**OAuth2.0 Implementation in IBM API Connect**

# Version History

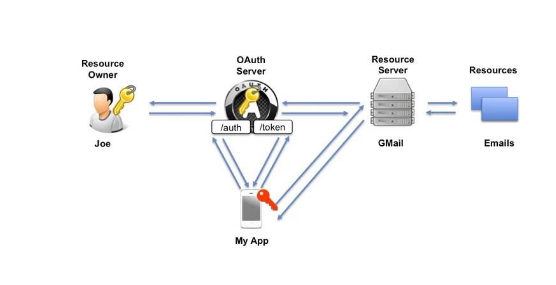
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date Created | Author | Version | Approved by: | Comments |
| 07/22/2020 | Memar Demeke | 1.0 |  | Initial version |
| 07/24/2020 | Memar Demeke | 2.0 |  | Second version (updated) |
|  |  |  |  |  |
|  |  |  |  |  |

**OAuth2.0 Implementation in IBM API Connect**

1. **Why OAuth2.0?**

* OAuth is a token based authentication and authorization mechanism that provide API security. There are two OAuth types used in APIC: User OAuth token and Application OAuth token. The token types are determined by an API architect.

1. **OAuth Actors**



1. **Advantages of OAuth2.0**

* This flexible protocol relies on SSL (Secure Sockets Layer) to ensure data between the web server and browsers remain private
* It uses tokenization to give limited access to the user's data
* It is easy to implement and provides strong authentication. In addition to the two-factor authentication, tokens can be revoked if necessary (i.e, suspicious activity).
* It allows applications to read data of a user from another application
* Uses single sign on
* It gives users more control over their data. Users can selectively grant access/ revoke to various functionalities for applications they want to use

1. **OAuth2.0 Grant Types and Usage Scenario**

OAuth2.0 provides several grant types for different use cases. These grant types are:

* **Authorization code**
* **Password**
* **Application/Client credentials**
* **Implicit**

|  |  |  |
| --- | --- | --- |
| **No** | **OAuth Grant Types** | **Description and Usage Scenario** |
| 1 | Authorization Code/Three legged | * Application has the user provide authorization through a form provided by the gateway server, which, if they grant authorization, provides an authorization code to the application. * The application sends the authorization code to the provider API to get an access token * Secure storage for tokens, client id and client secret needs to be provided by the client. * Client id and secret are used when creating new tokens, to confirm the identity of the client * **Used for apps running on a web server, browser-based and mobile apps** |
| 2 | Application/client credentials/Two legged | * The user is not required to provide authorization at any stage * The client needs to provide secure storage for client id and secret; and access token. * **Used for application access in a scenario without a resource owner** |
| 3 | Password | * The user provides the application with a user name and password. Then, the client will directly contact the provider API to request an access token. * This has an advantage over the application using the password directly, because the validity of the access token or client ID can later be revoked without impacting other applications that do not need their access revoked. * Application must be trusted to not store the user name and password. * **Used for logging in a username and password and if the resource owner trusts the client and willing to share credentials to the client.** |
| 4 | Implicit | * Application requests an access token from the gateway server and the user grants permission, at which point an access token is provided to the user, who must then pass the token to the application * It was previous recommended for clients without a secret, but has been suspended by using Authorization Code grant with no secret. * **Used for clients that do not securely store the refresh token, client id and secret** |

1. **OAuth Token (s)**

* There are two types of access token
  1. **User Token (Three Legged OAuth):** authentication and authorization by user role. User controls as what data shall be accessed by a given user based on his grant types. User credentials are authenticated against an identity stores/management like Active Directory (LDAP).
  2. **Application Token (Two Legged OAuth):** Authentication and authorization is straightforward. Client application is registered in API Connect with a client id and secret. The client application will invoke an **OAuth access token endpoint, passing client ID and secret** to retrieve the token to get access to services on the resource server. We can use this token flavor for our domain.

**Application Token:**

* In application grant types, application may need an access token to act on behalf of themselves rather that a user. For example, the service may provide a way for application to update their own information such as their website URL or icon, or they may wish to get statistics about users of the app.
* In this case, application need a way to get an access token for their own account, outside of the context of any specific user. OAuth provide the client credentials grant type for this purpose.

