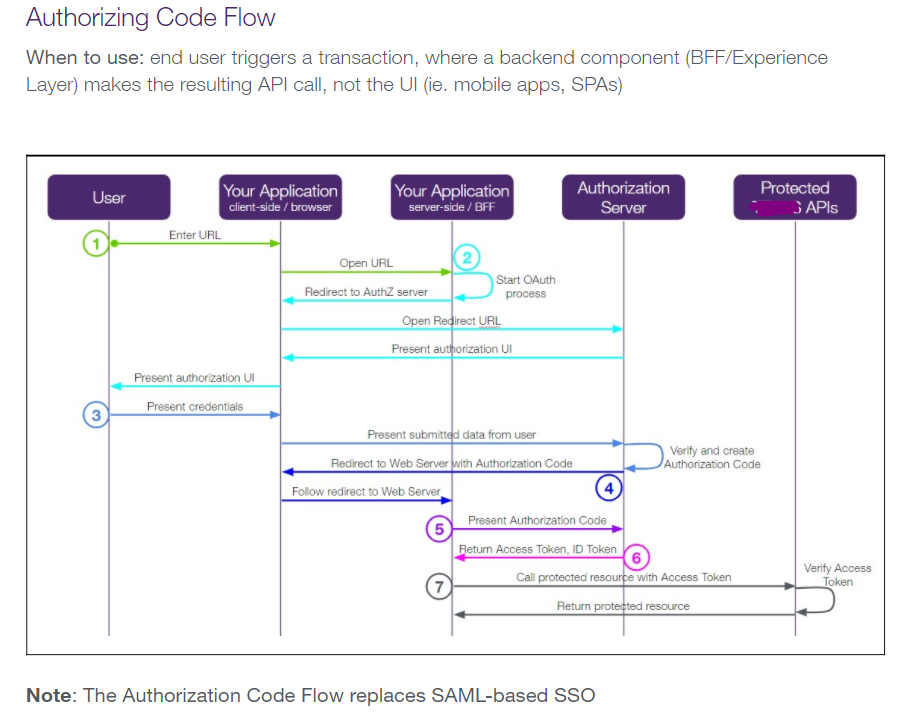
### Authorizing Code Flow

When to use: end user triggers a transaction, where a backend component (BFF/Experience Layer) makes the resulting API call, not the UI (ie. mobile apps, SPAs)



**Note**: The Authorization Code Flow replaces SAML-based SSO

Overview:

* Phase 1: The end user will attempt to access your application
* [Phase 2](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.qq77wamd1rwu): Your application will redirect the user to the Authorization Server for login
* Phase 3: The end user will login to the Authorization Server using their own credentials
* [Phase 4](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.ts5hgkgeq3pt): The Authorization Server will send your application an Authorization Code
* [Phase 5](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.q2dzzhx0fmwu): Your application will use its Client ID, Client Secret, and the received Authorization Code to make a token exchange request to the Authorization Server
* [Phase 6](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.6g6c65sctnae): The Authorization Server will send your application an Access Token, Refresh Token, and optionally an ID token if **openid** was requested
* Phase 7: Your application will use the received Access Token to access protected backend services

#### **Phase 2 Details**

Your application will use the following URL syntax to redirect the user to the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s):

https://teamsso.com/as/authorization.oauth2?**response\_type**=code&**client\_id**=*yourClientID*&**redirect\_uri**=*yourSuccessURI*&**scope**=*yourTargetAPIID*

where

* The host name depends on the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) endpoint for your deployment environment. Change this as appropriate.
* response\_type=code means an auth code is requested.
* client\_id is the Client ID of your application, assigned during the [registration process](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r). Change this as appropriate for your application.
* redirect\_uri is where, for your application, you want to redirect the user after their login is successful. Change this as appropriate for your application.
* scope  is the API ID (as listed in Stoplight) that your application needs to invoke.  You can only request for scopes that your client application is authorized for; this is setup during the [registration process](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r). The scope value is space-delimited.  If your application needs to use ID Token, you will also request these scopes: “openid email profile”.

#### **Phase 4 Details**

The URL syntax for the Auth Code response that your application will receive from the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) is

*http://yourSuccessURI*/?**code**=receivedAuthCode  
  
Where code is the Auth Code granted to you by the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) for your application to save and use in Phase 5.

