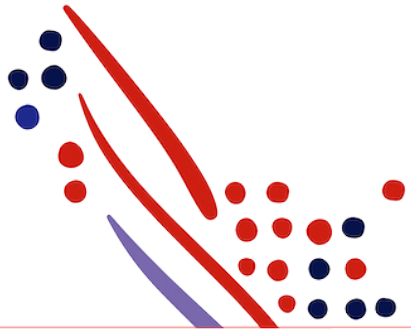
Guide

Webhooks

Published on
Sep 08, 2025, 07:07 PM

Last modified
Oct 31, 2025, 01:25 PM

Always Designing for People™



ADP Copyright Information

Published on
Sep 08, 2025, 07:07 PM

Published on
Oct 31, 2025, 01:25 PM

Table of Contents

Suggested Work Around

Chapter 1

ADP Event APIs and Event Notification Guide

Summary

Event notifications allow your application to be aware of data change events. They can also be used to retrieve the latest data to sync your system with ADP products and applications. When data is updated in an ADP application, ADP generates an event notification message. Then, the message is sent to a subscriber's message queue.

What's New

October 2025

- Added support for webhooks

About Events and Event Notifications

Event notifications provide details about the changes to an event subscriber's application. This prompts action by the partners and keeps their records updated.

For your application to receive these notifications, ADP offers two options:

- Regularly check and retrieve event notification messages from the queue (polling), or
- Have ADP send event notification messages directly to your application's endpoint (webhook).

Pre-requisites

Before you can receive event notifications, you must first subscribe to the specific data-change events the application requires. For example, subscribing to the "Worker Hire" event notifies you when a new hire is added to an ADP product, allowing your application to process the new hire's data.

Adding or requesting event notifications will vary by consumer type:

- **ADP API Central clients:** You may add event notifications to your existing projects by navigating to **Projects > Select a Project > APIs > Events > Add Events**.
- **ADP Marketplace partners:** If you're a Partner Self-Service user, reach out to your Marketplace Technical Advisor (MTA) to discuss the scope changes and add additional event notifications.

Polling workflow

#	Actor	Description
1	A human user or a system	Makes a change to a record. For example, an employee updates a worker's email address in an ADP product.
2	ADP product	Generates an event notification message and sends copies to a subscriber's message queue. Messages are queued based on the first in, first out (FIFO) method.
3	Consumer application	On scheduled time, retrieves one event notification message from its message queue and acts as needed.
4	Consumer application	Deletes the event notification message from its message queue and repeats steps 3 and 4 until there are no more messages in the queue.

Polling Data Dictionary References

1. The schema of event notifications varies depending on /events/eventNameCode/codeValue as well as the ADP product. Once you identify the interested event notification(s) to subscribe for your application, you should analyze the response payload and map out interested resources.
2. We recommend your application use event notifications as a trigger instead of the true source of real-time data, as there is always a time lag between when a notification was generated and when your application retrieves it. For example, if a notification indicates a worker is hired, your application should do GET /hr/v2/workers/{aoid} to retrieve the real-time data for the worker instead of relying on the value provided in the worker.hire notification, as the data may have changed since the notification was generated.
3. In regard to effective dating: Unless otherwise noted, when a change has an effective date, the notification is generated at the time when the change is issued, not on the effective date. Your application should be able to look up the effective date in the notification response payload.

Schema Location	Description	Note
/events/data/eventContext/worker/associateOID	Indicates the worker record being changed.	In most cases, the object in the **eventContext** is worker.
/events/eventNameCode/codeValue	Indicates the event which triggered this notification.	Your application should use this value to build logic on actions to take for each event.
/events/transform/	Specifies the object's new value.	Depends on the product and event. Your application may need to retrieve the record to get the current value, instead of leveraging the data within this component as there may be additional change events to the same record occurring from the time when the notification was generated.

Use Case 1: Retrieve an Event Notification

Use Case Description

When a change is completed for a worker in an ADP system, a notification is pushed to a message queue for you to query. Let's assume that your application subscribes to the "Worker Personal Communication Email Add" notification. When you add a new personal email for a worker, it creates a notification and pushes it to your message queue. After you retrieve and process the message, you must delete the notification to retrieve later notifications.

API Usage

Method	Uniform Resource Identifier (URI)	Description	Note
GET	/core/v1/event-notification-messages	Retrieves an Event Notification.	Refer to the specific product's ADP API data dictionary and API guide for a list of supported Event Notifications related to your use case.

