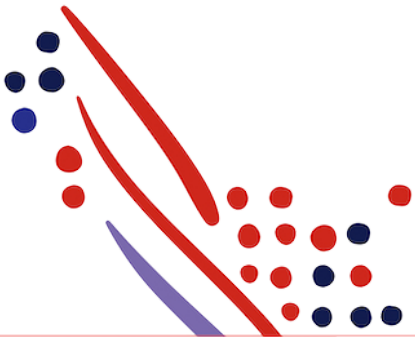
Guide

ADP Marketplace Integration - Separate Credentials (OAuth2 and Basic Auth)

Published on
Jun 18, 2021 2:35PM

Last modified
Mar 30, 2023 9:56AM

Always Designing for People™



ADP Copyright Information

Published on
Jun 18, 2021 2:35PM

Last modified
Mar 30, 2023 9:56AM

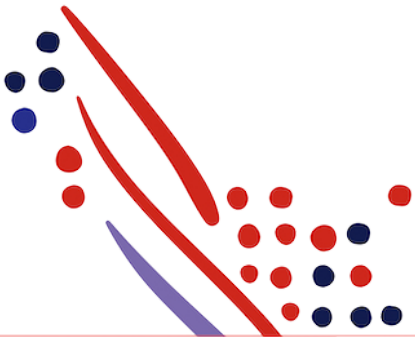


Table of Contents

Chapter 1

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 and Basic Authentication. This method is an alternative for partners who do not have an OAuth2 server protecting their subscription endpoints.

Prerequisites

- Understanding of Basic Authentication
  - https://tools.ietf.org/html/rfc2617#page-5

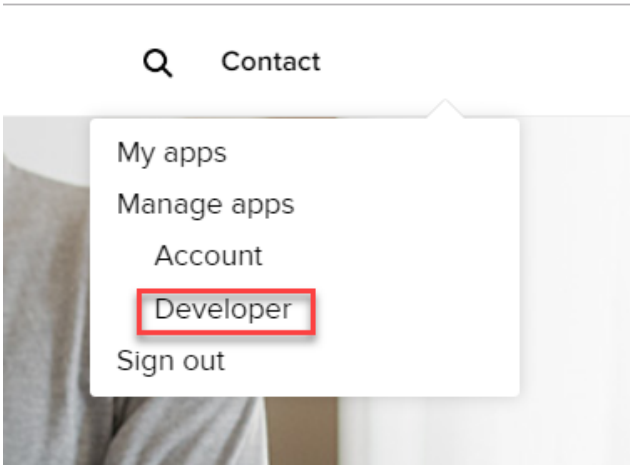
Info

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

Chapter 2

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. You can create your application listing by going to this guide **How to Create a Core Solution App Listing**

3. Access the menu Products -> {your product listing} -> Edit.

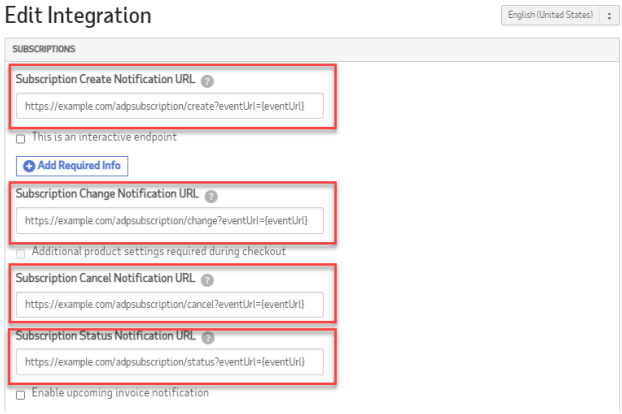
4. From the menu on the left sidebar, navigate to Integration -> Edit Integration.