#### **Phase 5 Details**

To exchange the Auth Code from [Phase 4](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.ts5hgkgeq3pt) for a token, submit an HTTP POST request to the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) as follows:

POST /as/token.oauth2 HTTP/1.1

Host: teamauthz.com

Content-Type: application/x-www-form-urlencoded

Authorization: Basic [*yourClientIDandClientSecret*](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r)

**code**=receivedAuthCode&**grant\_type**=authorization\_code&**redirect\_uri**=[*yourSuccessURI*](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.qq77wamd1rwu)

where the Authorization header must contain the [Client ID](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r) and [Client Secret](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r) and code is the value obtained in [Phase 4](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.ts5hgkgeq3pt).  Note that the grant\_type and redirect\_uri parameters are also required in the POST body.

#### **Phase 6 Details**

The response from the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) will look like this (if **openid** was requested):

{

    "**access\_token**": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ.eyJzY29wZSI6WyJvcGVuaWQiLCJlbWFpbCIsInByb2ZpbGUiXSwiY2xpZW50X2lkIjoia29uZzEiLCJhdWQiOiJrb25nIiwibGFzdE5hbWUiOiJMZXVuZyIsInN1YiI6IlQwNzE2MDMiLCJlbWFpbCI6ImVkbXVuZC5sZXVuZ0B0ZWx1cy5jb20iLCJnaXZlbk5hbWUiOiJFZG11bmQiLCJleHAiOjE1ODEwMTgwMzh9.huhTK0izFQ7TbyZL-JxlWZ9SoNoSXeOF80VIV6xSb-weBHvMVYYFLvzee3FNe0XCtUkz1zsgfRCsq0dgRI5pdR7ynCMYi33LUlYVFDeryGKGLn8okl\_1-UoBdzG-6TBLagZDZ-mowCELbcV3yxT4OgjtA-0ArhA9IZmHRLjsvAgUZu4FAmPSjJLwSfjGKPoWccfF-OKakqlO5TC2ZWRP8mkjo5VQ4X5Td5knDQ4yLX-4GqZLMB1waIYJjEg7zpNh3VuramSMhQFFGW95Lwouw1Okz8nsBHcrrPPDX75CeIfigqt1kVbHrhbQGUo\_fzN\_n9\_7opNQBaxLHZWKqTsc1g",

    "refresh\_token": "S8FvzqwZXy0l0WAKYLNMZRgmxz3SGaXJK98QPIPXNe",

    "id\_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6Im9RUG44M3ZrQlo0YTlUblFpS1I3SmkifQ.eyJzdWIiOiJUMDcxNjAzIiwiZ2l2ZW5fbmFtZSI6IkVkbXVuZCIsImZhbWlseV9uYW1lIjoiTGV1bmciLCJlbWFpbCI6ImVkbXVuZC5sZXVuZ0B0ZWx1cy5jb20iLCJhdWQiOiJrb25nMSIsImp0aSI6InlzOHBsVkdNSU1FRnFldUNvdG5sMlUiLCJpc3MiOiJodHRwczpcL1wvdGVhbXNzby1kdi50ZWx1cy5jb20iLCJpYXQiOjE1ODEwMTc3MDMsImV4cCI6MTU4MTAxODAwMywiYWNyIjoidXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOmFjOmNsYXNzZXM6dW5zcGVjaWZpZWQiLCJhdXRoX3RpbWUiOjE1ODEwMTc3MDJ9.SLqqEinngMdM3wHY2UJ1pBxxm2vL0wX\_jGfqwBcS\_9knMUnd-2clX3dfflWwJk2QobuksM87r43Zem764V0CyGxBxhQa-xsMNn0974KM5bkWEYJ-iITlyxY13vJUIdXNZJSQl0zOCVRKXWk3OvcZeHADPrMsoHhvRkwwNEmlb8wg58THPPDB4WTs2nMcgOy1klzbcbjJC2dMr2fYmY-pVCWFJCLBUSuhv6qSRvV2nf5Yxg8h2KYsEpoDH6GGkC06iS1g1V3YAP3zi9Grh1lezCN6Oo8gr5JeD-37YNqpjmytXaD3ts\_r8YVBw\_bk0D\_PKJcOxBImdkJELEPskUURLQ",