Request Header Parameters

Parameter Name	Required (Y/N)	Usage	Value	Sample
Authorization	Y	Type of authorization	Bearer token	Authorization: Bearer xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Info

Each event notification message has a unique ID located in the **header of the response**, which is coded as **adp-msg-msgid**. It needs to be used for message deletion. Here is an example **adp-msg-msgid: 0x_414d51204253494e464f425136202020f2f4ca5503748720**

Responses

Response Code	Condition	Message Text	Tips to Handle
200 OK	Request is processed successfully.		
204 No Content	There are no messages in the queue.	empty body	There are no messages existing in the queue.
403 Forbidden	The request is not authorized.	empty body	Contact ADP support from within the API Central or Partner Self-Service help menu to add the notification to your application scope
404 Not Found		empty body	Contact ADP support from within the API Central or Partner Self-Service help menu.

Use Case 2: Deleting an Event Notification

API Usage

Method	URI	Desctiption
DELETE	/core/v1/event-notification-messages/{adp-msg-msgid}	Deletes an event notification message.

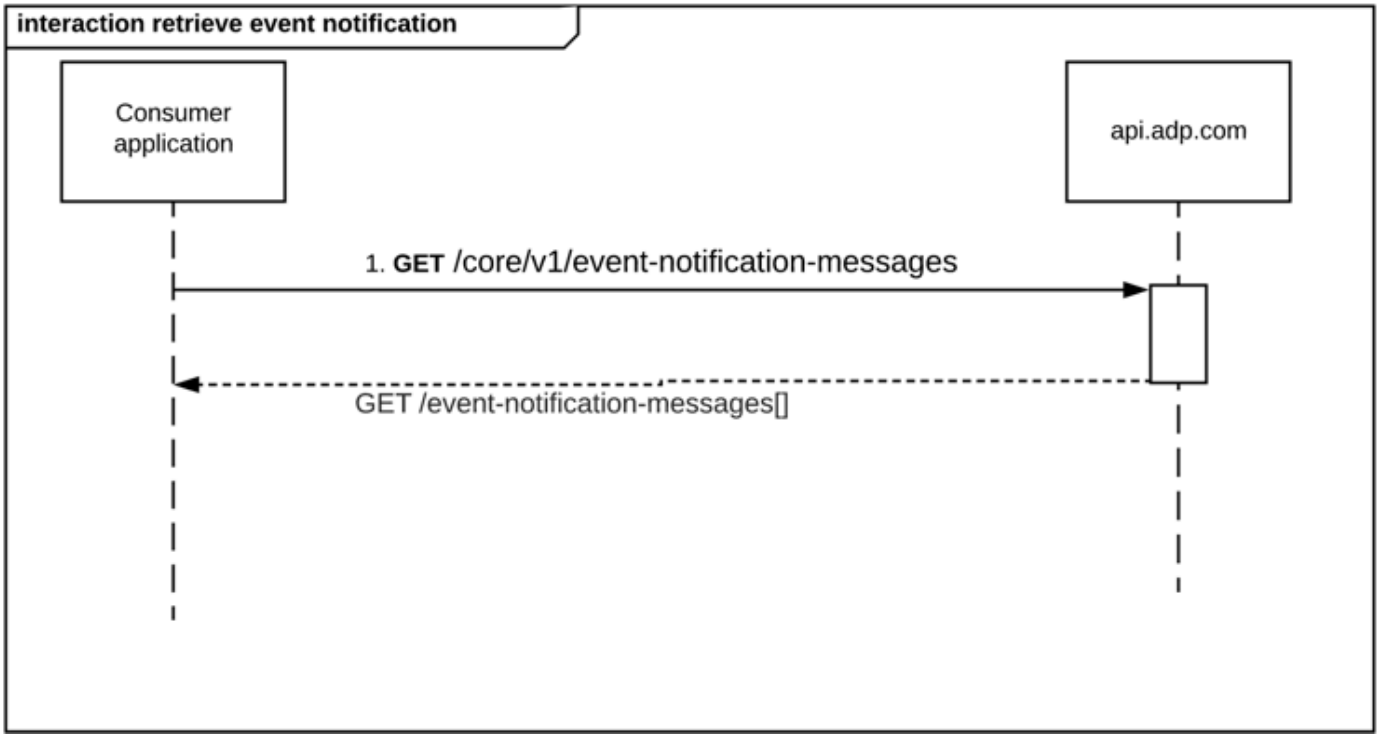
Info

The response header **adp-msg-msgid** is returned when retrieving an event notification. For example: **adp-msg-msgid: 0x_414d51204253494e464f425136202020f2f4ca5503748720**

Request Header Parameters

Parameter Name	Required (Y/N)	Usage	Value	Sample
Authorization	Y	Type of authorization	Bearer token	Authorization: Bearer xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Sequence of Interactions



Here the steps shown in the previous diagram:

1. Your consumer application makes a /core/v1/event-notification-messages/{adp-msg-msgid} request to the ADP API endpoint for a notification.
2. The ADP API endpoint responds to your consumer application with the events deleted.