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}



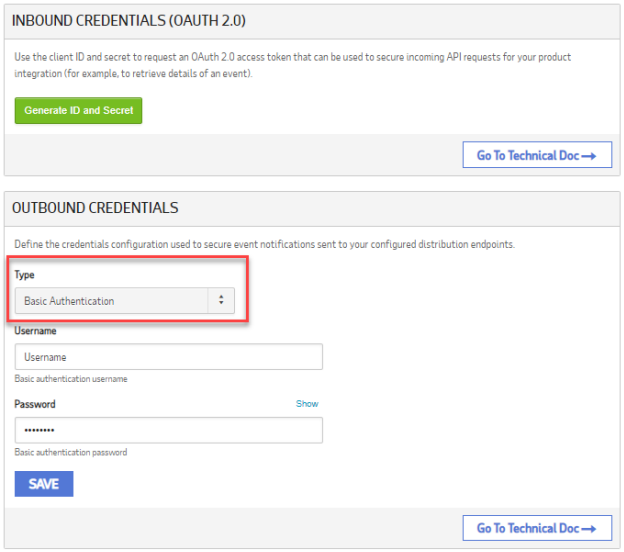
Info

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 **Basic** as the type. Enter in your Username and Password and click **"Save"**.



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**.)

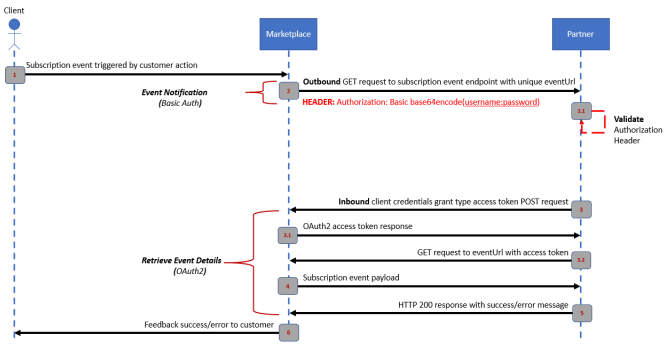
Chapter 3

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 an URL, e.g., **https://apps.adp.com/api/integration/v1/events/12345**.
2. ADP Marketplace sends a subscription event notification to the partner application using the registered URL, e.g., **https://partnerapplication.com/subscription/create?eventUrl=https://apps.adp.com/api/integration/v1/events/12345**
    1. The GET request made to your endpoint will attempt to authenticate using Basic Authentication and you must **validate the Authorization header**.

3. Partner application initiates authorization via a POST request to ADP Marketplace's token endpoint (US Marketplace = **https://apps.adp.com/oauth2/token** Canada Marketplace = **https://ca.apps.adp.com/oauth2/token**).
    1. 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.

    2. ADP Marketplace's OAuth2 server exchanges the client credentials for an access token

    3. 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
    4. **The subscription event payload response would look something ike the following.**

:

5. 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/XML back to ADP Marketplace, passing the value for **organizationOID** for "accountIdentifier", and status information. The two variants of the status notifications are "success":true, or "success": "false". Here are sample JSON documents:

   Success

   {

   "accountIdentifier":"**<OrganizationOID>**",

   "success":true

}

False

{

"success": "false",

"errorCode": "USER_ALREADY_EXISTS",

"message": "Optional message about the user already existing on ISV"

}

Info

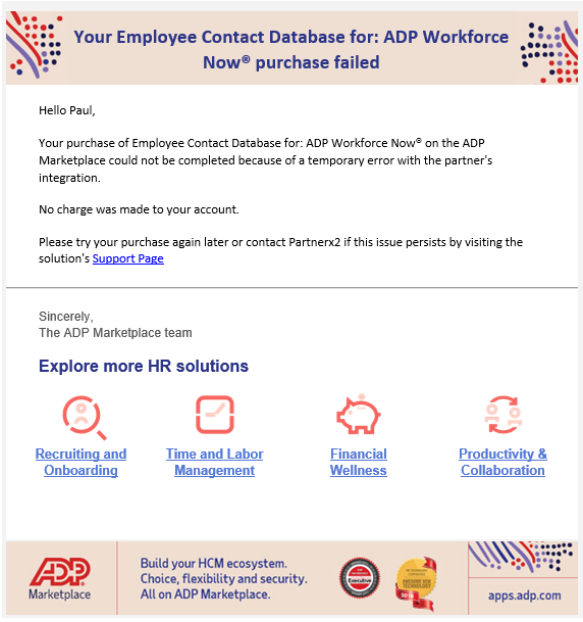
The partner application should always return a 200 status with properly formatted JSON or XML as mentioned above. **Do not return a 500 or 404 status.**

Note: Once you have repsonded 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:



Partner Facing Email (will be enabled End of May 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

Info

The partner facing email will be trigged 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.

