Merchant Services API – OAuth Guide

May 2024 **v2.1**

J.P. Morgan & Co.

OAuth Process Overview

This guide provides the steps necessary to use OAuth as an authentication method. These steps are similar for both Customer Acceptance Testing (CAT) and Production (PROD), however client certificates, client IDs, public certificates, private keys, and Application Programming Interface (API) endpoints are different between CAT and PROD.
The URL used to retrieve Access Tokens is the same for both environments.

CAT:

1. Generate a Certificate Signing Request (CSR) or Self-Signed Certificate, and provide the CSR/Cert to your J.P. Morgan Technical Representative via Email.
   This CSR/Cert will be used to generate your CAT Client ID & Public Certificate, which will then be provided to you via email, alongside the token Uniform Resource Indicator (URI).
   NOTE: Please feel free to leverage the J.P. Morgan JavaScript SDK or Java SDK to generate your CSR or Self-Signed Certificate.
2. Use your private key and public certificate to sign the JSON Web Token (JWT).
3. Send a request to the J.P. Morgan Token URI to retrieve an access token.
4. Use the access token to send requests to the Merchant Services API endpoints.

PROD:

1. Generate a Certificate Signing Request (CSR) or Certificate signed by a Certificate Authority (CA), and provide the CSR/Cert to your J.P. Morgan Technical Representative via Email.
   This CSR/Cert will be used to generate your PROD Client ID & Public Certificate, which will then be provided to you via email, alongside the token Uniform Resource Indicator (URI).
2. Use your private key and public certificate to sign the JSON Web Token (JWT).
3. Send a request to the J.P. Morgan Token URI to retrieve an access token.
4. Use the access token to send requests to the Merchant Services API endpoints.

Step 1: Generate a client certificate request or self-signed certificate using the provided node script

***OpenSSL command instructions/example also included below the SDK/Node.js instructions*

The following section describes how to use the sample node script *(provided to generate a self-signed certificate)* that can be used in CAT, or a CSR that can be used to generate an X.509 certificate for use in CAT/PROD. This guide assumes that you are using a compatible version of node, but should work similarly on any platform/OS with NPM available via the command line.

For CAT, either a CSR or a self-signed certificate is acceptable.

For PROD, either a CSR or a certificate signed by one of the following J.P. Morgan approved CA's is acceptable:

- Entrust
- DigiCert (formerly Symantec)
- GoDaddy Group
- GlobalSign
- Thawte

Note: If using a CA signed X.509 certificate, it is the merchant's responsibility to ensure the certificate is updated before it is expired. New certificates will need to be provided to J.P. Morgan to prevent service interruptions. If using a CSR, J.P. Morgan will reach out to request an updated CSR prior to the merchant certificate expiring.

1.1: Confirm Node.js Installation

Check whether Node.js is installed by using the following command. The output of this command should display the Node.js version you are using.
For example:

```
$ node
Welcome to Node.js v14.16.1.
```

If you do not have Node.js installed on your computer, the output will be similar to the following:

```
$ node
command not found: node
```

This response means that you do not have Node.js installed on your machine, or it is not in your current executable **$PATH**. Consult the NodeJS Docs for instructions regarding how to install this software on your platform.

1.2: Generate a CSR or self-signed certificate

Download and unzip the J.P. Morgan JavaScript SDK file onto your machine. Open the terminal (or shell), navigate to the project's base directory, and then run the following command to install any required node dependencies:

```
$ npm install
```

Run the **generateCertificates.js** script to generate a client certificate request and a self-signed certificate using the following command:

```
$ node generateCertificates.js
```

After running the command, a series of prompts will display, prompting you to enter your country, state, city, organization, organization unit, and common name. This information will be used to construct your client certificate request and self-signed certificate.

```
Country Name (2 letter code): US
State or Province Name (2 letter code): FL
Locality Name (eg, city): Tampa
Organization Name (eg, company): SampleOrganization
Organizational Unit Name (eg, section): SampleUnit Common
Name (eg, identifier): SampleCommon
```

If successful, you should see the following output. The corresponding **certificateRequest.pem** (CSR), **selfSignedCertificate.pem** and **PrivateKey.key** files are created within project **config** directory.

```
Generating 2048-bit key-pair...
Key-pair created successfully...
Creating certification request...
Certification request created...
Creating self-signed certificate... Certificate
created...
```

****Using OpenSSL**

OpenSSL may also be used to generate the CSR, rather than leveraging the SDK.

