



Guide

ADP Marketplace Integration - Separate Credentials (OAuth2)

Published on
Jul 22, 2021 4:08PM

Last modified
Mar 30, 2023 9:53AM





ADP Copyright Information

ADP, the ADP logo, and Always Designing for People are trademarks of ADP, Inc.

Windows is a registered trademark of the Microsoft Corporation.

All other trademarks are the property of their respective owners.

Copyright © 2023 ADP, Inc. ADP Proprietary and Confidential - All Rights Reserved. These materials may not be reproduced in any format without the express written permission of ADP, Inc.

These materials may not be reproduced in any format without the express written permission of ADP, Inc. ADP provides this publication "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. ADP is not responsible for any technical inaccuracies or typographical errors which may be contained in this publication. Changes are periodically made to the information herein, and such changes will be incorporated in new editions of this publication. ADP may make improvements and/or changes in the product and/or the programs described in this publication.

Published on
Jul 22, 2021 4:08PM

Last modified
Mar 30, 2023 9:53AM

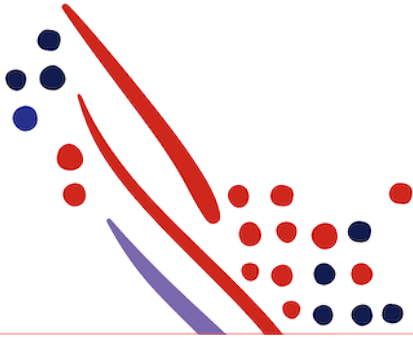


Table of Contents

Chapter 1

Overview

Chapter 2

Steps for Implementing Subscription Endpoints

Chapter 3

Synchronous Subscription Events Workflow

Chapter 4

Asynchronous Subscription Event Workflow

Asynchronous Subscription Event workflow

Chapter 5

Steps for implementation of user assignment events

Chapter 6

Testing Your ADP Marketplace Integration

Overview

ADP Marketplace provides a few different methods of authentication as it pertains to subscription events. In this article we explain the steps required when using separate credentials which leverages OAuth2. This method will **require** that your application endpoints are protected behind an OAuth2 server

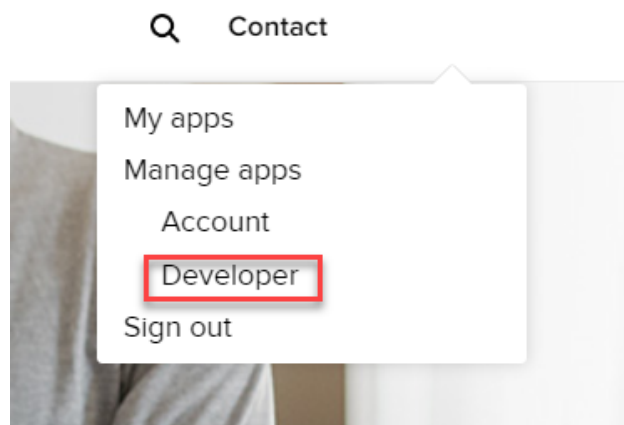


Info

Do not use the "Ping Test" option for testing endpoints. It doesn't work properly in some networks and cannot be relied upon for completion of the development and testing. Please rely on the "Integration Reports" for testing endpoints

Steps for Implementing Subscription Endpoints

1. Log into ADP Marketplace and access the developer menu under your profile.




2. If you haven't already created an app listing, create the app listing using the template provided by ADP.
3. Access the menu Products -> {your product listing} -> Edit.
4. From the menu on the left sidebar, navigate to Integration -> Edit Integration.
5. Now, navigate to the "Subscriptions" section under Edit Integration. Developers must register URL endpoints to handle notifications for subscription create, subscription change and subscription cancel. Make sure to append your URL Endpoint with "?eventUrl={eventUrl}", if you do not you will never get the event URL to call later. Sample URLs:
Subscription create - <https://example.com/adpsubscription/create?eventUrl={eventUrl}>
Subscription cancel - <https://example.com/adpsubscription/cancel?eventUrl={eventUrl}>
Subscription change - <https://example.com/adpsubscription/change?eventUrl={eventUrl}>
Subscription status - <https://example.com/adpsubscription/status?eventUrl={eventUrl}>

Edit Integration


English (United States)

SUBSCRIPTIONS


Subscription Create Notification URL 


This is an interactive endpoint

[Add Required Info](#)

Subscription Change Notification URL 

Additional product settings required during checkout

Subscription Cancel Notification URL 

Subscription Status Notification URL 

Enable upcoming invoice notification



Info

Do NOT select "This is an Interactive Endpoint", this option is not supported.

