Guide

Single Sign-On Integration (SSO)

Published on
Jun 14, 2021 10:54AM

Last modified
Feb 17, 2022 10:40AM

Always Designing for People™

ADP Copyright Information

Published on
Jun 14, 2021 10:54AM

Last modified
Feb 17, 2022 10:40AM

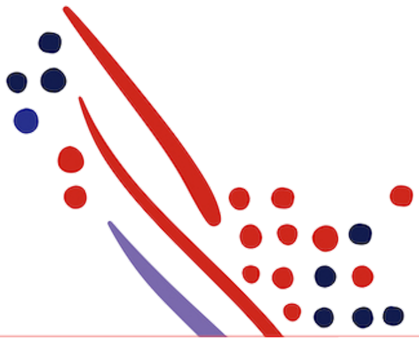


Table of Contents

Chapter 1

Introduction

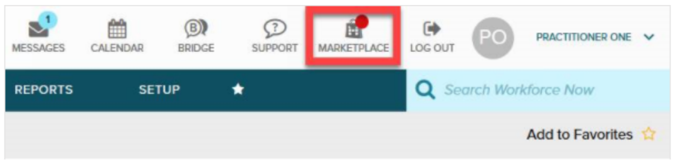
Overview

Integrating ADP Marketplace partner applications with ADP® single sign-on (SSO) is **required** for both **data connector and core applications**, as it helps clients get seamless activation and instant access to applications immediately upon subscribing. In this approach, ADP is the openID provider for partner applications.

ADP clients expect SSO integration between ADP and partner applications. SSO helps them switch back and forth between their ADP platform (e.g., ADP Workforce Now®, RUN Powered by ADP® or ADP Vantage HCM®) and partners' applications. When clients have more than one solution from ADP Marketplace, SSO integration enables them to use multiple applications with one set of credentials.

When clients are logged into their ADP application, they can access their purchased partner applications through SSO by clicking the shopping bag icon as shown below.

ADP Workforce Now, ADP Vantage HCM and RUN Powered by ADP (RUN)



There are many use cases — like employee scheduling, adding a new worker to ADP, time and labor management, or general deductions integrations — in which clients will have to switch back and forth between ADP and partner applications. In these scenarios, clients expect SSO, so they do not have to maintain multiple credentials or continuously sign into different applications.

1.
    - **In the case of work schedules:** A client makes job, time-off or other changes in their ADP application and would like to run the scheduling from the partner application while they are in session with the ADP application as well.
    - **In the case of time and labor management:** A partner's application will send the time and rate from their application for payroll processing. Then, the client will have to sign into their ADP platform for payroll processing. If there are errors in the payroll processing, they will have to switch back and forth between the applications. There could also be cases where the client has to process multiple batches of payroll, which require sending data back and forth between ADP and the partner's application. Some of the situations where practitioners will need to switch between a partner application and ADP as it relates to time partners include:
        - If payroll is rejected.
        - If out-of-sync information causes errors when importing time information into ADP.
            - An employee's pay cycle can change and cause an error in the timesheet import (e.g., weekly to biweekly)
            - Department changes
        - Processing off-cycle payroll.
        - Processing payroll in batches.
        - Other errors or exceptions.
            - It is not possible for partners to account for all exceptions and typically only a generic error message is displayed. Practitioners will need to go into their ADP application for further details.

Approach

Minimum requirement: Clients should be able to access your application via SSO by logging into ADP Marketplace and clicking the app listing from the My apps section of their account. This is best suited for partners who are listing only the data connector (+ referral listing) in the storefront.

If you are listing your core solution (+ integration) on ADP Marketplace, adding the following optional items to your solutions will enhance users' experience:

- Introducing ADP sign-in button and SSO in your application's login page.
- Enabling ADP SSO in your solution's mobile apps.

SSO Technologies

In ADP's SSO, the **authentication** aspect deals with validating the client's employee credentials and establishing their identity with ADP. The **authorization** aspect deals with access restrictions with respect to which APIs the user can access. ADP's SSO uses OpenID Connect for authentication and oAuth 2.0 for authorization.

IMPORTANT: ADP requires, at a minimum, TLS v1.2 for all secure and encrypted communication. Please ensure your application has been configured to use this if you encounter issues around establishing a connection.

ADP Libraries

ADP provides software libraries for SSO integration for technologies such as Java, Python, PHP, Ruby, Node.js and.NET. ADP recommends using software libraries to help you quickly ramp up your SSO integration and keep up with future changes.

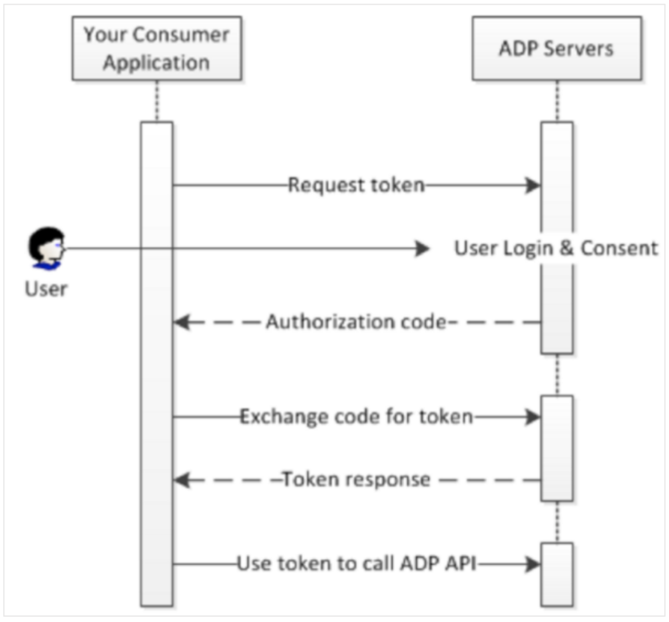
Here is the URL for software libraries: https://developers.adp.com/articles/guides/all/devlib/devlib

You should select only the library for **End User Application** for your relevant technology from the above URL. Note that the software libraries already contain test credentials, test certificates and SSO endpoints to the test environment, which will help you quickly develop and test your integrations. But, you will have to change credentials, certificates and SSO endpoints in the software libraries for the production environment.