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 similar to the implementation of the subscription order and subscription cancellation only. Below are the areas where other notifications help:
     - **Subscription change notification:** When your solution has more than one edition and pricing option.
     - **Subscription status notification:** When you would like to provide a free trial for your core solution.

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 Subscrption Workflow where your 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 an URL, e.g., **https://apps.adp.com/api/integration/v1/events/12345**.
2. ADP Marketplace sends a subscription event notification to the partner application using the registered URL, e.g., **https://partnerapplication.com/subscription/create?eventUrl=https://apps.adp.com/api/integration/v1/events/12345**
    1. The GET request made to your endpoint will attempt to authenticate using Basic Authentication and you must **validate the Authorization header**.
3. Partner application initiates authorization via a POST request to ADP Marketplace's token endpoint (US Marketplace = **https://apps.adp.com/oauth2/token** Canada Marketplace = **https://ca.apps.adp.com/oauth2/token**).
    1. 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.

    2. ADP Marketplace's OAuth2 server exchanges the client credentials for an access token
    3. 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
    4. **The subscription event payload response would look something like the following.**

:

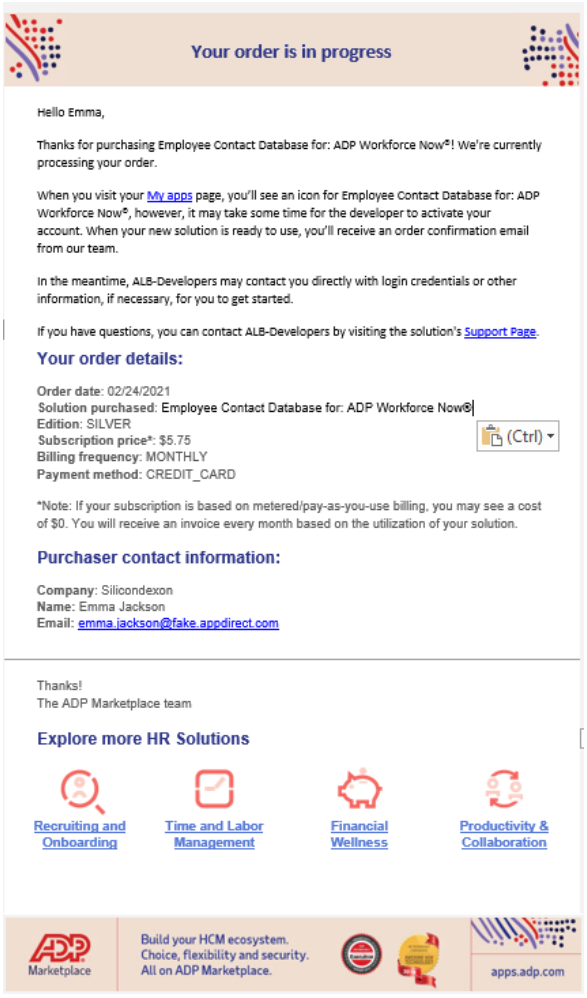
5. Then, your application returns a JSON/XML back to ADP Marketplace, passing the status information which would be a 202 see the example below.

Status Code: 202 Accepted

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

6. This will put the subscription into a pending state. The user cannot do anything with the subscritpion 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.

7. 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.