    "token\_type": "Bearer",

    "expires\_in": 299

}

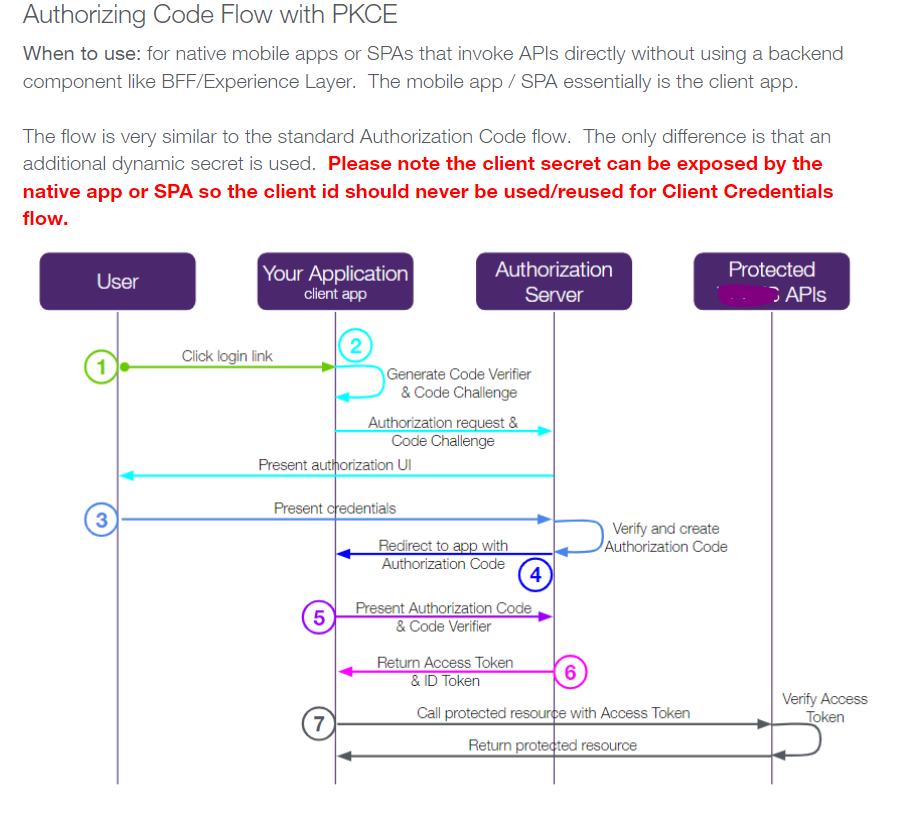
#### **Phase 7 Details**

Your application will use the Access Token received from the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) in [Phase 6](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.6g6c65sctnae) to call the API, following [these instructions](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.y1j4m1u2fzab).

### Authorizing Code Flow with PKCE

When to use: for native mobile apps or SPAs that invoke APIs directly without using a backend component like BFF/Experience Layer.  The mobile app / SPA essentially is the client app.

The flow is very similar to the standard Authorization Code flow.  The only difference is that an additional dynamic secret is used.  **Please note the client secret can be exposed by the native app or SPA so the client id should never be used/reused for Client Credentials flow.**



Overview:

* Phase 1: the end user will open your native application and attempt to login
* [Phase 2](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.bw6snfhb1ds2): your application will redirect the user to the Authorization Server for login with a Code Challenge
* Phase 3: the end user will login to the Authorization Server using their own credentials
* [Phase 4](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.w1tyoacckjf5): the Authorization Server will send your application an Authorization Code
* [Phase 5](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.ta0re33q456): your application will use its Client ID, Client Secret, Code Verifier, and the received Authorization Code to make a token exchange request to the Authorization Server
* Phase 6: the Authorization Server will send your application an Access Token, Refresh Token, and optionally an ID Token if **openid** was requested
* Phase 7: your application will use the received Access Token to access protected backend services

#### **Phase 2 Details**

Your application will generate a random secret called a **Code Verifier**. You will then hash this Code Verifier using the SHA256 algorithm to create what is called the **Code Challenge**.