Details about the production environment endpoints are included in the documentation for End User App Authorization. You should use the consumer application credentials and SSL certificates from ADP to implement the code with the production endpoints.

Single Sign-On Flow

The below diagram illustrates the flow of steps between the user, partner application (consumer application) and ADP servers.



Note: *SSO tokens are unique per user and should be stored separately.*

OpenID Connect Endpoints

The ADP endpoints involved in the OpenID Connect protocol are described below.

Endpoint	URI
Authorization endpoint	https://accounts.adp.com/auth/oauth/v2/authorize
Token endpoint	https://accounts.adp.com/auth/oauth/v2/token
Userinfo endpoint	https://api.adp.com/core/v1/userinfo
Logout endpoint	https://accounts.adp.com/auth/oauth/v2/logout

Info

Postman Collection
To accompany this document is a Postman Collection to use when Testing SSO as outlined in Chapter 10. Click on **SSO Test Postman Collection** to download the collection.

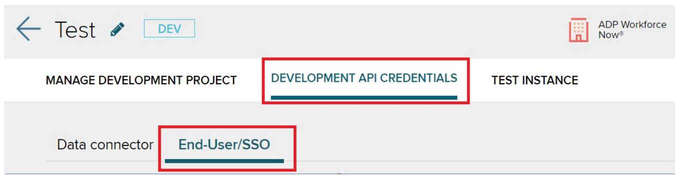
Chapter 2

Prerequisites

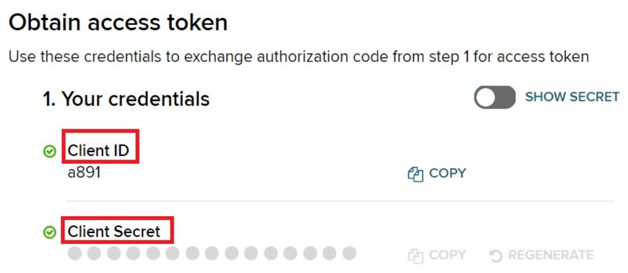
You must obtain the following from ADP Self Service tool in order to implement OpenID Connect with ADP:

- Signed Certificate
  The ADP Self Service tool allows partners to request a Mutual SSL certificate, that is required for **all calls exchanged with ADP API Gateway - *api.adp.com* OR *accounts.adp.com***

- Partner SSO Credentials (End User/SSO Credentials) which can be found by going to:
  1. ADP Self Service Tool and logging in with your partner credentials
  2. Click on your project
  3. Click on "Development API Credentials" then click on "End-User app/SSO"



4. Under Step 2 "Obtain your access token" You will see the Client ID and Client Secret that will need to be used when integrating with SSO.


Obtain access token

Use these credentials to exchange authorization code from step 1 for access token

1. Your credentials

SHOW SECRET

Client ID
a891

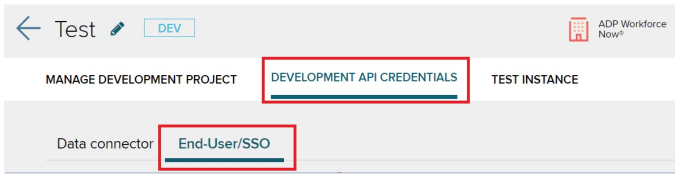
COPY

Client Secret
●●●●●●●●●●●●●●●

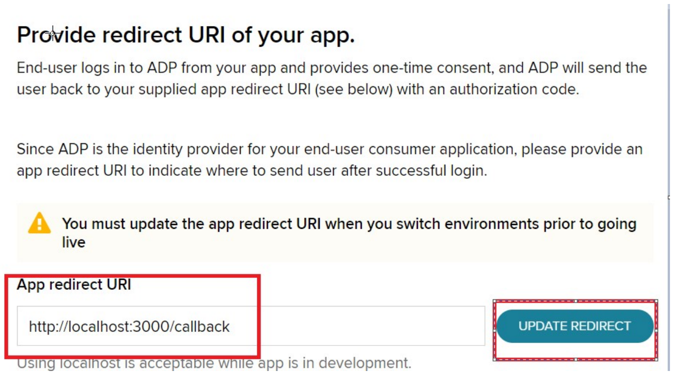
COPY REGENERATE

- SSO Redirect URL which can both be found and updated by going to:
  1. ADP Self Service Tool and logging in with your partner credentials

2. Click on your project
3. Click on "Development API Credentials" then click on "End-user app/SSO"



4. Under Step 1 "Provided redirect URI for you app" you can view and update your Redirect URI (Note only "https" is supported)



Provide redirect URI of your app.

End-user logs in to ADP from your app and provides one-time consent, and ADP will send the user back to your supplied app redirect URI (see below) with an authorization code.

Since ADP is the identity provider for your end-user consumer application, please provide an app redirect URI to indicate where to send user after successful login.

You must update the app redirect URI when you switch environments prior to going live

App redirect URI: http://localhost:3000/callback

Using localhost is acceptable while app is in development.