6. Navigate to the "**Credentials**" section under "**Edit Integration**", and ensure you are selecting "**Separate Credentials**" for the Authorization type. This will configure your listing to utilize OAuth2.0 client credentials grant type flow in order for you to retrieve the subscription event payload from the eventUrl.
7. Generate a client ID and Secret in the "**Inbound Credentials**" section.
8. Under the "**Outbound Credentials**" section select **OAuth2 (client credentials)** as the type. Enter in your OAuth2 server's token endpoint along with client id, client secret, and scope that was generated for ADP Marketplace to authenticate.

Credentials

Authorization type

Separate Credentials

Shared Credentials (Legacy)

Separate Credentials

type uses distinct OAuth 2.0 client keys and secrets to secure distribution event notifications and integration. We recommend that you create products using this model, and that you migrate existing products to this model.

INBOUND CREDENTIALS (OAUTH 2.0)

Use the client ID and secret to request an OAuth 2.0 access token that can be used to secure incoming API requests for your product integration (for example, to retrieve details of an event).

Generate ID and Secret

Go To Technical Doc →

OUTBOUND CREDENTIALS

Define the credentials configuration used to secure event notifications sent to your configured distribution endpoints.

Type

OAuth2 (client credentials)

Token endpoint

https://partnerapplication.com/oauth/token

Endpoint used to request an OAuth 2.0 access token

Client ID

11

Client identifier used to request an OAuth 2.0 access token from the configured token endpoint

Client secret

Client secret used to request an OAuth 2.0 access token from the configured token endpoint

Scope (Optional)

A space-separated list of scope values associated with the requested OAuth 2.0 access token. Leave blank if not needed.

Save

Go To Technical Doc →

9. There are different subscription events, such as create, update, cancel and status notifications. However, they all follow the same mechanism in terms of how they notify your application.

10. Subscription events should be developed as synchronous events. (ADP Marketplace should receive the response immediately from your application **after you have consumed the payload in the eventUrl**)

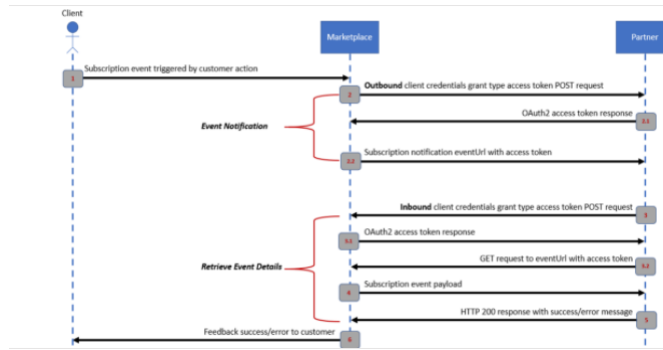
Chapter 3

Synchronous Subscription Events Workflow

Use the appropriate domain depending on which Marketplace you are developing for:

ADP Marketplace U.S. Integration (<https://apps.adp.com>)

ADP Marketplace Canada Integration (<https://ca.apps.adp.com>)



+

1. An event is triggered by a customer action (e.g., subscribing to an application). This creates a SUBSCRIPTION_ORDER event identified by a URL, e.g., <https://apps.adp.com/api/integration/v1/events/12345>.
2. ADP Marketplace will authenticate to your OAuth2 server's token endpoint and make a POST request using the client credentials grant type flow
 1. Your OAuth2 server should respond back with access token
 2. ADP Marketplace will utilize the access token to authenticate to the respective subscription event endpoint via a GET request and will providing you with the eventUrl which has the subscription details. As an example, the request will look like the following (GET <https://partnerapplication.com/subscription/create?eventUrl=https://apps.adp.com/api/integration/v1/events/12345>)
- Partner application initiates a POST request to ADP Marketplace's token endpoint:
 1. US Marketplace = <https://apps.adp.com/oauth2/token>
Canada Marketplace = <https://ca.apps.adp.com/oauth2/token>.
 2. The request must be authenticated via basic authentication using the generated **Inbound** Client ID and Client Secret

Header	Query Parameter	Description
	grant_type	REQUIRED. The grant_type parameter must be set to the value client_credentials .
	scope	REQUIRED. The scope parameter must be set to ROLE_APPLICATION
Authorization		REQUIRED. The Authorization header will authenticate with the Inbound Credentials client id and client secret with basic authentication.
Accept		REQUIRED (one of the following). The Accept header must be set to the value application/json or application/xml depending on which format your application is expecting.
Content-Type		REQUIRED (one of the following). The Content-Type header must be set to the value application/json or application/xml depending on which format your application is sending.

