DocuSign | Developer Center

APIs

APIs Overview

eSignature REST API

eSignature SOAP API

Click API

Organization Admin API

DocuSign Signature Appliance

TSP Partner Program

DocuSign for Salesforce

Resources

Blog

Live Webinar

Tools

Training

API Success Stories

Community

Stack Overflow

Developer Events

Search

eSignature REST API

Overview Guides Code Examples API Reference SDKs & Tools Support

Build It.

Authorization Code Grant

The **Authorization Code Grant** is an OAuth 2.0 flow that is used to grant an access token to server-hosted applications. Each request to the DocuSign APIs must include a valid access token.

In this flow, your client application requests an authorization code which can be exchanged for the access token that it needs to make API calls. It differs from the Implicit Grant in the following ways:

- The transaction is more secure, but the client must be able to store the resulting access token on a secure server. Because the access token granted in this flow is passed directly to the web server hosting the client (rather than going through the user's device via their web browser,

Code Examples:

The following code examples are working, runnable example programs that use the Authentication Code Grant flow:

MVP Program

Developer Spotlight

Pricing

API Plans

Contact Sales

as in the Implicit Grant), this flow does not rely on the client device being secure.

- The additional step of requesting and exchanging the authorization code adds a small amount of overhead, compared to simply requesting an access token.
- After authentication, your application is also granted a refresh token, which they can use to refresh their access token when it is close to expiration. For details, see Using Refresh Tokens.

Because the authorization code grant is the most secure flow, DocuSign recommends that it be used by user applications wherever possible. For user applications that are only able to use local storage, use the Implicit Grant instead.

- C# example project
- Node.js example project
- Java example project
- Python example project
- PHP example project
- Ruby example project
- Curl example project

Note: This topic describes how to authenticate using the DocuSign eSignature API. If you are using one of the DocuSign SDKs, the methods and syntax used for each step may vary. See the DocuSign SDK documentation for details.

Prerequisites

- Your application has an Integration Key
- Your application has defined a redirect URI for your Integration Key
- Your application has a Secret Key

Step 1: Request the Authorization Code

To begin the Authorization Code flow, your application redirects the user's browser to the DocuSign authorization URI, where they will grant consent and obtain an authorization code. Note that this is **not** a standard GET request and cannot be directly sent by the application. Instead, the user's browser is redirected to an authorization request URI and the request is sent from there to the account server.

- For the developer sandbox environment, the base URI is https://account-d.docusign.com/oauth/auth
- For the production platform, the base URI is https://account.docusign.com/oauth/auth

Login URI syntax

```
https://account-d.docusign.com/oauth/auth?
        response_type=code
        &scope=YOUR_REQUESTED_SCOPES
        &client_id=YOUR_INTEGRATOR_KEY
        &state=YOUR_CUSTOM_STATE
        &redirect_uri=YOUR_REDIRECT_URI
```

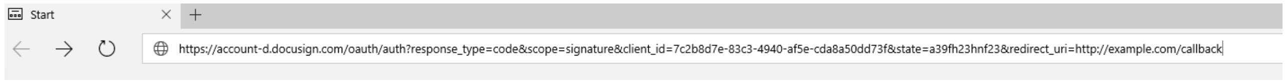
The authorization request includes several parameters used in conjunction with the base URL that configure specific authentication options:

Parameter	Required?	Description
response_type	Yes	The type of grant to be used.

		For the Implicit Grant, this value should be `token` . For the Authorization Code grant, this value should be `code` .
scope	Yes	The scope of the permissions being requested for the client is a space-separated list of the following values: • `signature` — Allows your application to create and send envelopes, and obtain links for starting signing sessions. • `extended` — Issues your application a refresh token that can be used any number of times. This scope may only be granted in the Authorization Code flow. • `impersonation` — Allows your application to access a user's account and act on their behalf even when that user is not present. This scope is only used by the JSON Web Token Grant flow. **Note:** Encode the space character as %20 in URIs
client_id	Yes	Your application's Integration Key.
prompt	No	If set to `login` then the authentication server will prompt the user for re-authentication, even if they have an active login session If the parameter is omitted, then the user may not be prompted (silent authentication) if they have an active

		login session.
state	No	Holds a special implementation specific value that is returned with the authorization code. Although the state parameter is optional, we strongly recommend that you use it to avoid cross-site request forgery attacks. You should set the state parameter to a secure string known only to your application, and check to make sure that the value has not changed between requests and responses. See the following sections in the OAuth2 RFCs to learn more: • Cross-Site Request Forgery in The OAuth 2.0 Authorization Framework • state Parameter in OAuth 2.0 Threat Model and Security Considerations
redirect_uri	Yes	The URI to which DocuSign will redirect the browser after authorization has been granted by the user. The redirect URI must exactly match one of those pre-registered for the integrator key in your DocuSign account. This determines where to redirect the user after they successfully authenticate.