If you are targeting more than one SOR/ADP platform you will need to add a product identifier so your application knows which partner SSO credentials to use when calling the **userinfo** API

- **Example:** https://adppartner.com/adp/callback?productId=RUN

Chapter 3

Authentication Request

Note: *If you are using the ADP library, the required steps in the next 4 chapters are covered Please skip to* **Chapter 7 "Retrieve End-User Identity Profile"**

Your application should redirect the user to the crafted URI below. ADP will validate the identity of your consumer application and the request parameters. Then it authenticates the end user and obtains the end user's consent to access the data requested by your consumer application. If the end user denies consent or fails authentication with ADP, your consumer application receives an error on the redirect URI. Otherwise, your consumer application receives an authorization code.

For a basic request, specify the following parameters:

Parameter	Description
response_type	REQUIRED. The response_type parameter must be set to the value "code".
client_id	REQUIRED. The client_id parameter must be set to the consumer application's account identifier.
redirect_uri	REQUIRED. The redirect_uri parameter must be set to the value provided during the registration of your consumer application. This value must

	be in URL encoded format and use the HTTPS protocol.
scope	STRONGLY RECOMMENDED. the scope must include the keywords openid or openid profile.
state	STRONGLY RECOMMENDED. The state parameter should be set to a unique value in time. This parameter is used to prevent cross-site request forgery (CSRF) attacks against the redirection URI. It is used to link the request with the response delivered on the redirect URI. Uniqueness is required to prevent potential replay attacks of the authorization request.
login_hint	OPTIONAL. When your app knows which user it is trying to authenticate, it can provide this parameter as a hint to the authentication server. The value must be a user's ADP User ID.

Here is an example of a complete authentication URL.

OpenID Connect Authentication URL (*line breaks and spaces for readability*)

```
https://accounts.adp.com/auth/oauth/v2/authorize?
    client_id=x6b8144af3ca14d0e821deefb5b310ecc&
    response_type=code&
    scope=openid%20profile&
    redirect_uri=https%3A%2F%2Fconsumerapp%2E&com%2Fcallback&
    state=25757208711097519312159
```

ADP conveys the authorization code (or the error) by redirecting the end-user agent to your consumer application's redirect_uri. This URI must be hosted by the server-side component of your consumer application and must use HTTPS.ADP conveys the authorization code (or the error) by redirecting the end-user agent to your consumer application's redirect_uri. This URI must be hosted by the server-side component of your consumer application and must use HTTPS.

Chapter 4

Confirm Anti-Forgery State Token for SSO

The response to your authentication request is sent to the redirect_uri that you specified in the request. All responses are returned in the query string, as shown below:

```
https://consumerapp.com/callback?code=d7289a844107481dbf6a6555de2052e2&state=25757208711097519312159&scope=openid%20pro
```

Parameter	Description
code	The authorization code granted to the consumer application. The authorization code is an alphanumeric value between 25 and 128 characters.
state	The exact value provided by the consumer application on the authorization request.
scope	The exact value provided by the consumer application on the authorization request.

Your consumer application will only receive the code if the user successfully authenticates with ADP, and must check the value of the state parameter to verify it matches the value in the authorize request. If the value does not match, your consumer application must ignore the response and stop processing the authorization request. This round-trip verification helps to ensure that the user, not a malicious script, is making the request.

Chapter 5

Exchange Authorization Code for Access Token

The response to your authentication request includes a code parameter, a one-time authorization code that your server can exchange for an access token. Your server makes this exchange by sending an HTTPS POST request. The POST request is sent to the token endpoint.

The request must include the following body within the POST request for the ADP access_token:

Header	Query Parameter	Description
	grant_type	REQUIRED. The grant_type parameter must be set to the value "authorization_code".
	code	REQUIRED. The code parameter must be set to the value returned by the ADP Authorization Service in the authorization response.
	redirect_uri	REQUIRED. The redirect_uri parameter must be set to value provided during the registration of the consumer application. This value must be provided in URL encoded format and use the HTTPS protocol.
client_id		REQUIRED. The client_id parameter is the consumer application's account identifier. In general, the consumer application should ***use the HTTP Authorization header to pass the client_id and the client_secret parameters via Basic Authentication***.
client_secret		REQUIRED. The client_secret parameter is the consumer application's account secret. In general, the consumer application should ***use the HTTP Authorization header to pass the client_id and the client_secret parameters via Basic Authentication.***

Your consumer application must send the request with the X.509 certificate provided during registration.

In general, your consumer application should provide the authentication credentials using the HTTP Basic authentication scheme (or other designated scheme) and provide the HTTP Authorization header in the access token request. The consumer application's client_id and client_secret must be provided as required by IETF RFC 2617 - the string : encoded in base 64 where client_id and client_secret are the values assigned to the consumer application during registration (or secret reset).

Your consumer application must pass all parameters in an URL encoded format with UTF-8 character encoding as specified by the HTTP header:

```
Content-Type: application/x-www-form-urlencoded
```

The actual request might look like the following example

(line breaks and spaces added for readability):

```
POST /auth/oauth/v2/token HTTP/1.1
Headers: Host: accounts.adp.com
         Authorization: Basic QURQVGFibGV0OnRoZXRhYmxldHBhc3N3b3Jk
         Content-Type: application/x-www-form-urlencoded
Body:      code=d7289a844107481dbf6a6555de2052e2
         &redirect_uri=https%3A%2F%2Fconsumerapp%2E&com%2Fcallback
         &grant_type=authorization_code
```

A successful response to this request contains the following fields in a JSON array:

Parameter	Description
access_token	The access_token parameter is set to the value of the access token issued by the ADP authorization service in exchange for the authorization

	code.
token_type	Identifies the type of token returned. At this time, this field always has the value Bearer.
expires_in	The expires_in parameter is set to the time remaining in the token's life (in seconds). For example, the value "3600" indicates that the access token will expire in one hour.
id_token	A JWT that contains identity information about the user that is digitally signed by ADP.

```
{
 "access_token": "314ec73f-7eb5-4eff-b0d6-6fc2d5508f65"
 "token_type": "Bearer"
 "expires_in": 3600
 "refresh_token": ""
 "scope": "openid"
 "id_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ewogInN1YiI6ICJodHRwczovL2FjY291bnRzLmFkcC5jb20vdXNlci9HM1haQUpZSFhFVjZESDFOL0czQjcyQjgxSEdZM0VZ
 "id_token_type": "urn:ietf:params:oauth:grant-type:jwt-bearer"
}
```

Chapter 6

Validate JWT (id_token)

An identity token is an assertion of the authentication of the end user by the OpenID Connect identity provider. The assertion is provided as a JSON Web Token (JWT) object in the id_token parameter of the JSON object returned on a successful authorization code exchange.

A JWT contains three binhex-encoded sections separated by dots ("."): a header, a set of asserted claims, and the signature of the header and the claims. As a best practice, partner application should decode and validate the JWT.

The encoded claims section of the id_token decodes to the following JSON object.

```
{
 "sub": "https://accounts.adp.com/user/G3XZAJYHXEV6DH1N/G3B72B81HGY3EY0D",
 "aud": "fa3e3282-6369-4494-95f3-d87b8c30d489",
 "c_hash": "9IPd-aaM96-1X7LUUP_wOg",
 "acr": "bWg6Teie6YTrxNArBsaxuZcWO04=",
 "azp": "fa3e3282-6369-4494-95f3-d87b8c30d489",
 "auth_time": 1570308958,
 "iss": "https://accounts.adp.com",
 "exp": 1570395358,
 "iat": 1570308958
}
```

Parameter	Description
iss	The issuer of the authentication assertion. In this case, theADP OpenID Connect Provider contains the following value: https://accounts.adp.com.
sub	The subject of the assertion. A unique identifier associated with the end user authenticating with ADP.
aud	The party to which the id_token was issued. The value is your consumer application's client_id.
exp	The expiration time of the assertion. Your consumer application should not accept expired assertions.
iat	The time the assertion was issued. Your consumer application can use this parameter to determine the age of the assertion.
auth_time	The time when the end-user authentication occurred.
nonce	Returned if your consumer application provided this parameter in the authorize request. This parameter mitigates replay attacks.

azp	The party to which the id_token was issued. The value is your consumer application's client_id.
c_hash	Authorization code hash value. It is the binhex encoding of the left-most half of the authorization. The hash algorithm is the algorithm specified in the header section.

Normally, it is critical that you validate an ID token before you use it, but since you are communicating directly with ADP over an HTTPS channel and using your client credentials and signed certificate to authenticate yourself to ADP, you can be confident that the token you receive really comes from ADP and is valid. If your server passes the ID token to other components of your app, it is extremely important that the other components validate the token before using it.

Your consumer application may decode and minimally validate the id_token as follows:

- Validate the signature of the JWT.
- Validate that the value of the nonce parameter matches the value provided in the authorize request if your consumer application provided one.
- Validate that the value of the iss (Issuer) parameter matches https://accounts.adp.com.
- Validate that the "aud" (audience) parameter contains the client_id assigned to your consumer application.
- Validate that the "azp" (authorized party) parameter contains the client_id assigned to your consumer application.
- Validate that the id_token has not expired or that the current time is before the value specified in the "exp" parameter.
- Validate that the c_hash parameter matches the authorization code provided by the result of the authorization request to the ADP Authorization Service.

Chapter 7

Retrieve End-User Identity Profile

If your consumer application is authorized to retrieve an end user's basic identity profile (your application uses "openid profile" in the authorize request and the user grants consent), then your application can use the access token to invoke the userinfo Web API and retrieve the end user's profile information. The following shows an example request, with line breaks and spaces added for readability.

```
GET /core/v1/userinfo HTTP/1.1
    Host: api.adp.com
    Authorization: Bearer 024ded5f831d4483a9c606710026b09b
    Accept: application/json
```

If successful, the Userinfo V1 API provides the information described below about the end user.

Parameter	Description
sub	Identifier for the end user.
name	The end user's full name in a displayable form.
given_name	The end user's first name.
family_name	The end user's last name.
email	The end user's work email address if available.
oraganizationOID	A unique organization identifier within ADP.
associateOID	A unique user identifier within ADP.

```
{
 "sub":"https://accounts.adp.com/user/G3XZAJYHXEV6DH1N/G3B72B81HGY3EY0D",
 "organizationOID":"G3XZAJYHXEV6DH1N",
 "associateOID":"G3B72B81HGY3EY0D",
 "given_name":"Usha",
 "family_name":"Rani",
 "name":"Usha Rani",
 "email":"developer@adp.com"
}
```

Chapter 8

SSO Logout

When an end user is logged into your app with their ADP identity, ADP is serving as the OpenID provider (OP) and your app is the OpenID relaying party (RP). When an end user logs out of your application, your app should allow the end user to log out of their OP account and redirect the end user's user agent to the OP's logout endpoint URL.