3. ADP Marketplace calls the Outbound Credentials endpoint to retrieve an access_token from your application
4. Partner application initiates a GET request to the eventUrl provided from the subscription notification, including sending the access token in the Authorization: Bearer token header
By default eventUrl response will be in XML, utilize the Accept: application/json header if you wish to receive a JSON response. The partner application must not respond to the initial subscription notification until after a response is received from the eventUrl.

5. The subscription payload response would look something like the following

```
{
  "type": "SUBSCRIPTION_ORDER",
  "marketplace": {
    "partner": "ADP",
    "baseUrl": "https://apps.adp.com"
  },
  "applicationUuid": null,
  "flag": "DEVELOPMENT",
  "creator": {
    "uuid": "30a0c21c-e5f7-451d-a16f-149eac7061c2",
    "openId": "https://apps.adp.com/openid/id/30a0c21c-
e5f7-451d-a16f-149eac7061c2",
    "email": "John.Doe@adp.com",
    "firstName": "John",
    "lastName": "Doe",
    "language": "en",
    "locale": "en-US",
    "address": null,
    "attributes": null
  },
  "payload": {
    "user": null,
    "company": {
      "uuid": "699da487-d17c-403e-90b7-aac7fb3e1d48",
      "externalId": "FFFFFFFFFFFFFFFF",
      "name": "ADP",
      "email": null,
      "phoneNumber": null,
      "website": null,
      "country": "US"
    },
    "account": null,
    "addonInstance": null,
    "addonBinding": null,
    "order": {
      "editionCode": "Trial + Monthly",
      "addonOfferingCode": null,
      "pricingDuration": "MONTHLY",
      "items": [],
      "customAttributes": []
    },
    "notice": null,
    "configuration": {
      "organizationOID": "FFFFFFFFFFFFFFFF",
      "applicationID": null,
      "associateOID": "G3G898NGDZCHEW8G"
    }
  },
  "returnUrl": null,
  "links": [
    {
      "rel": "oidcClient",
      "href":
      "https://apps.adp.com/api/account/v2/subscriptions/fa51b3a6-
6198-4ed4-bf59-b5e418c43b0d/oidcClient"
    }
  ]
}
```

For more subscription related event examples please click [Here](#) to download.

6. Your application creates/updates account information for the subscriber details. Then, your application returns a JSON response back to the Marketplace event, and passes the following values depending on the below scenarios. The two variants of the status notifications are "success":true, or "success": "false". Here are sample JSON response documents:

Success


```
{
```

```

"accountIdentifier": "<OrganizationOID>",
"success": true
}

False
{
"success": "false",
"errorCode": "USER_ALREADY_EXISTS",
"message": "Optional message about the user already existing on ISV"
}

```



Info

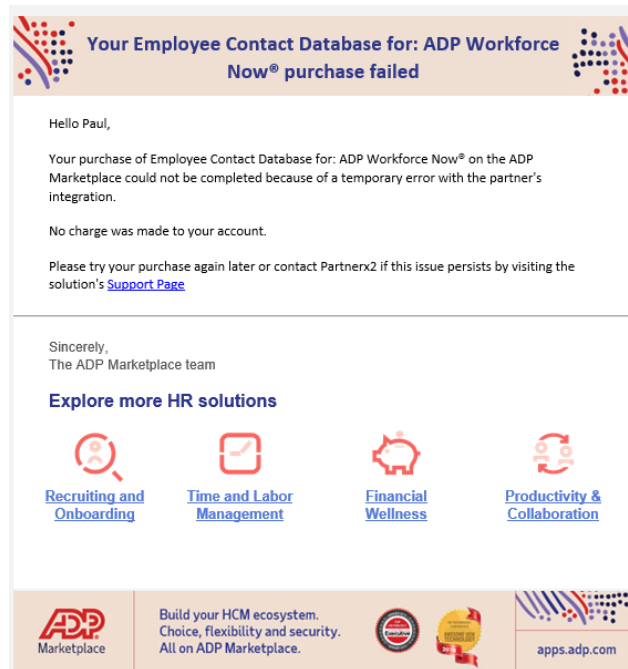
The partner application should always return a 200 status with properly formatted JSON or XML as mentioned above. **Do not return a 500 or 404 status.**

Note: Once you have responded to the ADP Marketplace with either of the 200 Status response examples above you will no longer be able to pull the event payload. Please ensure you have consumed the event payload before responding to the ADP Marketplace.