Your application will then use the following URL syntax to redirect the user to the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s):

https://teamsso.com/as/authorization.oauth2?**response\_type**=code&**client\_id**=yourClientID&**redirect\_uri**=*yourSuccessURI*&**scope**=*yourTargetAPIID*&**code\_challenge**=yourGeneratedCodeChallenge&**code\_challenge\_method**=S256

Where:

* The host name depends on the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) endpoint for your deployment environment. Change this as appropriate.
* response\_type=code means an auth code is requested.
* client\_id is the Client ID of your application, assigned during the [registration process](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r). Change this as appropriate for your application.
* redirect\_uri is where, for your application, you want to redirect the user after their login is successful. Change this as appropriate for your application.
* scope is the API ID (as listed in Stoplight) that your application needs to invoke.  You can only request for scopes that your client application is authorized for; this is setup during the [registration process](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r). The scope value is space-delimited.  If your application needs to use ID Token, you will also request these scopes: “openid email profile”.
* code\_challenge is an additional parameter unique to PKCE, where the value is a generated secret that has been hashed using the SHA256 algorithm
* code\_challenge\_method=S265 is an additional parameter unique to PKCE, which indicates which hashing algorithm you used to create the code challenge.

#### **Phase 4 Details**

The URL syntax for the Auth Code response that your application will receive from the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) is

*http://yourRedirectURI*/?**code**=receivedAuthCode  
  
Where code is the Auth Code granted to you by the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) for your application to save and use in [Phase 5](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.ta0re33q456).

#### **Phase 5 Details**

To exchange the Auth Code from [Phase 4](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.w1tyoacckjf5) for a token, submit an HTTP POST request to the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) as follows:

POST /as/token.oauth2 HTTP/1.1

Host: teamauthz.com

Content-Type: application/x-www-form-urlencoded

Authorization: Basic [*yourClientIDandClientSecret*](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r)

**code**=receivedAuthCode&**grant\_type**=authorization\_code&**redirect\_uri**=[*yourRedirectURI*](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.qq77wamd1rwu)&**code\_verifier**=yourGeneratedCodeVerifier&**client\_id**=yourClientID

where code is the value obtained in [Phase 4](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.w1tyoacckjf5), and code\_verifier is the unhashed secret your application generated back in [Phase 2](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.bw6snfhb1ds2).  Note that the grant\_type and redirect\_uri parameters are also required in the POST body.

#### **Phase 6 Details**

The response from the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) will look like this (if **openid** was requested):

{

    "**access\_token**": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ**.**eyJzY29wZSI6WyJvcGVuaWQiLCJlbWFpbCIsInByb2ZpbGUiXSwiY2xpZW50X2lkIjoia29uZzEiLCJhdWQiOiJrb25nIiwibGFzdE5hbWUiOiJMZXVuZyIsInN1YiI6IlQwNzE2MDMiLCJlbWFpbCI6ImVkbXVuZC5sZXVuZ0B0ZWx1cy5jb20iLCJnaXZlbk5hbWUiOiJFZG11bmQiLCJleHAiOjE1ODEwMTgwMzh9.huhTK0izFQ7TbyZL-JxlWZ9SoNoSXeOF80VIV6xSb-weBHvMVYYFLvzee3FNe0XCtUkz1zsgfRCsq0dgRI5pdR7ynCMYi33LUlYVFDeryGKGLn8okl\_1-UoBdzG-6TBLagZDZ-mowCELbcV3yxT4OgjtA-0ArhA9IZmHRLjsvAgUZu4FAmPSjJLwSfjGKPoWccfF-OKakqlO5TC2ZWRP8mkjo5VQ4X5Td5knDQ4yLX-4GqZLMB1waIYJjEg7zpNh3VuramSMhQFFGW95Lwouw1Okz8nsBHcrrPPDX75CeIfigqt1kVbHrhbQGUo\_fzN\_n9\_7opNQBaxLHZWKqTsc1g",