Parameter	Description
id_token_hint	RECOMMENDED. Previously issued ID Token passed to the logout endpoint as a hint about the End-User's current authenticated session with the Client. This is used as an indication of the identity of the End-User that the RP is requesting be logged out by the OP. The OP need not be listed as an audience of the ID Token when it is used as an id_token_hint value.
post_logout_redirect_uri	OPTIONAL. URL to which the RP is requesting that the end user's user agent be redirected after a logout has been performed. The value MUST have been previously registered with the OP, either using thepost_logout_redirect_uris Registration parameter or via another mechanism. If supplied, the OP SHOULD honor this request following the logout.
state	OPTIONAL. Opaque value used by the RP to maintain state between the logout request and the callback to the endpoint specified by the post_logout_redirect_uri parameter. If included in the logout request, the OP passes this value back to the RP using the state query parameter when redirecting the User Agent back to the RP.

Here is an example of a complete OpenID Connect logout URL **(with line breaks and spaces for readability)**.

```
https://accounts.adp.com/auth/oauth/v2/logout?
    post_logout_redirect_uri=https://adppartner.com/login
```

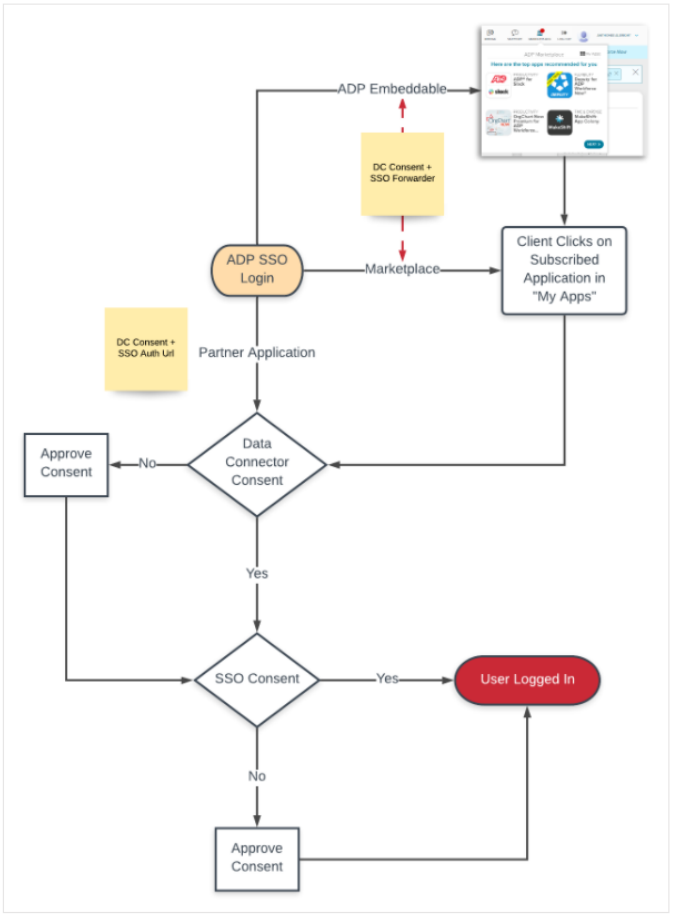
Note: Be sure to destroy the user's session first in your application before redirecting them to the logout URI.

Chapter 9

Client Experience Implementation

There are two ways in which a client should access your application to single sign on. The client can either enter in from **ADP Marketplace** or they may login directly through your **Partner Application Login Page**. In each case we want to ensure that we capture both Data Connector Consent and SSO Consent prior to them accessing the contents of your partner application. The below diagram explains the intended workflow.

Note: Consent only needs to be provided once on behalf of the client's organization by their practitioner for the data connector and each user will only need to provide consent for SSO to release their End-User Identity Profile just once as well.

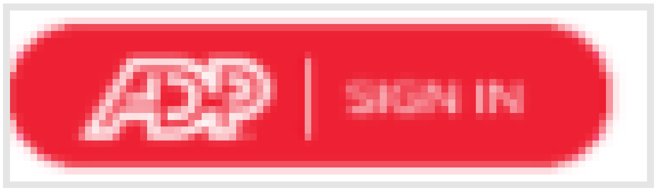


To achieve this, what we will do is chain the Data Connector Consent with the respective SSO URI, depending on where the client initiates their login from.

Note: Please refer to the ADP Self Service tool for the Data Connector Consent URL that is designated for your Connector consumer application.

Partner Application Login

Clients who access your application's login page directly should be forwarded to the below URI chain through an ADP login button



Your href should consist of the Data Connector Consent with a success redirect to your **OpenID Connect Auth URL** as seen in **Chapter 3 Authentication Request**

https://**{DataConnector-Consent}**&successUri=***urlencode*({SSO-AuthURL})**

Example:

```
https://adpapps.adp.com/consent-manager/pending/direct?consumerApplicationID=6bd8ff67-03a0-40c9-aae0-bff806910aaa&successUri=https%3
```

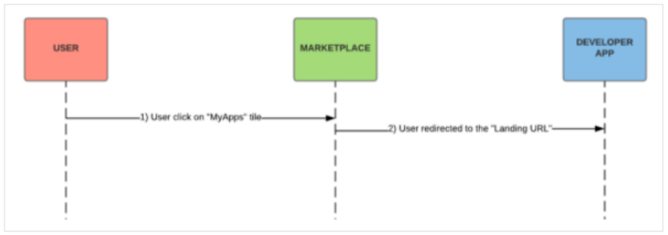
Info

Click Here to download a copy of the "Login With ADP" button.

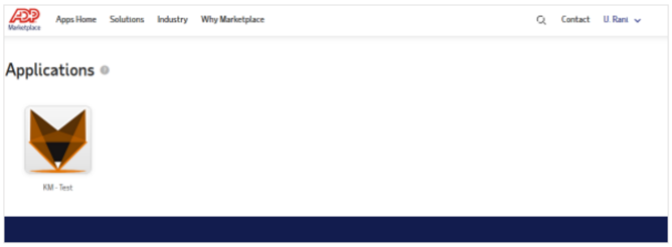
ADP Marketplace SSO Login

Clients may also access your partner application directly from ADP Marketplace where they are already authenticated via ADP SSO and are conveniently redirected to the "My apps" section after a successful subscription to your application.