Below is a sample openssl command you may use to generate the CSR:

```
openssl req -new -newkey rsa:2048 -nodes -out <filename>.txt -keyout <filename>.key -subj
"/C=<Country>/ST= <State>/L=<City>/O=<Organization>/OU=<Organization Unit>/CN=<Common
Name>"
```

- Filename.txt: This file contains your CSR that needs to be zipped and shared with JPMorgan.
- Filename.key: This file contains your private key. DO NOT Send to JPMorgan.
- Country: must be two letter code
- Common Name - CN must be unique (Must be different for CAT/Test and Production)

1.3: Share your client certificate with J.P. Morgan

Email your CSR or certificate (self-signed for CAT, CA-signed for PROD) from Step 1.2 to your J.P. Morgan Technical Representative.

Once we receive your CSR/Cert, we will create an OAuth profile, and generate a Client ID and public certificate. We will then email the client ID, public certificate, resource ID, and token URI to you once this step is complete.

Step 2: Get Access Token

Once you have received your connectivity information from the onboarding team, you must retrieve the access token prior to making any requests to the API. Outlined below are two solutions that can be used.

Note: The client certificate, client ID, public certificate, and API endpoints are different for CAT and PROD.

Solution 1: Sign the JWT with your private key and call the API to retrieve an access token via the getAccessToken.js script

Reference the provided **getAccessToken.js** script found in the JavaScript SDK, and use the enclosed instructions to sign your JWT, retrieve your access token, and begin making API calls to the API resource endpoint.
Note: The access token is valid only for 8 hours.

2.1: Update the configuration file

Update the **idanywhereConfig.json** configuration file within **config** project directory.

- `client_id`: Provided by J.P. Morgan
- `certPath`: Filename containing public certificate (**.cer**) provided by J.P. Morgan
- `privateKey`: Filename containing private key
- `resource_id`: Provided by J.P. Morgan

```
{
 "client_id": "<Enter data>",
 "certPath": "./config/<Enter filename>.cer",
 "jti": "12345",
 "privateKey": "./config/<Enter filename>.key",  "audience":
"https://idag2.jpmorganchase.com/adfs/oauth2/token",
 "expiresIn": "8h",
 "algorithm": "RS256",
 "client_assertion_type": "urn:ietf:params:oauth:client-assertion-type:jwt-bearer",
 "grant_type": "client_credentials",
 "resource_id": "<Enter data>",
 "ida_url": "https://idag2.jpmorganchase.com/adfs/oauth2/token",
 "content_type": "application/x-www-form-urlencoded" }
```

2.2: Execute a script to obtain access token

Use the following command to run the **getAccessToken.js** script to sign the JWT and to retrieve the access token:

```
$ node getAccessToken.js
```

Example of **access_token** returned:


```
{  "access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI9IwJEYXo0Ke5jZ0Y2eHRsY0RfVDl6bzFUU1BaCDJ8.eyJhdWQiOiJKU
E1DOlVSSTpSUy0xMDMyNTctMjQ5KWEtQ29ubmVjdFBheW1lbnRzVWF0QXBwLVVBVCIsImlzcyI6Imh0dHA6Ly9pZGF1YXQuanBt
b3JnYW5jaGFzZS5jb20vYWRmcy9zZXJ2aWNlcy90cnVzdCIsImlhdCI6MTU3NDM3MzU3NCwiZXhwIjoxNTc0NDAyMzc0LCJKUE1
DSWRlbnRpZmllciI6IkY2Nzc4NDQiLCJSb2xlIjpbIk1TQ29ubmVjdFVhdENvbnRyb2xHcm91cC0yNDk0MS0xMDMyNTctVUFUIi
wiQ29ubmVjdFBheW1lbnRzVWF0QXBwX0FMTC0yNDk0MS0xMDMyNTctVUFUIl0sIkNsaWVudElQQWRkcmVzcyI6IjE3Mi4yNi4xN
jMuMjMyIiwiYXV0aG1ldGhvZCI6WyJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvYXV0
aGVudGljYXRpb25tZXRob2QvdGxzY2xpZW50IiwiaHR0cDovL3NjaGVtYXMubWljcm9zb2Z0LmNvbS93cy8yMDA4LzA2L2lkZW5
0aXR5L2F1dGhlbnJqY2F0aW9ubWV0aG9kL3g1MDkiXSwiYXBwdHlwZSI6IkNvbmZpZGVudGlhbCIsImFwcGlkIjoiQ0MtMTAzMj
U3LUY2Nzc4NDQtMzgzNzEtVUFUIiwiYXV0aF90aW1lIjoiMjAxOS0xMS0yMVQyMTo1OTozNC4wMTlaIiwidmVyIjoiMS4wIn0.O
1LeuTzHpCZ9eja4sBgpl9Ypf71s747alq0uOQ4XJ-
198AKRzvPzeltJotl063qfxShHrfpP1Z9jBh2bt7iu8yh3EeUZc_dpl7cjAlllZSAmKuDyP8d2evHxp0P4gfIAixTeaNBIwwbiAu
-CczN30g2RgQvWyTqhRpw2uLXvTFR4QnVSztgcAgKqq-
PmMYXACNziWqzUMGsHEK8Qw5eQP0v6pXdx4LLiJBxNIc0R1OFN-
GT8X8bsQVasHF0ASBG9UuawbpV6h3qoySgZgfDjOpJcpAHsdz4xDKNkOsMfLGah51wgNumt4wz-
S6XYkWPgobe4TOqk9FdWnimgruaOg",
 "token_type": "bearer",
 "expires_in": 28800
}
```


2.3: Using the access token

Use value returned during Step 2.2 as the bearer token to send Hypertext Transfer Protocol (HTTP) requests to Merchant Services API endpoints.