Your application should open the resulting URI in a web browser to initiate the authentication process. You can test the process by pasting your URI string into a browser window, similar to the one shown below.

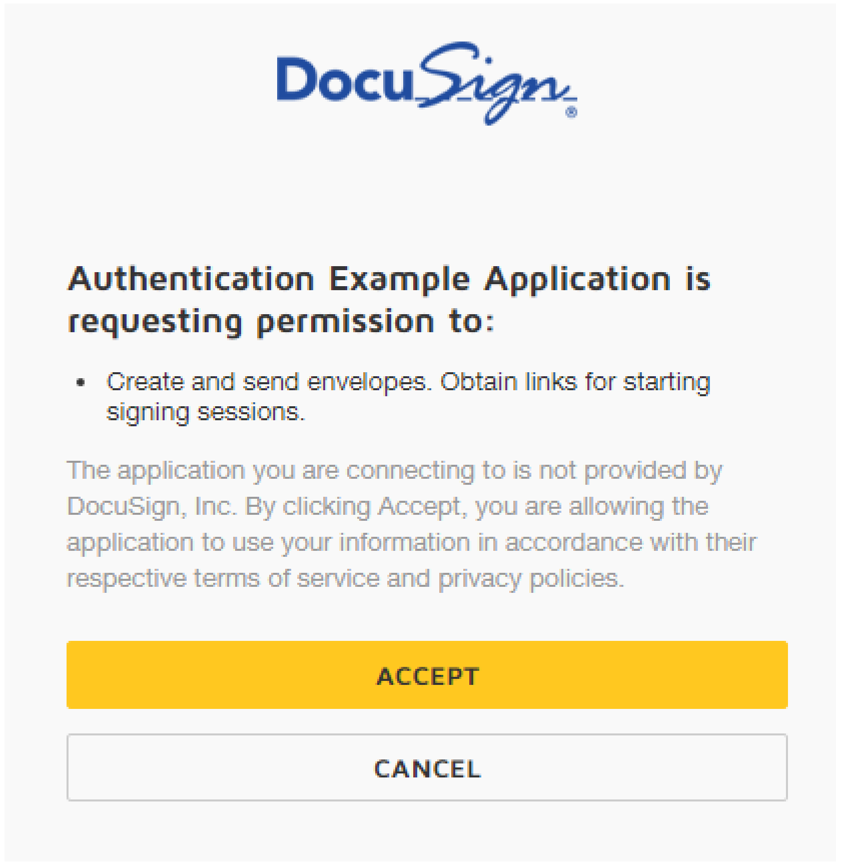


https://account-d.docusign.com/oauth/auth?response_type=code&scope=signature&client_id=7c2b8d7e-83c3-4940-af5e-cda8a50dd73f&state=a39fh23hnf23&redirect_uri=http://example.com/callback

Example login URI

```
https://account-d.docusign.com/oauth/auth?response_type=code&scope=signature&clie
```

If the user hasn’t already given consent (or if consent has been revoked using the DocuSign web app), they are notified that they must give consent to the application for the requested scopes. See Obtaining Consent for details.



Authentication Example Application is requesting permission to:

- Create and send envelopes. Obtain links for starting signing sessions.

The application you are connecting to is not provided by DocuSign, Inc. By clicking Accept, you are allowing the application to use your information in accordance with their respective terms of service and privacy policies.

After consent has been granted, the Authentication Service verifies that the client application is valid and has access to the requested scope. If so, it returns the authentication code to the provided callback URI as a query parameter.

Example response

```
http://example.com/callback/?code=eyJ0eXAiOiJNVCIsImFsZyI6IlJTMjU2Iiwia2lkIjoiNjg
```

Note: The response's code parameter is always present. The state parameter is present only if you provided it in the initial request.

Step 2: Obtain the Access Token

Once you have an authorization code, use the authentication service /oauth/token endpoint to obtain access and refresh tokens. You will use the access token to make API calls in a later step.

- For the developer sandbox environment, the token URI is https://account-d.docusign.com/oauth/token
- For the production platform, the token URI is https://account.docusign.com/oauth/token

Note: The obtained authorization code is only viable for 2 minutes. If more then two minutes pass between obtaining the authorization code and attempting to exchange it for an access token, the operation will fail.

Request syntax

```
curl --header "Authorization: Basic BASE64_COMBINATION_OF_INTEGRATOR_AND_SECRET_K
    --data "grant_type=authorization_code&code=YOUR_AUTHORIZATION_CODE"
    --request POST https://account-d.docusign.com/oauth/token
```

Each Access token request contains the following headers

Name	Description

Content-Type	A standard HTTP entity header indicating the media type of the resource. The value of Content-Typ `application/x-www-form-urlencoded`

Authorization	The Authorization header contains your Integrator Key and secret key, concatenated by a colon ch into base64, and prefixed with the word Basic. This value, together with the authorization code, is access token.
	