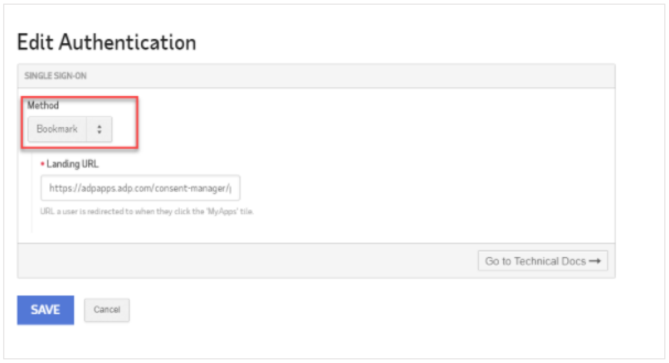
The following image describes the Bookmark authentication user flow for applications that are integrated with a marketplace.


USER
MARKETPLACE
DEVELOPER APP
1) User click on "MyApps" tile
2) User redirected to the "Landing URL"

A screenshot of a sample app under the MyApps section.



In order to configure the link for your application tile navigate to the **Edit Authentication** section in your product listing and choose **Bookmark** as the login method.



Edit Authentication

SINGLE SIGN-ON

Method: Bookmark

• Landing URL: https://adpapps.adp.com/consent-manager/

URL a user is redirected to when they click the 'MyApps' tile.

The landing URL will be slightly different than the approach directly from the login page since we can't generate a CSRF for the state parameter of the OpenID Connect Auth URL. We will chain our Data Connector Consent with a landing page on your partner application that will forward them to a properly crafted **OpenID Connect Auth URL** as seen in **Chapter 3 Authentication Request**. Please be aware your **{SSO-Forwarder}** should begin the authentication process outlined in **Chapter 3 Authentication Request.**

https://**{DataConnector-Consent}**&successUri=***urlencode*({SSO-Forwarder})**

```
https://adpapps.adp.com/consent-manager/pending/direct?consumerApplicationID=6bd8ff67-03a0-40c9-aae0-bff806910aaa&successUri=https%3.
```

Bookmark authentication mechanism supports placeholders to send values from Marketplace to your application when the app is clicked from the "MyApps" section. These place holders will have to be configured as part of the landing url that you define for the bookmark method. A sample URL with the placeholder will look like the below:

https://adpapps.adp.com/consent-manager/pending/direct?consumerApplicationID=6bd8ff67-03a0-40c9-aae0-bff806910aaa&successUri=https%3A%2F%2Fadppartner.com%2Fadp%2Fsso%2Fforwarder&organizationOid={accountIdentifier}

Please see the link Bookmark authentication and placeholders section below for the list of placeholders that are supported by Marketplace.

Bookmark authentication and placeholder values

When you configure Bookmark authentication, you can use placeholders with Name ID and Attribute Values. AppDirect supports the following placeholders.

- **{accountIdentifier} template field**—Optional. It is populated with the unique account identifier provided by you, the developer, during a subscription order.
- **{marketplace.url}**—Marketplace in which the subscription resides
- **{company.uuid}**—UUID of the company in AppDirect
- **{company.name}**—Company name
- **{companyEntitlement.uuid}**—UUID of the subscription in AppDirect
- **{companyEntitlement.externalVendorIdentifier}**—Identifier optionally provided by your application when the subscription was purchased
- **{user.uuid}**—UUID of the user in AppDirect
- **{user.emailAddress}**—Email address of the user
- **{user.firstName}**—First name of the user
- **{user.lastName}**—Last name of the user
- **{user.language**}—User language in IETF BCP 47 format. For example, en-UK
- **{userEntitlement.externalVendorIdentifier}**—Identifier optionally provided by your application when the user was assigned to the subscription

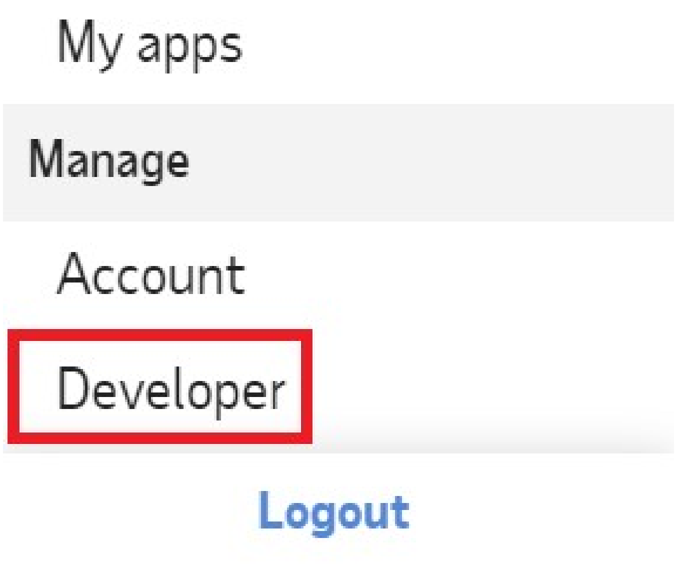
Chapter 10

Testing SSO

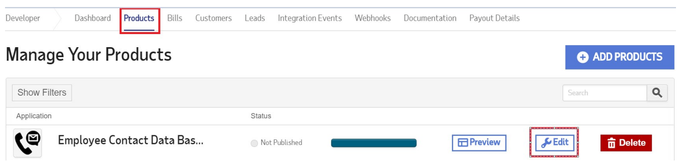
ADP Marketplace SSO Testing

Note: as a prerequisite to this you would have had to follow the steps in **Chapter 2 Prerequisites**

