# Instructions for Machine-to-Machine Communication

This is the client ID and client secret that will be used to authenticate machine to machine. This should be kept secure and in a safe place. Do not expose this information to the public.

**Client Id: accessMedicarePushServer**
**Client Secret: ef3684401523269b56c0669a529df6ffa96fd56035fb8c5b4347fc4a7c64374d**

## Application Domains

**Development Domain**: access-medicare.zdidesign.com
*used only during early development*

**Test Harness Domain**: test.accessmedicareny.com
*used for testing new features before going live, can be whatever client chooses*

**Live Domain**: app.accessmedicareny.com
*Used for the live application, can be whatever client chooses*

If a client secret is compromised, you will need to contact ZDI to generate a new key.
*NOTE: These instructions are never included in the online api documentation for security reasons.*

Before your code can post or retrieve data from the API, you must receive an access token. The purpose of this token is for the API to know that the machine is authorized to communicate with the API.
The API always inspects the "Authorization" header of any request and if it is not present or the token is invalid, access is denied.

An access token should be retrieved at the start of any new software resource and only needs to be retrieved once. You do not need an access token before each and every call to the API. You should create the token once at the beginning of your program.

You can get an access token by making a "POST" request to http(s)://domain.com/api/oauth with the following JSON data:

```
{
      "client_id":"accessMedicarePushServer",
      "grant_type":"client_credentials",
      "client_secret":"ef3684401523269b56c0669a529df6ffa96fd56035fb8c5b434
7fc4a7c64374d"
}
```

The API will not automatically try to detect the type of data you are "POST"ing for security reasons. Instead, you must tell the API what the format of the posted data is by setting the "Content-Type" header:

```
Content-Type:application/json
```

Additionally, you must tell the API the format in which you want responses. You do this by setting the "Accept" Header:

```
Accept:application/json
```

The completed request would look similar to:

```
POST /api/oauth HTTP/1.1
Content-Type: application/json
Accept: application/json
Cookie: PHPSESSID=nq2s8be9itph31rklhab4vqln1
Host: domain.com
Connection: close
User-Agent: Paw/2.2.5 (Macintosh; OS X/10.10.5) GCDHTTPRequest
Content-Length: 164
{
      "client_id":"accessMedicarePushServer",
      "grant_type":"client_credentials",
      "client_secret":"ef3684401523269b56c0669a529df6ffa96fd56035fb8c5b434
7fc4a7c64374d"
}
```

When you receive a token/response from the server, you should always check the response status code to see if the request was successful. Check the documentation for the class/object/language that you are using to inspect the response.

If you receive a response with a status code of 200, the authentication was successful. Included in this response will be your access token and some additional information. This JSON data will look similar to:

```
{
  "access_token": "cd8812b9a4cc471d9fcd8cd09881ea81cd79f7f9",
  "expires_in": 2592000,
  "token_type": "Bearer",
  "scope": "pushServer"
}
```

You should store the access token in a variable so that you can use it for subsequent requests.

You can now make as many machine-to-machine requests to the API as you like.

Other possible response status codes are:
**400** - Client Credentials are invalid or invalid grant type supplied
**500** - Server Error - should contact ZDI

Tokens automatically come with a lifetime and the lifetime is described in seconds from the response. Upon inspecting the token response, note a parameter "expires_in". This is how many seconds from now that the token will expire.

You can opt to store tokens in a database or the like and only generate a new token when the old one expires. However, for increased security it is recommended that you revoke access tokens after you have completed your API queries.

You can revoke a token by passing the token as part of the url, as a "DELETE" request to:

```
http://domain.com/api/oauth/revoke-token/:token
```

Where ":token" would be the access token you are looking to revoke.

For all other operations, please see the online api documentation.