In the event of an error, partner application should return only the following predefined error codes listed **here**.

7. If a failure is sent the client will be notified via email as well as you the partner. Below are examples of the partner facing failure email as well as the client facing failure email.

Client Facing Email:



Your Employee Contact Database for: ADP Workforce Now® purchase failed

Hello Paul,





Your purchase of Employee Contact Database for: ADP Workforce Now® on the ADP Marketplace could not be completed because of a temporary error with the partner's integration.


No charge was made to your account.

Please try your purchase again later or contact Partnerx2 if this issue persists by visiting the solution's [Support Page](#)

Sincerely,
The ADP Marketplace team

Explore more HR solutions

-  [Recruiting and Onboarding](#)
-  [Time and Labor Management](#)
-  [Financial Wellness](#)
-  [Productivity & Collaboration](#)

ADP Marketplace | Build your HCM ecosystem. Choice, flexibility and security. All on ADP Marketplace. |  | [apps.adp.com](#)

Partner Facing Email (will be enabled End of May 2022):



Info

The partner facing email will be triggered of when:

1. You send a 200 success = false message(like the one shown above)
2. If your endpoint is down

It will contain links to our documentation on how to test your endpoint using the "**Integration Report**" as well as a link to put a ticket in if you need additional assistance.

In the event of an error, partner application should return only the following predefined error codes listed [here](#).

8. The return status information from the above step is used by ADP Marketplace to provide feedback to the customer via the ADP Marketplace UI.

- **Subscription cancel notification:** When a buyer cancels the subscription, your application will receive a notification (JSON message) through the registered URL. This means the client no longer wants to use the application. You should develop code to take the necessary actions as per your policy when you receive the cancel notification.
- You are required to implement a **subscription change** notification and **subscription status** notification as well. The technology and flow are the same as the implementation of the subscription create and subscription cancellation.

Chapter 4

Asynchronous Subscription Event Workflow

The synchronous subscription event workflow, described in the previous chapter, is the recommended approach when building ADP Marketplace solutions. This chapter will go over the **Asynchronous Subscription Workflow** where you can purposely put a subscription event into a pending state if you have a use case for this flow. This could be used for example on your core listing if it will take time for you to provision an instance for a new client (example 24 hours). Please follow the steps below:

Asynchronous Subscription Event workflow

1. An event is triggered by a customer action (e.g., subscribing to an application). This creates a SUBSCRIPTION_ORDER event identified by a URL, e.g., <https://apps.adp.com/api/integration/v1/events/12345>.
2. ADP Marketplace will authenticate to your OAuth2 server's token endpoint and make a POST request using the client credentials grant type flow
 1. Your OAuth2 server should respond back with access token
 2. ADP Marketplace will utilize the access token to authenticate to the respective subscription event endpoint via a GET request and will provide you with the eventUrl which has the subscription details. As an example, the request will look like the following (GET <https://partnerapplication.com/subscription/create?eventUrl=https://apps.adp.com/api/integration/v1/events/12345>)
3. Partner application initiates a POST request to ADP Marketplace's token endpoint:
 1. US Marketplace = <https://apps.adp.com/oauth2/token>
Canada Marketplace = <https://ca.apps.adp.com/oauth2/token>.
 2. The request must be authenticated via basic authentication using the generated **Inbound** Client ID and Client Secret

Header	Query Parameter	Description
	grant_type	REQUIRED. The grant_type parameter must be set to the value client_credentials .
	scope	REQUIRED. The scope must be set to ROLE_APPLICATION
Authorization		REQUIRED. The Authorization header will authenticate with Basic Authentication using the Inbound client id and client secret.