1. Log into either https://apps.adp.com/en-US/home or https://ca.apps.adp.com/en-CA/home with your Partner Credentials.
2. Click on your name in the upper right-hand corner and then click on "Developer"



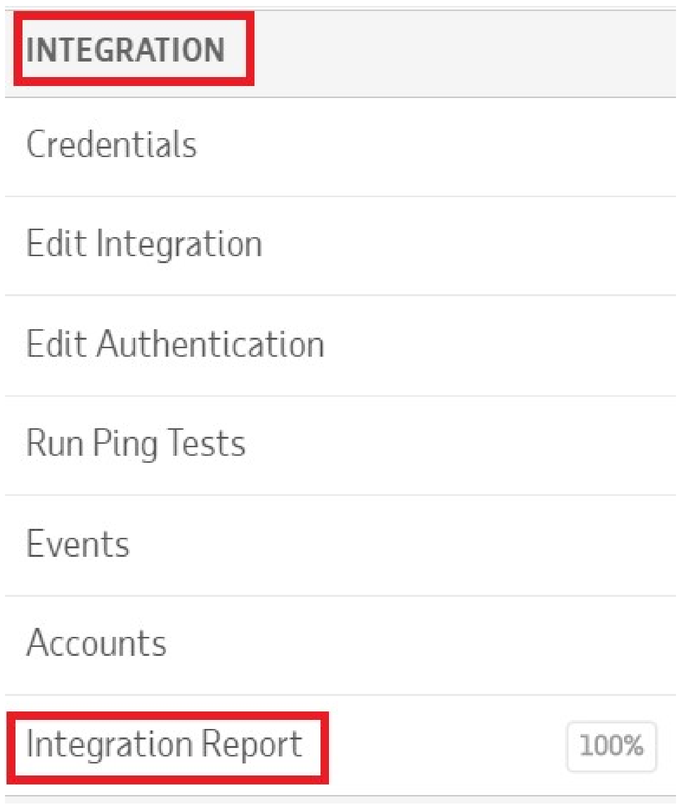
3. Then click on "Products" and click "edit" next to your application.



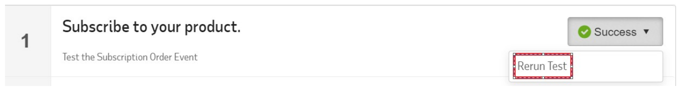
Manage Your Products

Application	Status
Employee Contact Data Bas...	Not Published

4. Under Integration click on "Integration Report"

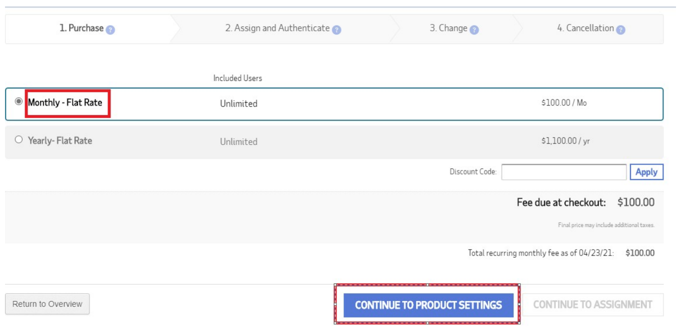


5. On the right-hand side click on "Test" or "Re-Run Test" next to "Subscribe to your product"



1 Subscribe to your product.
Test the Subscription Order Event

6. Choose an edition and then click on "Continue to Product Settings"



1. Purchase | 2. Assign and Authenticate | 3. Change | 4. Cancellation

	Included Users	
● Monthly - Flat Rate	Unlimited	$100.00 / Mo
○ Yearly- Flat Rate	Unlimited	$1,100.00 / yr

Discount Code: Apply

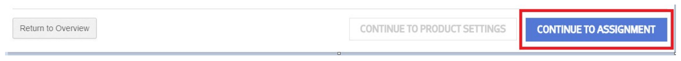
Fee due at checkout: $100.00

Final price may include additional taxes.

Total recurring monthly fee as of 04/23/21: $100.00

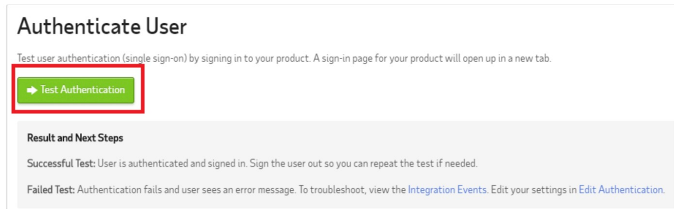
Return to Overview | CONTINUE TO PRODUCT SETTINGS | CONTINUE TO ASSIGNMENT

7. Once Subscribed you can now click on "Continue to Assignment"



Return to Overview | CONTINUE TO PRODUCT SETTINGS | CONTINUE TO ASSIGNMENT

8. At the bottom of the User Assignment page you can now click on "Test Authentication"



Authenticate User

Test user authentication (single sign-on) by signing in to your product. A sign-in page for your product will open up in a new tab.

Test Authentication

Result and Next Steps

Successful Test: User is authenticated and signed in. Sign the user out so you can repeat the test if needed.

Failed Test: Authentication fails and user sees an error message. To troubleshoot, view the Integration Events. Edit your settings in Edit Authentication.

