Event and Event Notification Guide
Contents

- Overview
  - Supported Product Version and Customer Base
  - About Events and Event Notifications
  - Process Overview
  - Required Setup Steps
  - Data Dictionary References

- Use Cases
  - Use Case 1: Retrieve an Event Notification
    - Description
    - API Usage
    - Request Header Parameters
    - Actors
    - Sequence of Interactions
    - Responses
  - Use Case 2: Delete an Event Notification
    - Description
    - API Usage
    - Request Header Parameters
    - Actors
    - Sequence of Interactions
    - Responses

- Known Issues and Limitations
  - US1191638: Ability to Bulk Retrieval and Deletion
    - Description
    - Suggested Work Around
Overview

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. To regularly check and retrieve event notification messages, subscribe your application to data change events.

Supported Product Version and Customer Base

For more information, see product specific data dictionaries and Application Programming Interface (API) Guides for supported products.

About Events and 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 in order 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.

Process Overview

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. To regularly check and retrieve event notification messages, 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. This is so you can do some work with the new hire’s data.

<table>
<thead>
<tr>
<th>Actor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 A human user or a system</td>
<td>Makes a change to a record. For example, an employee updates a worker's email address in an ADP product.</td>
</tr>
</tbody>
</table>
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.

On scheduled time, retrieves one event notification message from its message queue and takes action as needed.

Deletes the event notification message from its message queue and repeats steps 3 and 4 until there are no more messages in the queue.

**Required Setup Steps**

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

**Data Dictionary References**

Resources listed in the following table can be accessed through

<table>
<thead>
<tr>
<th>Schema Location</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>/events/data/eventContext/worker/associateOID</td>
<td>Indicates the worker record being changed.</td>
<td>In most cases, the object in the eventContext is worker.</td>
</tr>
<tr>
<td>/events/eventNameCode/codeValue</td>
<td>Indicates the event which triggered this notification.</td>
<td>Your application should use this value to build logic on actions to take for each event.</td>
</tr>
</tbody>
</table>
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 Cases

Refer to the following table to find the use case aligning with your business needs.

<table>
<thead>
<tr>
<th>#</th>
<th>USE CASE NAME</th>
<th>ADP SYSTEM(S) SUPPORT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Vantage</td>
</tr>
<tr>
<td>1</td>
<td>Retrieve an event notification</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Delete an event notification</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Use Case 1: Retrieve an Event Notification

Description

When you complete a change 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 retrieving and processing the message, you must delete the notification to retrieve later notifications.

API Usage

<table>
<thead>
<tr>
<th>Method</th>
<th>URI</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/core/v1/event-notification-messages</td>
<td>Retrieves an event notification.</td>
<td>Refer to the specific product's ADP API data dictionary and API Guide for a list of supported event notifications relate to your use case.</td>
</tr>
</tbody>
</table>
Request Header Parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
<th>Required</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Bearer {token}</td>
<td>Yes</td>
<td>Type of the authorization.</td>
</tr>
<tr>
<td>prefer</td>
<td>/adp/long-polling</td>
<td>No</td>
<td>Server delays response for certain time.</td>
</tr>
</tbody>
</table>

**Note:**
Each event notification message has a unique ID located in the header of the response coded as adp-msg-msgid. It needs to be used for message deletion.

Prefer Header Usage

- If header is **included** while making a get event notification call, the server delays its response, waiting for events to display on the respective subscriber’s event notification queue. When no event notifications are available for a given time period, a timeout occurs. The default waiting time is 15 seconds.
- If prefer header is **not included** while making get event notification call, the server will not wait for any events to display on the queue. It just responds with whatever is present in the queue.

**Note:**
It is always recommended to use **prefer:/adp/long-polling** while making get event notification calls in order to get an accurate response from the queue.
**Actors**

ADP APIs enforce data entitlement and access control. The actors listed on the following table are supported for this use case.

<table>
<thead>
<tr>
<th>Request Parameter</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>roleCode value</td>
<td></td>
</tr>
<tr>
<td>practitioner</td>
<td>Retrieves event notifications as a practitioner. A practitioner can usually access information for a broader collection of workers.</td>
</tr>
</tbody>
</table>
Sequence of Interactions

The following are the steps shown in the previous diagram:

1. Your consumer application makes a /core/v1/event-notification-messages request to the ADP API Endpoint for a notification.
2. The ADP API Endpoint responds with the events to your consumer application.

Responses

<table>
<thead>
<tr>
<th>Response Code</th>
<th>Condition</th>
<th>messageCode</th>
<th>messageText</th>
<th>Tips to Handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Request is processed successfully.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>204</td>
<td>When there are no messages in the queue.</td>
<td>204 No Content</td>
<td>empty body</td>
<td>There are no messages existing in the queue.</td>
</tr>
<tr>
<td>304</td>
<td>When performing a GET without first deleting a previous read message.</td>
<td>304 Not Modified</td>
<td>empty body</td>
<td></td>
</tr>
<tr>
<td>Status Code</td>
<td>Description</td>
<td>Response Code</td>
<td>Body Content</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>403</td>
<td>The request is not authorized.</td>
<td>403 Forbidden</td>
<td>empty body</td>
<td></td>
</tr>
<tr>
<td>404</td>
<td>When no message is created in the queue.</td>
<td>404</td>
<td>empty body</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
The 404 response remains only until the first notification is entered into the queue. After that notification is deleted and no other notifications exist, the response returned is a **204 – No Content**.

**Use Case 2: Delete an Event Notification**

**Description**

Delete an event notification message.

**API Usage**
### Method
<table>
<thead>
<tr>
<th>Method</th>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>/core/v1/event-notification-messages/{adp-msg-msgid}</td>
<td>Deletes an event notification message.</td>
</tr>
</tbody>
</table>

**Note:**
Response header 'adp-msg-msgid' is returned when retrieving an event notification.

### Example:
```
adp-msg-msgid: 0x_414d51204253494e464f425136202020f2f4ca5503748720
```

### Request Header Parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Required (Y/N)</th>
<th>Usage</th>
<th>Value</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Y</td>
<td>Type of authorization</td>
<td>Bearer token</td>
<td>Authorization: Bearer xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx</td>
</tr>
</tbody>
</table>

### Actors

ADP APIs enforce data entitlement and access control. The actors in the following table are supported for this use case.

<table>
<thead>
<tr>
<th>Request Parameter roleCode value</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>practitioner</td>
<td>Deletes an event notification as a practitioner. A practitioner can usually access information for a broader collection of workers.</td>
</tr>
</tbody>
</table>
Sequence of Interactions

The following are 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

<table>
<thead>
<tr>
<th>Response Code</th>
<th>Condition</th>
<th>messageCode</th>
<th>messageText</th>
<th>Tips to Handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Request is processed successfully.</td>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>404</td>
<td>When trying to delete the same message again.</td>
<td>404 Not Found</td>
<td>empty</td>
<td>The message is already deleted.</td>
</tr>
<tr>
<td></td>
<td>When invalid {adp-msg-msgid} is passed in the request.</td>
<td>404 Not Found</td>
<td>empty</td>
<td>Make sure a valid {adp-msg-msgid} is passed in the request.</td>
</tr>
</tbody>
</table>

**Note:**

{adp-msg-msgid} is obtained from the response headers of GET `/core/v1/event-notification-messages` call.

---

### Known Issues and Limitations

**US1191638: Ability to Bulk Retrieval and Deletion**

Ability to retrieve and delete notification messages in bulk.

**Description**

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

Currently, bulk retrieval is not supported. 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.