4. ADP Marketplace's OAuth2 server exchanges the client credentials for an access token
5. Partner application initiates a GET request to the eventUrl provided from the subscription notification including the access token in the Authorization header as a bearer token. By default eventUrl response will be in XML, utilize the Accept: application/json header if you wish to receive a JSON response. The partner application must not respond to the initial subscription notification until after a response is received from the eventUrl
6. The subscription event payload response would look something like the following.

```

{
  "type": "SUBSCRIPTION_ORDER",
  "marketplace": {
    "partner": "ADP",
    "baseUrl": "https://apps.adp.com"
  },
  "applicationUuid": null,
  "flag": "DEVELOPMENT",
  "creator": {
    "uuid": "30a0c21c-e5f7-451d-a16f-149eac7061c2",
    "openId": "https://apps.adp.com/openid/id/30a0c21c-
e5f7-451d-a16f-149eac7061c2",
    "email": "John.Doe@adp.com",
    "firstName": "John",
    "lastName": "Doe",
    "language": "en",
    "locale": "en-US",
    "address": null,
    "attributes": null
  },
  "payload": {
    "user": null,
    "company": {
      "uuid": "699da487-d17c-403e-90b7-aac7fb3e1d48",
      "externalId": "FFFFFFFFFFFFFFFF",
      "name": "ADP",
      "email": null,
      "phoneNumber": null,
      "website": null,
      "country": "US"
    },
    "account": null,
    "addonInstance": null,
    "addonBinding": null,
    "order": {
      "editionCode": "Trial + Monthly",
      "addonOfferingCode": null,
      "pricingDuration": "MONTHLY",
      "items": [],
      "customAttributes": []
    },
    "notice": null,
    "configuration": {
      "organizationOID": "FFFFFFFFFFFFFFFF",
      "applicationID": null,
      "associateOID": "G3G898NGDZCHEW8G"
    }
  },
  "returnUrl": null,
  "links": [
    {
      "rel": "oidcClient",
      "href":
"https://apps.adp.com/api/account/v2/subscriptions/fa51b3a6-
6198-4ed4-bf59-b5e418c43b0d/oidcClient"
    }
  ]
}

```

- Then, your application returns a JSON/XML back to ADP Marketplace, passing the status information which would be a 202 see the example below.

Status Code: 202 Accepted

```

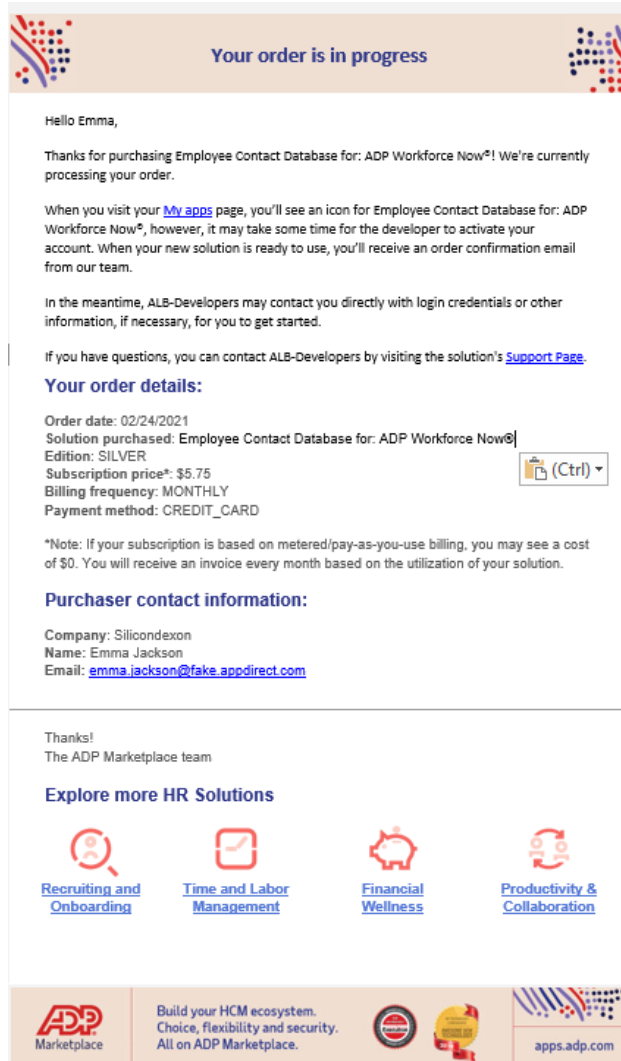
{
  "accountIdentifier":"<OrganizationOID>",
  "success":true
}

```

- This will put the subscription into a pending state. The user cannot do anything with the subscription at this time. The tile for your partner application will show as grey shaded and the client will not be able to click on it or perform any action on the tile. An example of what the client would see in their **"myapps"** section of the marketplace is below.