9. If successful, your user will be authenticated and signed into your application.

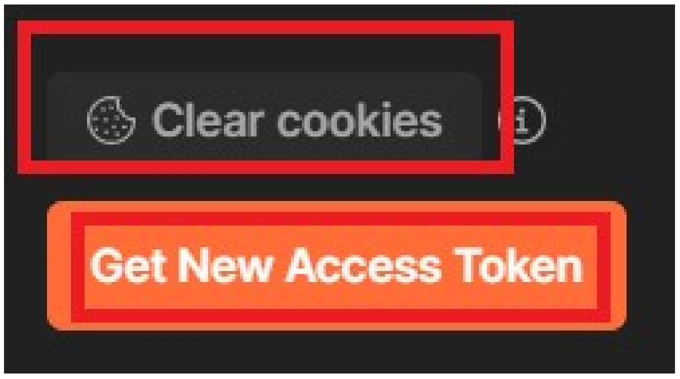
How to test SSO via POSTMAN

1. Add the Postman Collection at the bottom of **Chapter 1** to your Postman Instance.

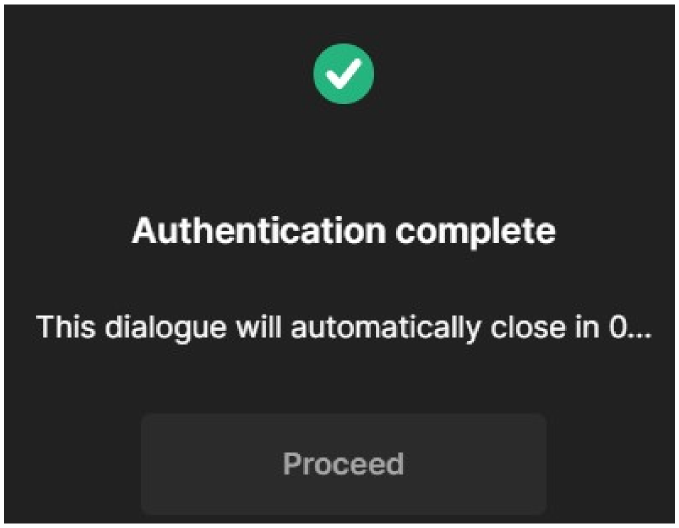
2. Make sure your Mutual SSL Certificate is added to Postman by following the steps in the guide **HERE.**

3. Go to the Authorization tab and add your end User/SSO credentials (Client ID and Client Secret) which can be found in the **Developer Self-Service Portal**

4. Once all information is entered go to the bottom of the Authorization Tab click on **"Clear cookies"** then click on **"Get New Access Token"**.



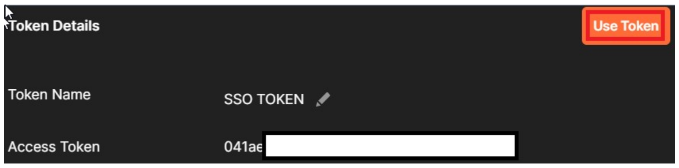
5. This will bring up the "Welcome to ADP" login page. Login with your **Sandbox Credentials**.
6. Once you successfully login you will see the Pop-up below just hit **"Proceed"**



Authentication complete

This dialogue will automatically close in 0...

7. On the next pop-up click on **"Use Token"**

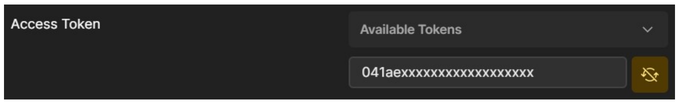


Token Details

Token Name SSO TOKEN

Access Token 041ae[redacted]

8. That token will then be automatically added to the Access Token field shown in the screenshot below

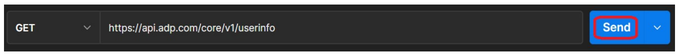


Access Token

Available Tokens

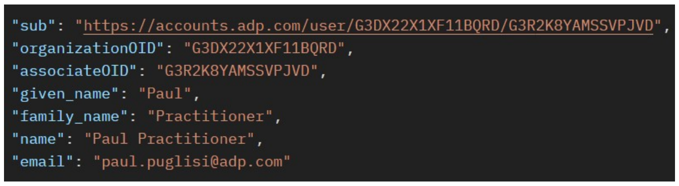
041aexxxxxxxxxxxxxxxxxxx

9. Once added you can now make the https://api.adp.com/core/v1/userinfo call by clicking on **"Send"**



GET https://api.adp.com/core/v1/userinfo

10. A successful call with result in displaying the basic profile information.



```
"sub": "https://accounts.adp.com/user/G3DX22X1XF11BQRD/G3R2K8YAMSSVPJVD",
"organizationOID": "G3DX22X1XF11BQRD",
"associateOID": "G3R2K8YAMSSVPJVD",
"given_name": "Paul",
"family_name": "Practitioner",
"name": "Paul Practitioner",
"email": "paul.puglisi@adp.com"
```

Note: Until your application is published you will not be able to test through the **"My apps"** menu, only through the **Integration Report**. Please enter in the landing URL in a different browser and login with the test client credentials provided in ADP Self Service tool . When testing SSO Sandbox Credentials should be used. These would be the credentials you are currently using to log into your Test Instance of ADP WFN, RUN or Vantage.

Chapter 11

FAQ

Question 1: Why am I getting an "invalid_grant" message when I exchange my authorization code for an access token?

Answer: If you are not using a developer library provided by ADP, but prefer to use a 3rd party OAuth 2.0 library, please ensure that your initial authorization request contains only these limited query parameters:

response_type
client_id
redirect_uri
scope
state
You should not have any additional parameters, such as ***code_challenge,*** because ADP does not support the PKCE-enhanced Authorization Code Flow.