Note: Once the token has expired, an HTTP status of **401** will be returned. Repeat Step 2.2 to obtain a new access token.

Solution 2: Sign the JWT with your private key, and call the API to retrieve an access token via an *HTTP* request.

3.1: Update the configuration file

Update **idanywhereConfig.json** configuration file within the **config** project directory.

- `client_id`: Provided by J.P. Morgan
- `certPath`: Filename containing public certificate (**.cer**) provided by J.P. Morgan
- `privateKey`: Filename containing private key
- `resource_id`: Provided by J.P. Morgan

```
{
 "client_id": "<Enter data>",
 "certPath": "./config/<Enter filename>.cer",
 "jti": "12345",
 "privateKey": "./config/<Enter filename>.key",  "audience":
"https://idag2.jpmorganchase.com/adfs/oauth2/token",
 "expiresIn": "8h",
 "algorithm": "RS256",
 "client_assertion_type": "urn:ietf:params:oauth:client-assertion-type:jwt-bearer",
 "grant_type": "client_credentials",
 "resource_id": "<Enter data>",
 "ida_url": "https://idag2.jpmorganchase.com/adfs/oauth2/token",
 "content_type": "application/x-www-form-urlencoded" }
```

3.2: Execute script to generate a signed JWT

Run the **getJWT.js** script to generate a signed JWT. This value will be used in the **client_assertion** header field during the next step. Example of JWT returned:

```
$ node getJWT.js


JWT: eyJhbGciOiJSUzI1NiVmInR5cCI6IkpXVCIsImtpZCI6IkE0RDAwOTIyOEUxNEJEMDZCOUJGMzI0Qjg4
QTUzOTE4MzEwQjM5NjkifQ.eyJqdGkiOiIxMjM0NSIsImlhdCI6MTU3NDM2OTY2OSwiZXhwIjoxNTc0N
DEyODY5LCJhdWQiOiJodHRwczovL2lkYXVhdC5qcG1ckmdhbmNaYXNlLmNvbS9hZGZzL29hdXRoMi90b
2tlbiIsImlzcyI6IkNDLTEwMzI1Ny1GNjc3ODQ0LTM4MzcxLVVBVCIsInN1YiI6IkNDLTEwMzI1Ny1GN
jc3ODQ0LTM4MzcxLVVBVCJ9.gWcdqXyifWvpZlaI5gxOVyEFFDZA0dJaOVTXJu28S9p4uFG-1L8D6Wx3 6mtfZ2EZqe3SMIVaQe-
yVSZYVvJ7_58NKBCIvI6IMkguQ5sP5LiyCs34Nt2BhqOibu1hCrxqp45piHX4
1BzNki-KklAx1OICDKp3RkhhykLOV2WU5bOj6HrlIjVvJ6hNkLDL5jRY3UgBDhRa7-xzO9vJ2wvxNJGJ
F9pVHOgYIwNbvifvfC_nvxxs1GaA8gGWbPdNO4Q0nj5_pTjHhIbL3GrkXmSdp7a6JHOzS1gfYOz9MHHu
h9QIF2cqpBZipiLdjWL84ft6jI0_5kpotMGkHsNZMX4RqB
```

3.3: Use signed JWT to obtain Access Token

Send a POST request to the token URI.

Headers:

`content-type: application/x-www-form-urlencoded`

Body:

- `grant_type`: client_credentials
- `client_id`: Provided by J.P. Morgan
- `client_assertion_type`: `urn:ietf:params:oauth:client-assertiontype:jwt-bearer`
- `client_assertion`: JWT (signed by the merchant with the client ID, public certificate, and private key using **RS256** as the signature algorithm)
  Note: Ensure the value does not contain any whitespaces or new lines.
- `resource`: Provided by J.P. Morgan

