
    "EndpointUrl": "api/v1/newhire/75067"
}
```


New Hire Self On Boarding Completed


```
{
    "Event_Id": "34192b9a4b024c7e1415dcf479dcd869",
    "Event_Name": "New Hire Self Onboarding Completed",
    "Event_DateTime": 1605927634,
    "ClientCode": "05510",
    "Resource_Field": "Employee OnBoarding",
    "Resource_Identifier": null,
    "Object": null,
    "Object_Identifier": 75070,
```

```
  "Data": "{"Employee_ID":"Null"}",
  "Endpoint": "New Hire",
  "EndpointUrl": "api/v1/newhire/75070"
}
```


New Hire Self On Boarding Completed


```
{
  "Event_Id": "34192b9a4b024c7e1415dcf479dcd869",
  "Event_Name": "New Hire Self Onboarding Completed",
  "Event_DateTime": 1605927634,
  "ClientCode": "05510",
  "Resource_Field": "Employee OnBoarding",
  "Resource_Identifier": null,
  "Object": null,
  "Object_Identifier": 75070,
  "Data": "{"Employee_ID":"Null"}",
  "Endpoint": "New Hire",
  "EndpointUrl": "api/v1/newhire/75070"
}
```


Employee Added


```
{
  "Event_Id": "8e8e39a7e505d3f771ee97192af54a8b",
  "Event_Name": "Employee Created",
  "Event_DateTime": 1606750722,
  "ClientCode": "05510",
  "Resource_Field": "Employee",
  "Resource_Identifier": null,
  "Object": null,
  "Object_Identifier": "A00Y",
  "Data": "{"Employee_ID":"A00Y","Is_Rehire":"N","Was_New_Hire":"Y"}",
  "Endpoint": "Employee",
  "EndpointUrl": "api/v1/employee/A00Y"
}
```


Employee Photo Added


```
{
  "Event_Id": "764da13267e76c8e9254144622b1cb0d",
  "Event_Name": "Employee Photo Added",
  "Event_DateTime": 1605930665,
  "ClientCode": "05510",
  "Resource_Field": "Employee Photo",
  "Resource_Identifier": null,
```

```
  "Object": "Employee",
  "Object_Identifier": "A00X",
  "Data": null,
  "Endpoint": "Employee Photo",
  "EndpointUrl": "api/v1/employee/A00X/photo"
}
```


Employee Change (address change)


```
{
  "Event_Id": "19c30616a070fd1fd2915d1b92d065a9",
  "Event_Name": "Employee Changes",
  "Event_DateTime": 1605929535,
  "ClientCode": "05510",
  "Resource_Field": "Employee Address Changes",
  "Resource_Identifier": null,
  "Object": "Employee",
  "Object_Identifier": "A00X",
  "Data": null,
  "Endpoint": "Employee Changes",
  "EndpointUrl": "api/v1.1/employee/A00X/change/463q63vzql4"
}
```


Employee Change (nickname change)


```
{
  "Event_Id": "e5475748d21083a574d7c3d50924fd8e",
  "Event_Name": "Employee Changes",
  "Event_DateTime": 1605929681,
  "ClientCode": "05510",
  "Resource_Field": "Employee Name Changes",
  "Resource_Identifier": null,
  "Object": "Employee",
  "Object_Identifier": "A00X",
  "Data": null,
  "Endpoint": "Employee Changes",
  "EndpointUrl": "api/v1.1/employee/A00X/change/463q63vq3z4"
}
```


Unsubscribe


```
{
  "Event_Id": "56f325d8184aaa574f19c69158420f7c",
  "Event_Name": "UnSubscribed",
  "Event_DateTime": 1605907191,
  "ClientCode": "05510",
```

```
  "Resource_Field": null,
  "Resource_Identifier": null,
  "Object": null,
  "Object_Identifier": null,
  "Data": "{"Event_URL":15}",
  "Endpoint": "",
  "EndpointUrl": ""
}
```

Best Practices

- Upon receipt of the webhook, a response should be generated as soon as possible. Any processing of the data received should be handled separately in another process or added to a job queue.
- Your webhook endpoints might occasionally receive the same event more than once. We advise you to guard against duplicated event receipts by making your event processing idempotent. One way of doing this is logging the event IDs you have processed, and then not processing already-logged event IDs.
- Paycom does not guarantee delivery of events in the order in which they are generated. Your endpoint should not expect delivery of these events in this order and should handle this accordingly.
- It's recommended to set up a full sync process that runs daily in case any changes were missed. This is also useful for initial sync, or manually triggered syncs.

Sample Code

Example Webhook Controller Class Method for handling the webhook

NOTES: This is NOT intended to be a production ready script and is only used to illustrate the workflow to handle webhooks. You will want to consult your own development resources to build a production ready script for your environment.

```
<?php
//this is a sample PHP script to help illustrate webhook listening
//the key items:
//1.  Extract headers to pull out secret key
//2.  Verify Secret Key, return 401 if not authorized
//3.  Convert Headers and Body to JSON
//4.  Save header and body to database and log file
//            The idea here is a separate process will consume messages out of the
queue to prevent the listening process from ack'ing the receipt of message
//5.  Return 200 response code to ack the receipt

//database credentials
$dbhost = 'localhost';
$dbname = 'webhooks';
$dbusername = 'webhooks';
$dbpassword = 'w$bh00ksSampleP@ss123!';

//get headers to extract key
```

```
$headers = apache_request_headers();

//check for verification that this is indeed a message we want to add to our queue
//in this instance, we've configured the Paycom Webhook configuration to include
secret-key of 123
if(isset($headers['Secret-Key']) == false || $headers['Secret-Key'] != '123'){
	echo "Not Authorized";
	http_response_code(401);
	header("HTTP/1.1 401 Unauthorized");
	exit;
}

//get JSON format of header for DB insert
$json_header = json_encode($headers);

//let's take the event and extract the body
if($json = json_decode(file_get_contents("php://input"), true)) {
	$json_body = json_encode($json);
}else{
	$myfile = fopen("errors.log", "a") or die("Unable to open file!");
	fwrite($myfile, "\n".  "Error with message. Invalid JSON. Input:" . "\n" .
file_get_contents("php://input"));
	fclose($myfile);
	exit;
}

//connect to DB and insert event into events
//a seperate process will be picking up events and processing them
//this is to make sure events are not being blocked and a proper code is returned to
Paycom
try {
	$conn = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbusername, $dbpassword);
	$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

	$cmd = $conn->prepare("INSERT INTO events (header, body, time_received) VALUES
(?,?,?)");
	$cmd->execute(array($json_header,$json_body,date("Y-m-d H:i:s")));

} catch (PDOException $ex) {
	$myfile = fopen("errors.log", "a") or die("Unable to open file!");
	fwrite($myfile, "\n".  $ex->getMessage());
	fclose($myfile);
}

//save data to file, too
$myfile = fopen("receipt.txt", "a") or die("Unable to open file!");
fwrite($myfile, date("Y-m-d H:i:s"));
fwrite($myfile, "\nheader\n". $json_header);
fwrite($myfile, "\nbody\n". $json_body);
fwrite($myfile, " \n------------------------------------------------------------------------
---------- \n");
fclose($myfile);
//respond back with proper response code to let Paycom know we're done
http_response_code(200);
header("HTTP/1.1 200 OK");

?>
```