**Token flow:**

1. Consuming application register itself in API Manager/developer portal to get client id and secret
2. Consuming application needs to subscribe the OAuth provider API
3. Consuming applications invokes API Manager OAuth token provider endpoint, passing the OAuth parameters **(grant\_type, scope, client id and secret)** to get an access token.
4. API Manager validates the OAuth parameters
5. API Manager returns an OAuth token to the consuming application
6. Consuming application calls the API endpoint with the access token
7. If the token is valid and not expired, the resource server/API will return a response

**Note:** There is no user interaction in the generation of an access token

**Sample OAuth Client Credentials grant type Implementation in IBM API Connect**

**OAuth Provider API Name:** Oauth2.0ProviderDemo-ApplicationFlow

**Access Token URL:**

<https://dev-apicgw.humana.com:48022/humana/dev/oauth20providerapi-applicationflow/oauth2/token>

**Grant type:** Application/client\_Credential

**Scope**: jsontoxml

**Consuming APP:** TestApp\_HCaaS

**OAuth Protected API Name:** OAuth2.0ProtectedAPI 1.0.0

**API URL:** <https://dev-apicgw.humana.com:48022/humana/dev/oauth20protectedapi/transformation>

**HTTP headers:**

X-IBM-Client-Id:0f81d1ac-24cb-4b78-a0fa-2c948077582d

X-IBM-Client-Secret:W5gN8mM0pY2fL8bW2jH7xW0rK1eK8hC2cX6wE4hX7pM1tK4pT4

Content-Type:application/json

**Sample request message:**

{

"id":123,

"firstName":"Thomas",

"lastName":"Bear"

}

**Sample API call response:**

<?xml version="1.0" encoding="UTF-8"?>

<Employee>

    <id>123</id>

    <firstName>Thomas</firstName>

    <lastName>Bear</lastName>

</Employee>

**Sample OAuth Application token call**

POST /humana/dev/oauth20providerapi-applicationflow/oauth2/token HTTP/1.1

Host: dev-apicgw.humana.com:48022

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

User-Agent: PostmanRuntime/7.19.0

Accept: \*/\*

Cache-Control: no-cache

Postman-Token: a2452b28-7888-4f7a-b54b-9bb5d725ddb7,182b1ed1-5d96-4b54-852b-50a31e91897b

Host: dev-apicgw.humana.com:48022

Accept-Encoding: gzip, deflate

Content-Length: 157

Connection: keep-alive

cache-control: no-cache

grant\_type=client\_credentials&client\_id=0f81d1ac-24cb-4b78-a0fa-2c948077582d&client\_secret=W5gN8mM0pY2fL8bW2jH7xW0rK1eK8hC2cX6wE4hX7pM1tK4pT4&scope=jsontoxml

**Sample access token Response:**

{

    "token\_type": "Bearer",

    "access\_token": "AAIkMGY4MWQxYWMtMjRjYi00Yjc4LWEwZmEtMmM5NDgwNzc1ODJkYh5ICJcElSxA\_g5pt2Ch98\_SlQwjlPoSa-7hp7-7787L5F\_F0eo2TMQ0cIApYpuEoKMk2Xx\_IEZLaRrSlMUyPb\_U2lWIhlv4ZVRdsNuxMvl3OaeOqRS1p5nZwU4rLwVo",

    "expires\_in": 3600,

    "consented\_on": 1595600925,

    "scope": "jsontoxml"

}

**Sample API call with access token:**

POST /humana/dev/oauth20protectedapi/transformation HTTP/1.1

Host: dev-apicgw.humana.com:48022

X-IBM-Client-Id: 0f81d1ac-24cb-4b78-a0fa-2c948077582d

X-IBM-Client-Secret: W5gN8mM0pY2fL8bW2jH7xW0rK1eK8hC2cX6wE4hX7pM1tK4pT4

Content-Type: application/json

Authorization: Bearer AAIkMGY4MWQxYWMtMjRjYi00Yjc4LWEwZmEtMmM5NDgwNzc1ODJkZucB6DBYqVdMnonNaFf5-hOOZr3n5RXG1P5MuXx5jg4o3hN70jjWg7maEbU5xE4j0Xlk5Uo8APF7Tbj-kM3gA4y3LzMV1YTsYKrPHv8TQlywndO66PRMNQwlq0-kmb5l

User-Agent: PostmanRuntime/7.19.0

Accept: \*/\*

Cache-Control: no-cache

Postman-Token: ae99ca2b-64cc-45e3-8d5d-96fccccd8294,c9eb4627-56a1-4b38-8a7d-911e59ab1b2b

Host: dev-apicgw.humana.com:48022

Accept-Encoding: gzip, deflate

Content-Length: 57

Connection: keep-alive

cache-control: no-cache

{

"id":123,

"firstName":"Thomas",

"lastName":"Bear"

}

1. **What changes on the consumer side (existing/new) to invoke API with OAuth?**

* Existing or New consumer applications need to subscribe OAuth Token provider API from API Manager or developer portal in order to get client id and client secret before invoking the access token endpoint.
* The API needs to be secured with OAuth security. This will be accomplished by configuring the OAuth details based on the grant types under the designer tab in security definition section.

1. **What changes on the deployment side of API with OAuth? Can we use one OAuth Provider API for multiple APIs?**

* Yes, securing an API with OAuth doesn’t impact promotion of APIs from lower to higher environments. However, subscription has to be created for the application trying to call the API secured with OAuth in the target environment
* We can use one OAuth provider API for multiple APIs per environment. The assumption here is that we are using the same the grant type and scope