Example payload:

```
grant_type:client_credentials client_id:CC-123456-F777777-12334-CAT
client_assertion_type:urn:ietf:params:oauth:client-assertion-type:jwt-bearer
client_assertion:
eyJhbGciOiJSUzI1NiVmInR5cCI6IkpXVCIsImtpZCI6IkE0RDAwOTIyOEUxNEJEMDZCOUJGMzI0Qjg4QTUzOTE4MzEwQjM5Njk
ifQ.eyJqdGkiOiIxMjM0NSIsImlhdCI6MTU3NDM2OTY2OSwiZXhwIjoxNTc0NDEyODY5LCJhdWQiOiJodHRwczovL2lkYXVhdC5
qcG1ckmdhbmNaYXNlLmNvbS9hZGZzL29hdXRoMi90b2tlbiIsImlzcyI6IkNDLTEwMzI1Ny1GNjc3ODQ0LTM4MzcxLVVBVCIsIn
N1YiI6IkNDLTEwMzI1Ny1GNjc3ODQ0LTM4MzcxLVVBVCJ9.gWcdqXyifWvpZlaI5gxOVyEFFDZA0dJaOVTXJu28S9p4uFG-
1L8D6Wx36mtfZ2EZqe3SMIVaQe-
yVSZYVvJ7_58NKBCIvI6IMkguQ5sP5LiyCs34Nt2BhqOibu1hCrxqp45piHX41BzNkiKklAx1OICDKp3RkhhykLOV2WU5bOj6Hrl
IjVvJ6hNkLDL5jRY3UgBDhRa7-
xzO9vJ2wvxNJGJF9pVHOgYIwNbvifvfC_nvxxs1GaA8gGWbPdNO4Q0nj5_pTjHhIbL3GrkXmSdp7a6JHOzS1gfYOz9MHHuh9QIF
2cqpBZipiLdjWL84ft6jI0_5kpotMGkHsNZMX4RqB
resource:JPMC:URI:RS-123456-12345-ConnectPaymentsUatApp-CAT
```

Example of **access_token** returned:


```
{  "access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI9IwJEYXo0Ke5jZ0Y2eHRsY0RfVDl6bzFUU1BaCDJ8.eyJhdWQiOiJKU
E1DOlVSSTpSUy0xMDMyNTctMjQ5KWEtQ29ubmVjdFBheW1lbnRzVWF0QXBwLVVBVCIsImlzcyI6Imh0dHA6Ly9pZGF1YXQuanBt
b3JnYW5jaGFzZS5jb20vYWRmcy9zZXJ2aWNlcy90cnVzdCIsImlhdCI6MTU3NDM3MzU3NCwiZXhwIjoxNTc0NDAyMzc0LCJKUE1
DSWRlbnRpZmllciI6IkY2Nzc4NDQiLCJSb2xlIjpbIk1TQ29ubmVjdFVhdENvbnRyb2xHcm91cC0yNDk0MS0xMDMyNTctVUFUIi
wiQ29ubmVjdFBheW1lbnRzVWF0QXBwX0FMTC0yNDk0MS0xMDMyNTctVUFUIl0sIkNsaWVudElQQWRkcmVzcyI6IjE3Mi4yNi4xN
jMuMjMyIiwiYXV0aG1ldGhvZCI6WyJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvYXV0
aGVudGljYXRpb25tZXRob2QvdGxzY2xpZW50IiwiaHR0cDovL3NjaGVtYXMubWljcm9zb2Z0LmNvbS93cy8yMDA4LzA2L2lkZW5
0aXR5L2F1dGhlbnRjqY2F0aW9ubWV0aG9kL3g1MDkiXSwiYXBwdHlwZSI6IkNvbmZpZGVudGlhbCIsImFwcGlkIjoiQ0MtMTAzMj
U3LUY2Nzc4NDQtMzgzNzEtVUFUIiwiYXV0aF90aW1lIjoiMjAxOS0xMS0yMVQyMTo1OTozNC4wMTlaIiwidmVyIjoiMS4wIn0.O
1LeuTzHpCZ9eja4sBgpl9Ypf71s747alq0uOQ4XJ-
198AKRzvPzeltJotl063qfxShHrfpP1Z9jBh2bt7iu8yh3EeUZc_dpl7cjAlllZSAmKuDyP8d2evHxp0P4gfIAixTeaNBIwwbiAu
-CczN30g2RgQvWyTqhRpw2uLXvTFR4QnVSztgcAgKqq-
PmMYXACNziWqzUMGsHEK8Qw5eQP0v6pXdx4LLiJBxNIc0R1OFN-
GT8X8bsQVasHF0ASBG9UuawbpV6h3qoySgZgfDjOpJcpAHsdz4xDKNkOsMfLGah51wgNumt4wz-
S6XYkWPgobe4TOqk9FdWnimgruaOg",
 "token_type": "bearer",
 "expires_in": 28800
}
```


3.4: Use access token

Use value returned in Step 3.3 as bearer token to send HTTP requests to Merchant Services API endpoints.

Note: Once token has expired, an HTTP status of **401** will be returned. Repeat Step 3.3 to obtain a new access token.