    "**refresh\_token**": "S8FvzqwZXy0l0WAKYLNMZRgmxz3SGaXJK98QPIPXNe",

    "id\_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6Im9RUG44M3ZrQlo0YTlUblFpS1I3SmkifQ.eyJzdWIiOiJUMDcxNjAzIiwiZ2l2ZW5fbmFtZSI6IkVkbXVuZCIsImZhbWlseV9uYW1lIjoiTGV1bmciLCJlbWFpbCI6ImVkbXVuZC5sZXVuZ0B0ZWx1cy5jb20iLCJhdWQiOiJrb25nMSIsImp0aSI6InlzOHBsVkdNSU1FRnFldUNvdG5sMlUiLCJpc3MiOiJodHRwczpcL1wvdGVhbXNzby1kdi50ZWx1cy5jb20iLCJpYXQiOjE1ODEwMTc3MDMsImV4cCI6MTU4MTAxODAwMywiYWNyIjoidXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOmFjOmNsYXNzZXM6dW5zcGVjaWZpZWQiLCJhdXRoX3RpbWUiOjE1ODEwMTc3MDJ9.SLqqEinngMdM3wHY2UJ1pBxxm2vL0wX\_jGfqwBcS\_9knMUnd-2clX3dfflWwJk2QobuksM87r43Zem764V0CyGxBxhQa-xsMNn0974KM5bkWEYJ-iITlyxY13vJUIdXNZJSQl0zOCVRKXWk3OvcZeHADPrMsoHhvRkwwNEmlb8wg58THPPDB4WTs2nMcgOy1klzbcbjJC2dMr2fYmY-pVCWFJCLBUSuhv6qSRvV2nf5Yxg8h2KYsEpoDH6GGkC06iS1g1V3YAP3zi9Grh1lezCN6Oo8gr5JeD-37YNqpjmytXaD3ts\_r8YVBw\_bk0D\_PKJcOxBImdkJELEPskUURLQ",

    "token\_type": "Bearer",

    "expires\_in": 299

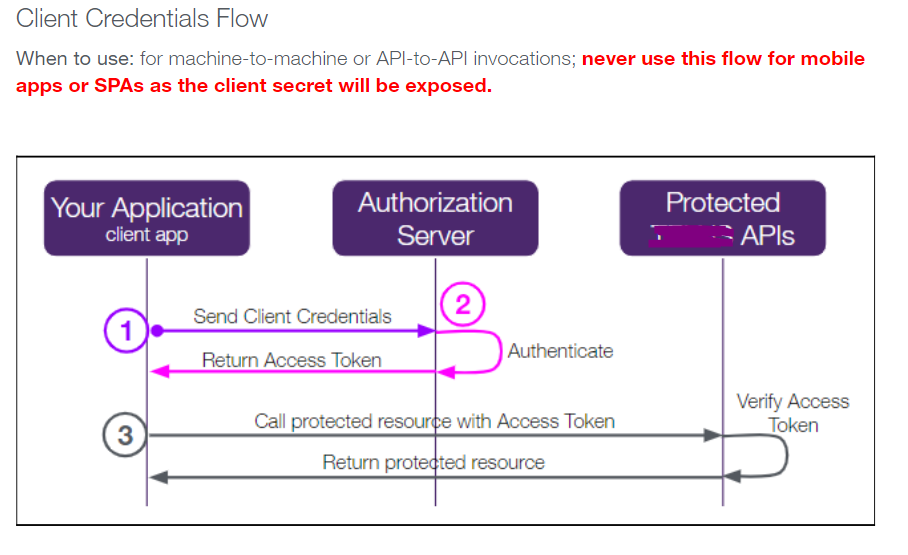
}

#### **Phase 7 Details**

Your application will use the Access Token received from the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) in [Phase 6](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.r1kcz5t3q2hl) to call the API, following [these instructions](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.y1j4m1u2fzab).

### Client Credentials Flow

When to use: for machine-to-machine or API-to-API invocations; **never use this flow for mobile apps or SPAs as the client secret will be exposed.**



Overview:

* Phase 1: your application will authenticate against the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) using its Client ID and Client Secret.
* Phase 2: the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) will send your application an Access Token
* Phase 3: your application will use the received Access Token to access protected backend services

#### **Phase 1 Details**

Your application will request an Access Token by submitting an HTTP POST request to the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) as follows:

POST /st/token HTTP/1.1

Host: apigw.com

Content-Type: application/x-www-form-urlencoded

Authorization: Basic [*yourClientIDandClientSecret*](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r)*(base64 encoded)*

**grant\_type**=client\_credentials&**scope**=*yourTargetAPIID*

where the Authorization header must contain the [Client ID](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r) and [Client Secret](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.yocwcpqypa7r), the grant\_type is client\_credentials, and the scope is the API ID (as listed in Stoplight) that your application needs to invoke.  You can only request for scopes that your client application is authorized for; this is setup during the [registration process](http://go/dev). The scope value is space-delimited.