	For example, if your Integrator Key is `7c2b8d7e-83c3-4940-af5e-cda8a50dd73f` and the secret key is `d7( 6842b7aa8861` you can get the base64 value in a JavaScript console with the following method call:
	
	`btoa('7c2b8d7e-83c3-4940-af5e-cda8a50dd73f:d7014634-3919-46f6-b766-6842b7aa8861')`
	
	This method call results in a new authorization header value:
	
	`NWMyYjhkN2UtODNjMy00OTQwLWFmNWUtY2RhOGE1MGRkNzNmOmQ3MDE0NjM0LTM5MTktNDZmNi1iNzY2L`

In addition to the headers, the access token request contains a set of body parameters:

Name	Description
grant_type	The type of grant being used. To exchange an authorization code for an access token, use authorization_code.
code	The authorization code that you received from the callback.
redirect_uri	The URI to which DocuSign will redirect the browser after authorization has been granted by the user. The redirect URI is optional, but DocuSign recommends that you include it.

Example request

```
    curl --header "Authorization: Basic NWMyYjhkN2UtODNjMy00OTQwLWFmNWUtY2RhOGE1MGRkNzNmO
         --data "grant_type=authorization_code&code=eyJ0eXAiOiJNVCIsImFsZyI6IlJTMjU2Iiwia
        --request POST https://account-d.docusign.com/oauth/token
```

If the authorization succeeds, access and refresh tokens are returned as a JSON object in the response, as shown below.

Response structure

```
{
  "access_token": "ISSUED_ACCESS_TOKEN",
  "token_type": "Bearer",
  "refresh_token": "ISSUED_REFRESH_TOKEN",
  "expires_in": 28800
}
```

Name	Description
access_token	The value of the access token. You must use this token in the Authorization header of all DocuSign API calls
token_type	The type of token. For access tokens, the value of this is `Bearer` .
refresh_token	A token that you can use to obtain a new access token without requiring user consent. The lifetime of a refresh token (typically around 30 days) can vary depending on business needs and is subject to change at any time. When you use a refresh token to

	authenticate, you can obtain a new refresh token with the following behavior: • If you have obtained the extended scope, a new refresh token is issued with a full lifetime (typically 30 days). • If you do not have extended scope, a new refresh token is issued that has the same expiration date as the original token (typically 30 days from first login)
expires_in	The number of seconds before the access token expires.

Example response

```
{
"access_token":"eyJ0eXAiOiJNVCIsImFsZyI6IlJTMjU2Iiwia2lkIjoiNjgxODVmZjEtNGU1MS00Y2U5LWFmM
"token_type":"Bearer",
"refresh_token":"eyJ0eXAiOiJNVCIsImFsZyI6IlJTMjU2Iiwia2lkIjoiNjgxODVmZjEtNGU1MS00Y2U5LWFm
"expires_in":28800
}
```

Step 3: Retrieve User Account Data

To make an API call to the DocuSign platform, your application needs both an access token (which you obtained in the previous step), and base URI that is unique to the user on whose behalf your application is making the API call.

To get the base URI, call the /oauth/userinfo endpoint, supplying your application's access token as a header.

- For the developer sandbox environment, the URI is https://account-d.docusign.com/oauth/userinfo
- For the production environment, the URI is https://account.docusign.com/oauth/userinfo

Request syntax

```
GET https://account-d.docusign.com/oauth/userinfo
        Authorization:Bearer YOUR_ACCESS_TOKEN
```

The request must contain an Authorization header that holds your access token.

Name	Description
Authorization	Contains the word Bearer and the access token that you received in the previous step.

Example request

```
curl --request GET https://account-d.docusign.com/oauth/userinfo
    --header "Authorization: Bearer eyJ0eXAiOiJNVCIsImFsZyI6IlJTMjU2Iiwia2lkIjoiN
```

If successful, this returns a set of data for each user who has granted consent and the account(s) that that user is a member of, as shown by the following example response.

Example response

```
{
    "sub": "564f7988-0823-409a-ac8a-781ee556ab7a",
    "name": "Example J Smith",
    "given_name": "Example",
    "family_name": "Smith",
    "created": "2018-04-13T22:03:03.45",
    "email": "Example.Smith@exampledomain.com",
    "accounts": [
        {
            "account_id": "18b4799a-b53a-4475-ba4d-b5b4b8a97604",
            "is_default": true,
            "account_name": "ExampleAccount",
            "base_uri": "https://demo.docusign.net"
        }
    ]
}
```

You will need the `account_id` and the `base_uri` claims of the user that your application is acting on behalf on to make calls to the DocuSign API.