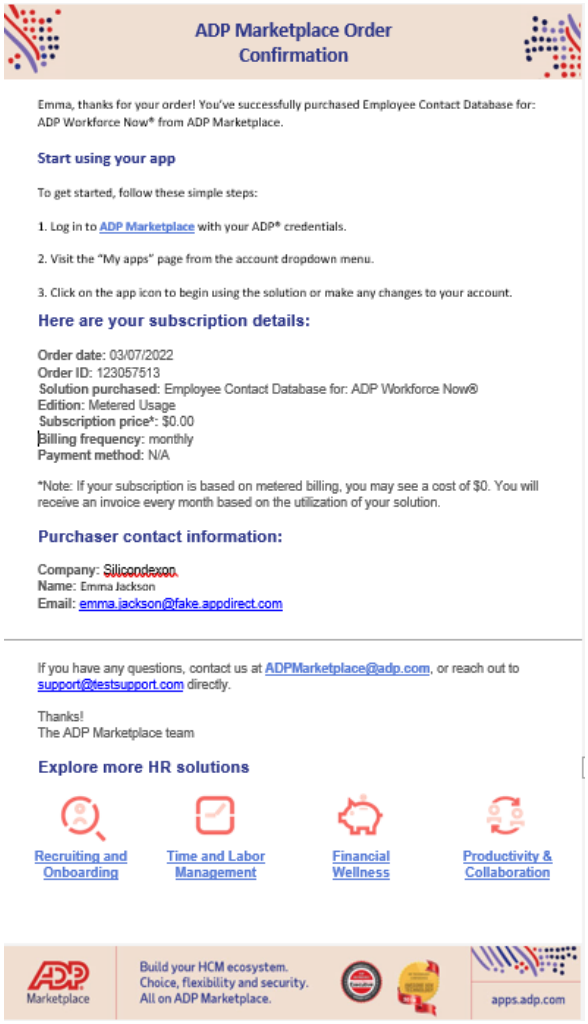
8. Once you have provisioned the client and are ready to complete activating the subscription your partner application would send a HTTP POST to the **https://apps.adp.com/api/integration/v1/events/12345/result** to indicate the account creation is completed and the client thas been provisioned. The 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
```

}

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



10. 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"

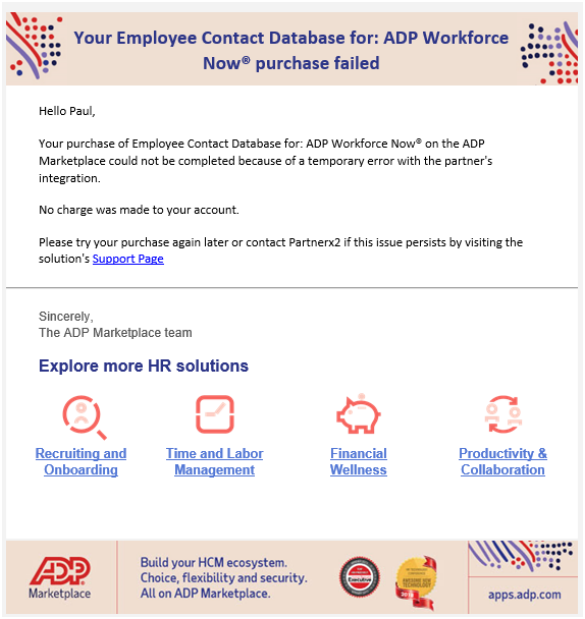
"errorCode": "USER_ALREADY_EXISTS"

"message": "Optional message about the user already existing on ISV"

}

11. 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:



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

Info

The partner facing email will be trigged of when:

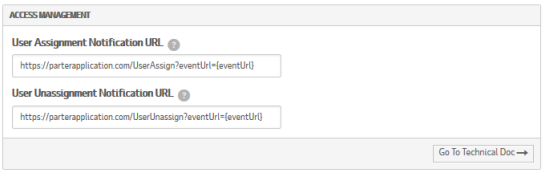
1. You send a 200 success = false message(like the one shown above)
2. If your endpoint is down

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.

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.

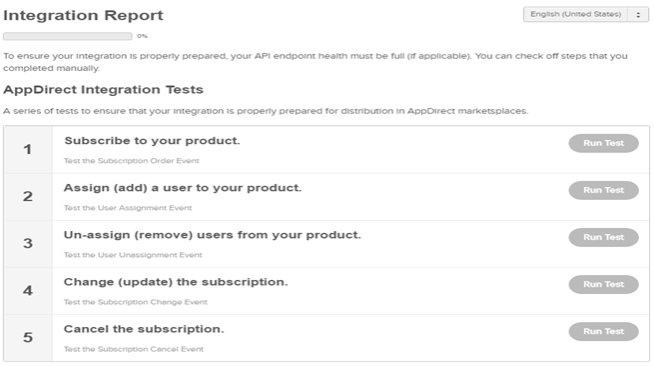


5. **User assignment:** When the buyer assigns a user to the app in ADP Marketplace, your application will receive a JSON message with the user details. Your application should consume the message and provision the user to access and use the application.
6. The client practitioner assigns billing contacts/admins 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.



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