Chapter 1

# Event notifications

from ADP Marketplace ESI - Event notifications

Published on
Apr 04, 2022, 11:53 AM

Last modified
Feb 11, 2026, 01:48 PM

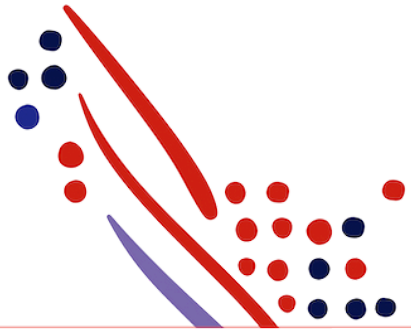
Always Designing for People™

# ADP Copyright Information



Published on
Apr 04, 2022, 11:53 AM

Published on
Feb 11, 2026, 01:48 PM



# Chapter Contents


Chapter 1

**ADP Marketplace ESI - Event notifications**

- About event notifications
- Required setup steps
- Data dictionary
  - Fields in the event notification list
  - Common fields in the event notifications
- Process Overview

Chapter 1

# Event notifications

Event Notifications allow your application to be notified of data changes. They can be used to retrieve the latest data, to sync your system with ADP products and applications. When data is updated in an ADP application, that application generates an Event Notification message. This message is stored in the Event Manager database, waiting for a subscribed consumer.

Examples:

- Creating/disabling a user in the facilities ticketing system when the employee is joining or leaving the company
- Update the name of supervisor registered in a purchasing system when the employee transfers to another department
- Register a vacation day in a time sheet application when the request for leave is approved.

Event notifications give you the opportunity to work with small amounts of transaction data without the need to use batch processing.

## About event notifications

ADP Representational State Transfer (REST) APIs use an event-based pattern for resource modification. Clients or partners need to know if there are any changes in the System of Record (SOR) at ADP to act upon or keep themselves aware . Events mark every change made in the system. Partners or clients will get to know about the changes after they subscribe to the specific changes for which they want to be notified. Event Notifications provide the details about the changes to partners. This prompts action by the partners and keeps their records updated.

### Note

Event notifications are used in near real-time scenarios and it's recommended to check the event notification list on scheduled basis with at least a maximum of one hour.

## Required setup steps

To subscribe to an event notification, contact your ADP Representative.

## Data dictionary

This dictionary describes the fields you will typically find in the event notifications list and in a single event. For SOR specific examples, you can check the SOR documentation.

## Fields in the event notification list

These are the field you can find in the list of notifications by doing a request to /core/v1/event- notification-messages. The events in that list need to be act upon by calling them with the callback.

| Resource location | Description | Sample |
|---|---|---|
| /events | | <pre>{<br>  "events": []<br>}</pre> |
| /events/eventID | Indicates the unique identifier of the event instance. This is set by the system of record (SOR) after an event is recorded as in progress or complete. | <pre>{<br>  "events": [<br>  {<br>  "eventID": "12456-3445-22424"<br>  }<br>  ]<br>}</pre> |

| Resource location | Description | Sample |
|---|---|---|
| | | |
| /events/eventNameCode | Indicates the canonical name of the event. For example, dependent.add and worker.hire. This field is always present and valued based on a standard codelist. | {<br>"events": [<br>{<br>"eventNameCode": {<br>"codeValue": "worker.hire"<br>}<br>}<br>]<br>} |
| /events/recordDateTime | Specifies the date and time the event is recorded in the SOR with an event status code equal to complete. This value is set by the SOR. | {<br>"events": [<br>{<br>"recordDateTime": "2022-02-28T11:09:46Z"<br>}<br>]<br>} |
| /events/creationDateTime | Specifies the date and time the event is created. Necessary for the order in which events are consumed. | {<br>"events": [<br>{<br>"creationDateTime": "2022-02-28T11:10:22.005Z"<br>}<br>]<br>} |
| /events/originator | Indicates an originator that triggered the event. This can be either a user, machine or event. | {<br>"events": [<br>{<br>"originator": {<br>"associateOID": "ADPNL1234567"<br>}<br>}<br>]<br>} |
| /events/links | The callback uri you need to call to get the event details; the actual data | {<br>"events": {<br>"links"<br>[<br>{<br>"rel": "full",<br>"href": "/events/hr/v1/worker.hire/5f3d1d4895d89c19d8ae3034"<br>}<br>]<br>}<br>} |

## Common fields in the event notifications

These are the fields you will find when you GET a single event notification through its callback url found in the event notification list. It contains all the business information.

| Resource location | Description | Sample |
|---|---|---|
| /events/data/ output | Specifies the event result. | {<br>"events": [<br>{<br>"data": {<br>"eventContext": {<br>"worker": {<br>"associateOID": "G3Q7JN6K3CNJJRQH "<br>}<br>},<br>"output": {<br>"worker": {<br>"businessCommunication": {<br>"mobile": {<br>"itemID": "40439828_3139",<br>"nameCode": {<br>"codeValue": "Work Cell",<br>"shortName": "Work Cell"<br>},<br>"countryDialing": "1",<br>"areaDialing": "973",<br>"dialNumber": "7133458",<br>"access": "1",<br>"formattedNumber": "(973) 713- 3458"<br>}<br>},<br>"associateOID": "G3Q7JN6K3CNJJRQH ",<br>"workerID": {<br>"idValue": "0NHSXHM4I"<br>}<br>}<br>}<br>}<br>}<br>}<br>]<br>} |
| /events/data/ eventContext | Specifies the data which sets the context for the event. That is, the keys which set context. | {<br>"events": [<br>{<br>"data": {<br>"eventContext": {<br>"worker": {<br>"associateOID": "ADP1234567"<br>}<br>}<br>}<br>}<br>}<br>]<br>} |

## Process Overview

When data is updated in an ADP application, the application generates an event notification message. That message is stored in the event manager database. To regularly check and retrieve event notification messages you can subscribe your application to data change events. For example, you can subscribe to the worker.hire event to be notified when a new hire is added to an ADP product, so you can do relevant work with the new hire's data.

| | 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 stored based on the first in, first out (FIFO) method. |

| | Actor | Description |
|---|---|---|
| 3 | Consumer application | On scheduled time, retrieves the list of Event Notifications. |
| 4 | Consumer application | Performs a callback on the first Event Notification in the list and acts on the received data |
| 5 | Consumer application | Deletes the Event Notification message from its message queue and repeats step 4 until the list is exhausted. |