For details on the `oauth/userinfo` method and its response object, see UserInfo Endpoint Reference.

Step 4: Call the eSignature REST API

Now that you've obtained an access token the user info, your application is authenticated and has all of the information that it needs to make a call to a DocuSign API. To make an eSignature REST API request, you must include:

- The acquired access token in the request's Authorization header
- The base URI for the request, built from the `account_id` and the `base_uri` of the user account

For example, the syntax to return a set of data for each brand associated with a user is:

```
GET restapi/v2/accounts/{accountId}/brands
```

Using the base URI of `https://demo.docusign.net` and account ID of `18b4799a-b53a-4475-ba4d-b5b4b8a97604` from the above response, the request path would be:

```
GET https://demo.docusign.net/restapi/v2/accounts/18b4799a-b53a-4475-ba4d-b5b4b8a
```

The final piece of information needed to access the API is the Access Token, prefixed with “Bearer” and provided in the Authorization header.

Example eSignature REST API request

```
curl -- request GET https://account-d.docusign.com/restapi/v2/accounts/18b4799a-b
      --header "Authorization: Bearer eyJ0eXAiOiJNVCIsImFsZyI6IlJTMjU2Iiwia2lkIjo
```

Using Refresh Tokens

You can use a refresh token to get new access token after the access token expires. Depending on account and system security policies, refresh tokens may have a longer lifetime than access tokens. Refresh tokens usually do not require the account user to log in again.

To get a new access token, use the refresh token as you would an authorization code, but with a grant_type value of refresh_token and a refresh_token parameter that holds the contents of the refresh token.

Request syntax

```
        POST https://account-d.docusign.com/oauth/token
                Authorization: Basic BASE64_COMBINATION_OF_INTEGRATOR_AND_SECRET_KEYS

                grant_type=refresh_token&refresh_token=YOUR_REFRESH_TOKEN
```

Name	Description
grant_type	The type of grant being used. To exchange an refresh token for an access token, use refresh_token
refresh_token	The full refresh token value that you originally received from authentication.

Example request

```
curl --header "Authorization: Basic MjMwNTQ2YTctOWM1NS00MGFkLThmYmYtYWYyMDVkNTQ5NGFkOjMwO
```

Refresh tokens may be used for a variable period of time (that partially depends on business needs and may change at any time, but typically about 30 days) after the first authentication. When your application uses the refresh token to authenticate, it will be issued a new refresh token.

- If you have obtained the extended scope, a new refresh token is issued with a full lifetime (typically 30 days). It is possible for a user to remain logged-in indefinitely by requesting the extended scope and continually refreshing its access token.
- If you do not have extended scope, a new refresh token is issued that has the same expiration date as the original token (typically 30 days from first login).
- If you receive any errors during the refresh operation, it should prompt the user to log in again to obtain a new access and refresh token.

Instead of waiting for an access token to expire, we recommend that you refresh the access token when it is close to expiring (within 30 minutes, for example). When you first get the access token, use the `expires_in` value to calculate an expiration time. Store this expiration time in the user's session.

Troubleshooting Errors

Invalid authentication request: The response type is not supported Indicates that the application attempting to use JWT authentication is configured to use Implicit Grant authentication. It must be set to use Authorization Code Grant.

To change the Authentication setting:

1. Log in to your DocuSign admin account

2. Under **Integrations**, select **API and Keys**
3. Under **My Apps / Integration Keys**, choose the Integrator Key to use, then select **Actions**, then **Edit**
4. Select **Authorization Code Grant**.

Bad Request

Ensure that your application is attempting to connect to the correct authentication URI.

- For the developer sandbox environment, you should use URIs from the `https://account-d.docusign.com/oauth` domain.
- For the production platform, you should use URIs from the `https://account.docusign.com/oauth` domain.

Invalid authentication request: The response type is not supported.

Indicates that the response type specified in the authorization code request is not set to `code` . Ensure that the value of the `response_type` header is `code` .

Invalid Grant

Indicates that the supplied authorization code is incomplete or expired. Ensure that there are no extra spaces on the start and end portions of the string, and that no characters have been truncated. If you used the state parameter, ensure that you've removed it from the end of the returned code value.
If you encounter this error repeatedly and/or after verifying that all parameters are correct, ask the user to log in again.

Invalid RedirectUri
Indicates that the redirect URI specified for authentication does not match any of the redirect URIs defined for the integrator key. The redirect URI strings must match exactly, including space and slash characters.

Consent_required
Indicates that the user has not consented to allow your application to make calls on their behalf.

If the user has supplied consent and are still receiving this error, there may have been something wrong with the URI being used. Double check to make sure the Integrator Key has not been cut off.

Page cannot be displayed after requesting the code
Indicates that the redirect URI is not set to a valid web page.