Responses

Response Code	Condition	Message Text	Tips to Handle
200 OK	Request is processed successfully.		
404 Not Found	When trying to delete the same message again.	empty	The message is already deleted.
	When **invalid {adp-msg-msgid}** is passed in the request.	empty	Make sure a valid {adp-msg-msgid} is passed in the request. **NOTE: {adp-msg-msgid}** is obtained from the response headers of the GET /core/v1/event-notification-messages/ call.

Webhook workflow

With the webhook method, ADP monitors the message queue and sends event notifications directly to your application's endpoint. This eliminates the need for you to poll the ADP endpoint for new notifications.

#	Actor	Description
1	A human user or a system	Makes a change to a record. For example, an employee updates a worker's email address in an ADP product.
2	ADP product	Generates an event notification message and sends copies to a subscriber's message queue. Messages are queued

#	Actor	Description
		based on the first-in, first-out (FIFO) method.
3	ADP webhook system	ADP's webhook system will authenticate with the destination system and transmit the event moments after being placed inside the message queue

4	Consumer application	Responds to the webhook message with an HTTP 2xx status code (200,201, or 202) to acknowledge successful receipt.
		Example:
		{
		"status": "success",
		"timestamp": "YYYY-MM-DDTHH:MM:SSZ" //current timestamp UTC
		}

Prerequisites for Webhook Configuration

1. You must have a publicly accessible U.S endpoint hosted on your infrastructure
2. Your endpoint must be secured using one of the following authentication types.Below are curl examples of the client implementation. Please ensure your server-side implementation will support the following client requests.

Supported Authentication Models

API Key

To utilize an API key endpoint to receive event notification messages, you will need to provide your:

- Webhook API endpoint
- API key

```
curl --location 'https://partnerwebhook.com/apikey/event' \
 \ --header 'Content-Type: application/json' \
 \ --header 'Authorization: Bearer valid_api_key' \
 \ --data '{ }' \
```

asic Authentication

To utilize a basic authentication endpoint to receive event notification messages, you will need to provide your:

- Webhook API Endpoint
- Username
- Password

```
curl --location 'https://partnerwebhook.com/event' \
 \ --header 'Content-Type: application/json' \
 \ --header 'Authorization: Basic #encoded' \
 \ --data '{ }' \
```

Note

Note: #encoded - This will be base 64 encoded user name and password given by the user in the format - username:password

OAuth 2.0

To use an OAuth 2.0 endpoint to receive event notifications, you must provide your:

- Webhook API endpoint
- Access token URL
- Client ID
- Client secret
- OAuth 2.0 scopes.

OAauth 2.0 Token Call

```
curl --location 'https://partnerwebhook.com/token' \
 \ --data \
-urlencode 'client_id=valid_client_id' \ --data \
-urlencode 'client_secret=valid_client_secret' \ --data \
-urlencode 'grant_type=client_credentials' \ --data \
-urlencode 'scope=valid_scope'
```

Webhook Call

```
curl --location 'https://partnerwebhook.com/oauth2/event' \
 \ --header 'Content-Type: application/json' \
 \ --header 'Authorization: Bearer valid_bearer_token' \
 \ --data '{ }' \
```

Required Response

Upon receiving an event, your application must respond with a successful acknowledgement:

- HTTP 2xx status code (200,201, or 202)
- A response body with the following structure

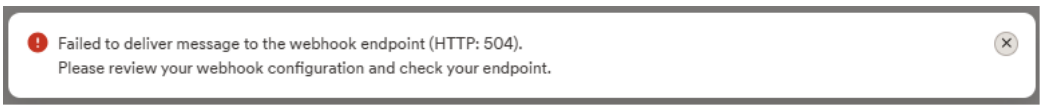
```
{
  "status": "success",
  "timestamp": "YYYY-MM-DDTHH:MM:SSZ"
}
```

> **Info**
>
> Failure to reply with a successful acknowledgement system will mark the attempt as failed and begin the retry mechanism.

Testing Webhook Configuration

Before enabling a webhook, you can — and should — test the connection to your endpoint. To send a test message, save the webhook configuration and select **Test webhook**.