- The client will also be notified via email that their order is currently processing. This email will contain a link to the **"Support Tab"** of your partner listing so that the client can reachout to you the Partner if they have any additional questions. Screenshot below.



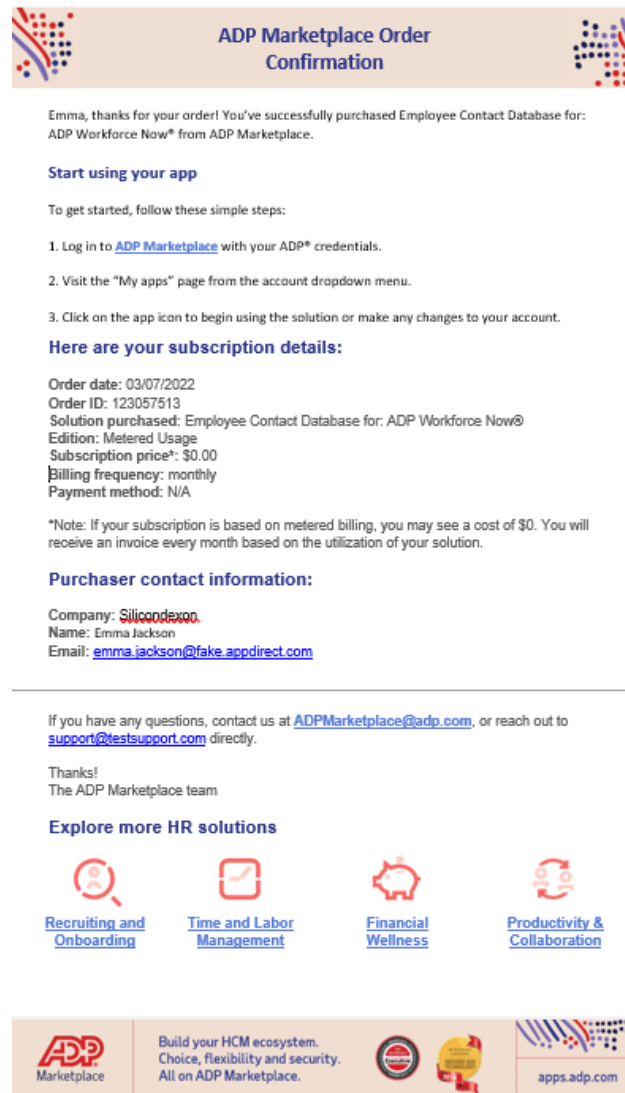
- Once you have provisioned the client and are ready to complete activating the subscription your partner application would send a HTTP POST to the <https://apps.adp.com/api/integration/v1/events/12345/result> to indicate the account creation is completed and the client has been provisioned. The post body that would be used would be the same seen in the previous chapter when following the synchronous process. See below:

Success

```
{
  "accountIdentifier":"<OrganizationOID>",
  "success":true
}
```

}

11. Once the client is activated they will also be notified via email that their order is successful. Below is a screenshot of the email.



12. If for some reason you will need to fail the event you can always send a POST to the <https://apps.adp.com/api/integration/v1/events/12345/result> to indicate the account creation has failed. The POST body that would be used would be the same seen in the previous chapter when following the synchronous process. See below:

False

```
{
  "success": "false"
  "errorCode": "USER_ALREADY_EXISTS"
  "message": "Optional message about the user already existing on ISV"
}
```

13. If a failure is sent the client will be notified via email as well as you the partner. Below are examples of the partner facing failure email as well as the client facing failure email.

Client Facing Email:

Your Employee Contact Database for: ADP Workforce Now® purchase failed

Hello Paul,


Your purchase of Employee Contact Database for: ADP Workforce Now® on the ADP Marketplace could not be completed because of a temporary error with the partner's integration.

No charge was made to your account.


