Setting up Single Sign-On (SSO) with ADP

One of the core tasks for integrating an End-User consumer application with ADP is setting up SSO. This article introduces SSO for ADP and describes the SSO setup process using the OpenID Connect specification.

Overview: Introduction to Setting Up SSO with ADP

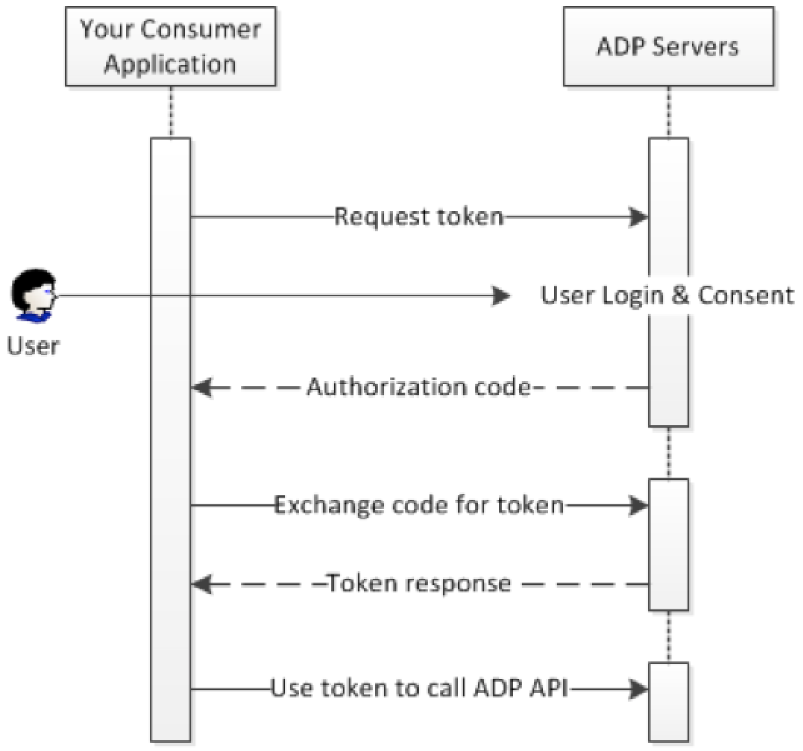
Introduction

ADP uses the OpenID Connect protocol to allow end-users to authenticate their identity with ADP credentials. ADP is the identity provider responsible for verifying the identity of users and applications, and issuing identity tokens upon successful authentication of those users and applications.

The basic authentication flow is:

1. An end-user agent accesses your consumer application.
2. Your consumer application redirects the end-user agent to the ADP authorization endpoint.
3. ADP authenticates the end-user's ADP credentials and obtains the end-user's consent to access the end-user's information requested by your consumer application.
4. ADP redirects the end-user agent to the redirect URI pre-registered with ADP; your consumer application receives an authorization code as a parameter of the redirection.
5. You consumer application contacts the ADP token endpoint to exchange the authorization code for an access token.
6. ADP authenticates your consumer application, verifies the validity of the authorization code and provides an access token to your application.
7. If your consumer application needs additional identity context to identify the end-user, your consumer application uses the access token provided by ADP to access the userinfo API and retrieve the end-users identity profile.

The following figure illustrates the login process.



ADP provides libraries that you can use to take care of many of the implementation details of authenticating users and gaining access to ADP APIs. If you choose not to use a library, follow the instructions in the remainder of this document, which describes the HTTP request flows that underly the available libraries.

Prerequisites

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

1. Signed Certificate
2. Client Credentials

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

Step 1: Create an Anti-Forgery State Token for SSO

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.

One good choice for a state token is a string of 30 or so characters constructed using a high-quality random-number generator.

Step 2: Send Authentication Request

The next step is forming an HTTPS GET request with the appropriate URI parameters. Your consumer application redirects the end-user agent to the ADP authorization endpoint and obtains the authorization code to authenticate the end-user.

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	SRONGLY 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.

Here is an example of a complete OpenID Connect authentication URL, with 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 validates 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.

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.

Step 3: Confirm Anti-Forgery State Token for SSO

The response 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
```

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.

Your consumer application must check the value of the state parameter to verify if it matches the value provided 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.

Step 4: Exchange Code for Access Token and ID Token

The response includes a code parameter, a one-time authorization code that your server can exchange for an access token and ID 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 parameters in the POST body:

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.
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.

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
     Host: accounts.adp.com
     Authorization: Basic QURQVGFibGV0OnRoZXRhYmxldHBhc3N3b3Jk
     Content-Type: application/x-www-form-urlencoded
     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.

Step 5: Validate 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. Your consumer application must decode and validate the JWT.

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

```
{ "iss":"https://accounts.adp.com",
  "sub":"https://accounts.adp.com/user/G123123123123213/G456783748392999",
  "exp":1412793735,
  "iat":1412786535,
  "auth_time":1412786535,
  "nonce":"163141324",
  "azp":"c5348fb0-efd6-4632-a359-d6c2f5ab04f6",
  "c_hash":"2oYJkgWbM3jgT5QjuSfNaQ"
}
```

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.

Step 6: 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.

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

If successful, the userinfo Web 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.

Step 7: Log Out End-User

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://partners.adp.com
```