#### **Phase 2 Details**

The response from the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) will contain the Access Token only:

{

    "**access\_token**": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ.eyJzY29wZSI6WyIxMjEiXSwiY2xpZW50X2lkIjoiMTEwZWRkMmUtY2Y1MC00ZGNmLWE5MzUtYWUyZTk2Yjk3NGNlIiwiYXVkIjoia29uZyIsImV4cCI6MTU4MTAxODc1N30.rCcRv1-uiQDHUaWW8JlusrLSMw1yvBfU13WmSyMkuosr6tQBzLCYp2Pyy7HP2Ga5Oyai96qVjwLYHuWqQwscnTOBgWaEuHmdcQ\_PaGh12VeR-wzHGsQHDU6alOYCzUvOgAjgHjG1g8eprdLu9nAdjncgOicq84GlcXAr1KNo3AyB\_GypYbZhTW0sZbDGS9-VSATmkkJryhE0Sa1UZ2QISzQzOwpkZnt-ejSmg9XVsZKekHghIwIpzVj8-C-svFVanLtiztU0DPo7HkFI\_xQD3UMZZ7f1rhD5OB8zfdcWA4FSEuQUoa\_QlCd7no5G2v\_1VB5Ie48IEK8cjVyEvWarAQ",

    "token\_type": "Bearer",

    "expires\_in": 299

}

#### **Phase 3 Details**

Your application will use the Access Token received from the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) in [Phase 2](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.q86hfmhie28a) to call the API, following [these instructions](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.y1j4m1u2fzab).

### Refreshing the Token

...

Note that this applies to the Authorization Code Flow ([with](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.9ey2ett4zpam) or [without](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.aynqns635uf) PKCE) only. When an access token has expired, the refresh token is used to obtain a new access token and a new refresh token.

To refresh, submit an HTTP POST request as follows:

POST /as/token.oauth2 HTTP/1.1

Host: teamauthz-dv.tsl.telus.com

Content-Type: application/x-www-form-urlencoded

Authorization: Basic \*\*\*\*\*\*\*\*\*\*\*\*

**grant\_type**=refresh\_token&**refresh\_token**=*receivedRefreshToken*

where grant\_type must be refresh\_token and refresh\_token is the refresh token received from the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) in [Phase 6](https://docs.google.com/document/preview?hgd=1&id=1sFk1RrLa6Z4OAegJFFMMdYXktFEcjUDoMONB_Exh6_4#heading=h.6g6c65sctnae).

If the refresh token expires, a login (step #1 in the Authorization Code Flow) should be triggered so a new refresh token can be obtained.  If the SSO session is still active, the end user will not need to provide credentials again.

### Caching the Token

Regardless of how an access token is obtained, the token must be cached until it expires so that the hit to the token endpoint (of the [Authorization Server](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s)) is minimized.  Tokens are issued with a TTL (“expires\_in”) and it should be used to determine if an access token is about to expire and a refresh needs to be performed.  **Caching is mandatory because we do not want to create unnecessary traffic to the** [**Authorization Server**](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s)**.  The** [**Authorization Server**](https://docs.google.com/document/d/1QhEkjy6eSmN-azyBVerZqw955EogfWw3feSKSqK7nBQ/edit#heading=h.ptz5ayxorj5s) **is a shared infrastructure; any degradation will impact all applications.**

On the Google Cloud, use Google’s [Memorystore](https://cloud.google.com/appengine/docs/standard/nodejs/using-memorystore) to cache the Access Token.

## **Handling an Expired Token**

Your application (i.e. the client app) is responsible for handling access/refresh token expiry. Your design should minimize the impact to the end user experience.  See the Token Refresh section above to learn how to refresh access token. 2 common approaches are:

1. Check the expiry (calculated via the expires\_in value) and refresh as needed before an API call.
2. Make the API call.  If the call is unsuccessful due to expired access token, catch the error, refresh the token and make the call again without propagation the error upwards.