Chapter 3

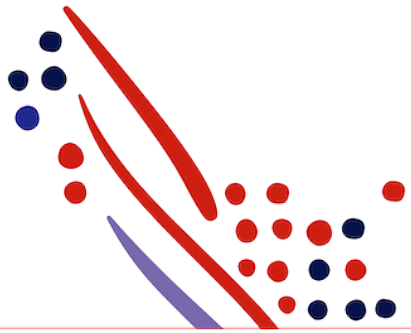
# Access Token

from ADP Marketplace ESI - Getting started

Published on
Mar 22, 2022, 09:18 AM

Last modified
Oct 26, 2026, 09:29 AM

Always Designing for People™



# 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.

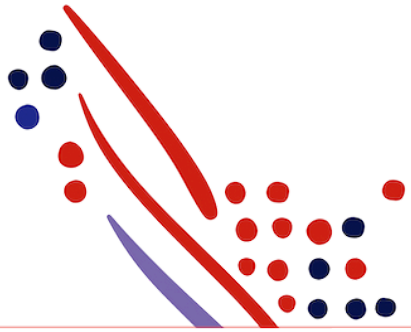
All other trademarks are the property of their respective owners.

Copyright © 2026 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 inaccurancies 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 programes described in this publication.

Published on
Mar 22, 2022, 09:18 AM

Published on
Oct 26, 2026, 09:29 AM



# Chapter Contents

Chapter 3

**ADP Marketplace ESI - Getting started**

- Request an Access Token from ADP
- Example (Curl) – Form POST
- Response
- How to use an access token
- ADP Web API Limitations
- Security Considerations

Chapter 3

# Access Token

As part of the OpenID Connect and Open Authorization (OAuth) 2.0 flows, Access Tokens are provided by ADP and used for secure calls to protect ADP APIs. This document outlines how Access Tokens are used as well as some limitations and security considerations.

## Request an Access Token from ADP

Your application can request an access token by sending an HTTPS POST request to the token endpoint:
**https://accounts.eu.adp.com/auth/oauth/v2/token**

The request must include the following parameters:

| Parameter | Description |
|---|---|
| grant_type | REQUIRED. Must be set to the value "client_credentials". |
| client_id | REQUIRED. The consumer application's account identifier, assigned during account registration or at secret reset. |
| client_secret | REQUIRED. The consumer application's account password, assigned during account registration or at secret reset. |

There are two options to send the required parameters. Your consumer application should pass the client_id and client_secret parameters in the HTTP Authorization header using the HTTP Basic authentication scheme or as a form POST. Examples of both options are documented below. The client_id and client_secret must be separated by a single colon (":") character and encoded within a base64-encoded string, as required by IETF RFC 2617.

Your consumer application must:

- Send the request with the X.509 certificate provided during registration.
- Pass all parameters in a URL-encoded format with UTF-8 character encoding as specified by the HTTP header *Content-Type: application/x-www-form-urlencoded*.

Example (Curl) – Basic Auth Headers

A Curl example of how to perform a request for a Access Token from ADP

```
curl
  --request POST
  --url 'https://accounts.eu.adp.com/auth/oauth/v2/token' --cert '/path/to/cert.crt'
  --key '/path/to/key.key'
  --user ': '
  --header 'Content-Type: application/x-www-form-urlencoded'
  --data-urlencode 'grant_type=client_credentials'
```

## Example (Curl) – Form POST

A Curl example of how to perform a request for a Access Token from ADP

```
curl
  --request POST
  --url 'https://accounts.eu.adp.com/auth/oauth/v2/token' --cert '/path/to/cert.crt'
  --key '/path/to/key.key'
  --header 'Content-Type: application/x-www-form-urlencoded'
  --data-urlencode 'client_id='
  --data-urlencode 'client_secret='
  --data-urlencode 'grant_type=client_credentials'
```

## Response

```
{
  "access_token": "27b16f3c-e4f3-44c2-b800-b571424c4e88",
  "token_type": "Bearer",
```

```
  "expires_in": 3600
}
```

| Parameter | |
|---|---|
| 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. |

## How to use an access token

After your application obtains an Access Token, it can use it to request multiple Web APIs for as long as the Access Token is valid.
Your application must follow these guidelines to request a protected Web API:

- Use the Transport Client Authentication in all Web API requests with the X.509 certificate provided during the registration of your application. For example, mutual Secure Sockets Layer (SSL)/Transport Layer Security (TLS).
- Use the HTTP Authorization header with the Bearer authentication scheme and a valid access token.

The following is an example of a Web API request with a bearer token:

**GET /hr/v2/workers HTTP/ 1.1 Host : api.eu.adp.com Authorization : Bearer 024ded5f831d4483a9c606710026b09b**

Your application must be able to process the errors as defined in Internet Engineering Taskforce (IETF) Request for Comment (RFC) 6750. Your application must be able to handle the errors in the following table.

| HTTP Status | Error code | Deescription |
|---|---|---|
| 400 Bad Request | invalid_request | Specifies that the Web API request is missing a required parameter or includes an invalid or distorted parameter value. |
| 401 Unauthorized | invalid_token | Indicates the Web API request contained an expired, revoked, or invalid Access Token. Additional information about the error might be provided in the WWW- Authenticate header. |
| 403 Forbidden | insufficient scope | Specifies that the Web API request requires higher privileges than provided by the Access Token. |
| 500 Internal Server Error | | Indicates the Web API cannot be processed due to a runtime error. |
| 503 Service Unavailable | | Indicates the Web API cannot be processed because the service is unavailable. |

The following is an example of an error:

**HTTP/ 1.1 401 Unauthorized WWW- Authenticate : Bearer realm="oauth", error="invalid_token", error_description=" Access token expired"**

For other errors, the authorization server might provide the error code and error_description in the body as a JavaScript Object Notation (JSON) object.

The following is another example of an error:
**HTTP/ 1.1 403 Forbidden Content-Type : application/json;charset=UTF- 8 { "error":"insufficient_scope", "error_description":" Unauthorized Web API"**

For other errors, the authorization server might provide the error code and error description in the body as a JSON object.

## ADP Web API Limitations

For security reasons ADP places limits on the usage of Access Tokens and Web APIs. You may want to consider the following limitations in your application:

- Access Tokens are tied to your application. You cannot use one Access Token in another application.
- Access Tokens may be tied to the computing environment used by your application during the authorization process. For example, restricted to a certain IP address. Access might not be allowed from a different computing environment.
- Access Tokens have a short life span. By default, Access Tokens expire in 60 minutes. If your application and security requirements require a smaller expiration time, indicate your requirements during application registration.

### Tip

Since tokens expire after 60 minutes, your application should only request a different token when the current one is about to expire. Do not call a new token for every API request.

If your application and security requirements require a different setting, indicate your requirements during application registration.

## Security Considerations

ADP recommends your organization considers keeping application credentials under strict control to prevent leakage and misuse . This includes the following:

- Account identifier
- Account secret
- X.509 certificate

Your application should never store or pass Access Tokens in cookies or parameters. They are transient and stored temporarily in your application's memory.
You should do the following:

- Discard Access Tokens when your application finishes its processing with ADP.
- Consider limiting the access scope requested in authorization requests. This limits the capabilities associated with Access Tokens produced by these authorization requests.