Please try your purchase again later or contact Partnerx2 if this issue persists by visiting the solution's [Support Page](#)

Sincerely,
The ADP Marketplace team


Explore more HR solutions




[Recruiting and Onboarding](#)




[Time and Labor Management](#)




[Financial Wellness](#)




[Productivity & Collaboration](#)



Build your HCM ecosystem.
Choice, flexibility and security.
All on ADP Marketplace.





apps.adp.com

Partner Facing Email (will be enabled Mid to End of March 2022):

An event could not reach your endpoint.

Hi Paul,

An ADP client has tried to purchase or update a subscription to Employee Contact Database for: ADP Workforce Now®. An event was sent, but was not consumed by your endpoint, so you will need to fix the issue with your endpoint. See event details below.

What should you do?

Please check and fix your endpoints to make sure they are up and running, then confirm by running an Integration Report to show that all endpoints are now accessible. If you do not fix the endpoint(s), we will continue to notify you to take action, and your sales could be impacted if clients are unable to successfully complete their purchase or make updates.

Supporting documentation

Review [this documentation](#) with articles on editing and testing your endpoints. If you need additional assistance, please file a ticket in our [help center](#).

Event details:

Event type: SUBSCRIPTION_ORDER
Marketplace: US Marketplace
Product name: Employee Contact Database for: ADP Workforce Now®
Product Edition: Classic Plan
Account identifier: 123-abc-456-def

Thanks!
The ADP Marketplace team



Build your HCM ecosystem.
Choice, flexibility and security.
All on ADP Marketplace.





apps.adp.com



Info

The partner facing email will be triggered when:

1. You send a 200 success = false message(like the one shown above)
2. If your endpoint is down


It will contain links to our documentation on how to test your endpoint using the "**Integration Report**" as well as a link to put a ticket in if you need additional assistance.

Chapter 5


Steps for implementation of user assignment events

1. Log into ADP Marketplace and access the developer menu.
2. Access the menu Products -> Integration -> Edit Integration.
3. Make sure you have the OAuth consumer key and secret you obtained earlier in the subscription events integration.
4. Now, navigate to the "Access Management" section under "Edit Integration". Provide URLs of your application, which ADP Marketplace can call for user assignment and unassignment events.

ACCESS MANAGEMENT

User Assignment Notification URL 

https://partnerapplication.com/UserAssign?eventUrl={eventUrl}

User Unassignment Notification URL 

https://partnerapplication.com/UserUnassign?eventUrl={eventUrl}

Go To Technical Doc →

5. **User assignment:** When the buyer assigns a user to the app in ADP Marketplace, your application will receive a JSON message with the user details. Your application should consume the message and provision the user to access and use the application.
6. The client practitioner assigns billing contacts/admins to your application, so they should be provisioned in your application with the appropriate permissions. These admins are the ones who will moderate the workers (extracted through the workers API call) in your application as authorized users or not. Refer the article [Creating users for your application](#) to learn how to add the non-admin users to your application.
7. **User unassignment:** When the buyer removes a user from the app in ADP Marketplace, your application will receive a JSON message with the user details. Your application should consume the message and remove the access for the user.
Note: Do not select the fields under the section **USER INFORMATION TO SYNC** for a user assignment event.

Chapter 6

Testing Your ADP Marketplace Integration

After developing the integration, test it through the menu option Integration -> Integration Report option.

Integration Report English (United States)

To ensure your integration is properly prepared, your API endpoint health must be full (if applicable). You can check off steps that you completed manually.

AppDirect Integration Tests
A series of tests to ensure that your integration is properly prepared for distribution in AppDirect marketplaces.

1	Subscribe to your product. Test the Subscription Order Event	Run Test
2	Assign (add) a user to your product. Test the User Assignment Event	Run Test
3	Un-assign (remove) users from your product. Test the User Unassignment Event	Run Test
4	Change (update) the subscription. Test the Subscription Change Event	Run Test
5	Cancel the subscription. Test the Subscription Cancel Event	Run Test

1. Each of the notification events can be tested by executing the "Run Test" option under the "AppDirect Integration Tests".
2. If the test is successful, the button "Success" will be displayed for each of the notification tests instead of "Run Test".
3. You can access the event logs through the option Products -> Integration -> Events. This gives the list of events generated and their status. Please also reference [this article](#) on how to troubleshoot pending/stuck events.



Info

Do not use the "Ping Test" option for testing purposes. It doesn't work properly in some networks and cannot be relied upon for completion of the development and testing.



Tip

Helpful Articles:

[White listing IP addresses for ADP Marketplace](#)