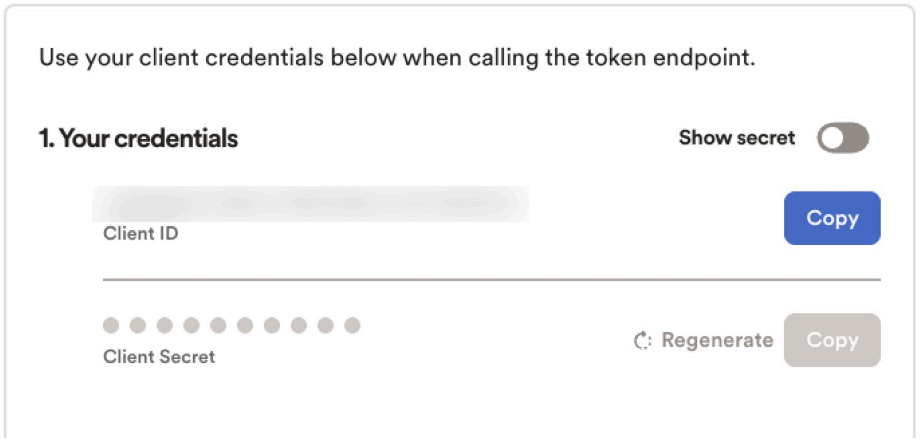
API Central or Partner Self-Service will display a success or failure notification. If the test fails, analyze the response message, make the necessary updates and retest the webhook.



When the test is successful, select **Save webhook** to update your webhook configuration.

Signature Verification (Header)

With each event, ADP sends a signature to verify that the message originated from us. This header is uses the key "adpx-messageauthentication" and is an HMAC-SHA256 hash using your data connector client secret as the secret key and data connector client ID as the message. You can obtain the data connector client ID and client secret by navigating to **Project Details** > **Credentials** in API Central or Partner Self-Service.



Example header value:

- adpx-messageauthentication 2269190bb875ee6f9de922d8e2d50d20fa48c7e999aecbed6b157cc810e7c242
  - Example Publisher Client ID: 8f8e6d8d-8f10-4c51-a308-bd0d1323808d
  - Example Publisher Client Secret: b269737b-f4fe-40fc-bce5-263bc9196011

Webhook Data Dictionary References

In the webhook delivery mechanism, we modify events you would get from the polling workflow to provide additional metadata wrapped around the polling event. This will give you more context about where the event originates from and what it's about.

Schema Location	Description	Example	Note
/meta/messageId	MessageID from message queue	0x:414d51204253494e4647f425131322020f7d5e76603575b4c	This will be unique per event
/meta/orgID	ADP Organization ID the event belongs	ABCDEFGHIJKLMNOP	
/meta/messageSentTime	UTC format of when this message was sent from ADP Marketplace to the Webhook	YYYY-MM-DDTHH:MM:SS.SSSZ	
/meta/attempts	Number of attempts made to deliver this message	1	
/meta/consumerApplicationID	The application or project ID associated with this event.	d6636414-6932-441f-bb7d-7561f645993c	
/meta/queueName	Message queue name	ADP.MP.164401E1E7234CB9A46AECB91BB3F51E	
/meta/webhookAuditTrailID	This is the transaction level ID of the attempted message.	6ff83d0c-4759-4699-a94e-bc604bfeaae8Rc_1	If the same message failed to be sent and was attempted again there would be a different webhookAuditTrailID generated

Webhook Failure Notifications

Once your webhook is successfully configured, ADP will begin monitoring messages sent to your endpoint. When ADP sends an event notification to a webhook endpoint and receives an error or no response, ADP will send a failure notification email to all registered users.

Tip

For failure notification emails to be sent, several ADP backend systems must be updated, which does not happen immediately. Therefore, it may take up to an hour for you to receive a failure notification. This delay pertains only to the failure notification emails, not webhook functionality.

Registering for Webhook Failure Notifications

To receive notifications when your webhook fails, log into API Central or Partner Self-Service, select the notification bell in the main menu bar, and save the new configuration with webhook failure alerts enabled.

Info

Failure notification is an opt-in feature. Users must log in and save the notification settings with webhook failure alerts enabled. The creator of the project will have this notification enabled by default.

Failure Retry Logic

If an event delivery fails, it will be retried 10 times in one-minute intervals. If the delivery still fails, 10 additional attempts will be made to deliver the event every six hours. New event notifications will be attempted, and when an event is successfully delivered, previously failed events will be reattempted.

Info

Failed events will be stored for five days

Known Issues and Limitation

US1191638: Ability to perform bulk retrieval and deletion

Issue Description

Currently, bulk retrieval and deletion of events are not supported.

Suggested Work Around

One event is retrieved and deleted at a time. In order to retrieve the next event, clients have to delete the current event using the /core/v1/event-notification-messages/ {adp-msg-msgid}. Then, they need do a GET /core/v1/event-notification-messages.