OAuth 2.0 Authentication
Introduction to OAuth 2.0
Understanding OAuth 2.0 Roles
OAuth 2.0 Access Tokens
Managing OAuth Credentials and Redirect URIs
Choosing an OAuth 2.0 Grant Type
OAuth 2.0 Using Authorization Code Grant
OAuth 2.0 Using Implicit Grant
OAuth 2.0 Using Client Credentials
OAuth 2.0 for Installed Applications
Procore API Authentication Endpoints
API Essentials
RESTful API Concepts
API Lifecycle

Using the OAuth 2.0 Client Credentials Grant Type

Introduction

The Client Credentials flow is perhaps the most simple of the OAuth 2.0 flows supported by the Procore API. The primary difference with the Client Credentials flow is that it is not associated with a specific Procore user (resource owner). In addition, it is not necessary to first obtain an authorization code before retrieving an access token when using the the Client Credentials grant type. The Procore API implementation of the Client Credentials flow relies on the use of a Procore Service Account.

The Client Credentials grant type provides a specific grant flow in which the resource owner (that is, the user) is not involved. When using this grant with Procore, the client application requests an access token using only its own credentials (via an existing Service Account), and uses the access token on behalf of the client application itself. This grant flow is best-suited for API methods that are used by the client application in general, instead of methods that apply to a certain resource owner, for example, API methods for reporting, analytics, administrative tasks, system maintenance, etc. This method for using an API is also referred to as *userless access*.

Common Use Cases

There are a number of scenarios in which using the Client Credentials grant flow in the Procore API is the preferred approach.

- "Machine-to-machine" integrations - where a specific user's permission to access their data is not required (i.e., non-delegated authorization)
- Report generators, data mining, or other integrations that access company-wide data
- Backend scripts, system maintenance and administration utilities

> CHOOSING THE CLIENT CREDENTIALS GRANT FLOW
>
> The Client Credentials grant flow is not intended to replace the Installed Applications flow. It should only be used for integrations and applications that perform actions not tied to, or on bahalf of, a specific Procore user.

Security Considerations

Depending on how you plan to use the Client Credentials grant flow, the credentials for your application (Client ID and Client Secret) could potentially provide access to a large amount of data. The more data a single set of credentials has access to, the greater the risk if the credentials become compromised. Therefore, it is imperative that the credentials used to authenticate your application with Procore are kept confidential. Ideally, these credentials would also be rotated regularly.

It is important to note that Service Accounts are set to read-only access on company-level tools by default. However, you can refine and customize the access levels for your Service Account as a means for establishing and enforcing proper security policies for your application. Please see Configure Service Account Permissions on the Procore Support Site for more information.

Client Credentials Grant Flow Example

The following sections outline the basic building blocks for implementing the Client Credentials grant flow in your application.

> PRODUCTION VS. SANDBOX ENVIRONMENTS
>
> It is important to note that access/refresh tokens are not shared across production and sandbox environments. API calls you make to Procore resources in your production environment must use a separate set of authentication keys (client ID and client secret) from those used for your sandbox environment. In addition, API calls to production must use the `https://api.procore.com` base URL, while calls to your sandbox must use the `https://sandbox.procore.com` base URL. Keep in mind that the examples presented below use the production base URL, rather than the sandbox base URL.

1. Create Service Account in Procore

The first step to implementing the Client Credentials grant flow in your application is to log in to Procore and create a Service Account as descibed in our support article. Once the Service Account is created, you will have access to the `client_id` and `client_secret` generated by the system.

2. Retrieve Access Token

Once you have created a Service Account in Procore, you can retrieve an OAuth 2.0 access token by making a POST call to the /oauth/token endpoint in the Procore API. The JSON payload for this call is formatted as follows:

```
{
  "grant_type": "client_credentials",
  "client_id": "242635f69bfc6fb9adax513875a0254a2a908f7bb176x1698d6x169a08f5646d",
  "client_secret": "041372a09c6e4x92aa08ce3axea1cfc0115cbfbf7134exa125fe10cc103ca626"
}
```

And here is a cURL example for calling the /oauth/token endpoint:

```
curl -F grant_type=client_credentials \
     -F client_id=242635f69bfc6fb9adax513875a0254a2a908f7bb176x1698d6x169a08f5646d \
     -F client_secret=041372a09c6e4x92aa08ce3axea1cfc0115cbfbf7134exa125fe10cc103ca626 \
     -X POST https://login.procore.com/oauth/token
```

The JSON response block from the call includes an `access_token` string and addtional attributes as shown here:

```
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzUxMiJ9xeyJhaWQiOiIwY2I3ZDc1...",
  "token_type": "bearer",
  "expires_in": 7200,
  "created_at": 1511289006
}
```

- `token_type` - value of "bearer" indicates that the Authorization header you pass when making Procore API calls will be the "bearer" type.
- `expires_in` - value of 7200 indicates that the access token is valid for 7200 seconds (2 hours).
- `created_at` - date/time the access token was genterated in UNIX time format.

3. Make a Test Call to the Procore API

Now that you have a valid access token you can make a test call to the Procore API. For this simple example, we will make a call to the List Projects endpoint to return a list of the active projects in our company.

```
curl -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzUxMiJ9xeyJhaWQiOiIwY2I..." \
     -X GET https://api.procore.com/vapid/projects?company_id=7656
```

If successful, this call will return a JSON response block similar to the following:

```
{
  "id": 123456,
  "name": "Demo Account Project",
  "display_name": "Demo Account Project",
  "project_number": null,
  "address": "5678 First Street",
  "city": "Anytown",
  "state_code": "CA",
  "country_code": "US",
  "zip": "93013",
  "county": null,
  "latitude": 34.385045633646,
  "longitude": -119.490841957738,
  "stage": "Course of Construction",
  "phone": "",
  "created_at": "2016-08-22T20:18:55Z",
  "updated_at": "2017-10-23T21:19:34Z",
  "active": true,
  "origin_id": null,
  "origin_data": null,
  "company": {
      "id": 1234,
      "name": "My Construction Company"
  }
}
```

Using Service Accounts with MPZ

If you intend to use Service Accounts with Multiple Procore Zones (MPZ), you *must* include the `Procore-Company-Id` request header when making calls to the `/vapid/me` or `/vapid/companies` endpoints.

Here is a cURL example showing a call to `GET /vapid/me`:

```
curl -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzUxMiJ9xeyJhaWQiOiIwY2I..." \
     -H "Procore-Company-Id: xxxxxxxxx" \
     -X GET https://api.procore.com/vapid/me
```

Here is a cURL example showing a call to `GET /vapid/companies`:

```
curl -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzUxMiJ9xeyJhaWQiOiIwY2I..." \
     -H "Procore-Company-Id: xxxxxxxxx" \
     -X GET https://api.procore.com/vapid/companies
```

The value to use for the `Procore-Company-Id` header can be retrieved from the URL in the address bar of your browser once you have logged into a company. The company ID displays in the URL as a path parameter as shown in the following example.

`https://app.procore.com/<COMPANY ID>/company/home/list`

Where: <COMPANY ID> is the integer value for the ID of the company you are currently logged into.

For addtional information on MPZ, see Working with Multiple Procore Zones.