Ordo - Platform Guide for API clients

Version 1.0 September 2020

Proprietary and confidential

Document history

Version	Date	Material amendments
1.0	September 2020	N/A

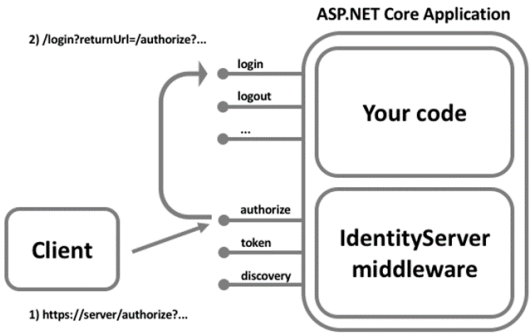
Contents

1 Introduction

1.1 Purpose

This document is intended for clients to give them the complete information about how to consume the Ordo platform in their applications. Clients have to obtain a security token from the Ordo platform identity server and submit the token to the Ordo API Gateway to request Ordo's APIs.

API Security – Authentication and Authorisation



The Ordo platform implements an Identity Service to provide Authentication and Authorisation services and each client application will be assigned a set of credentials which must be presented to the Identity Service whenever it is used.

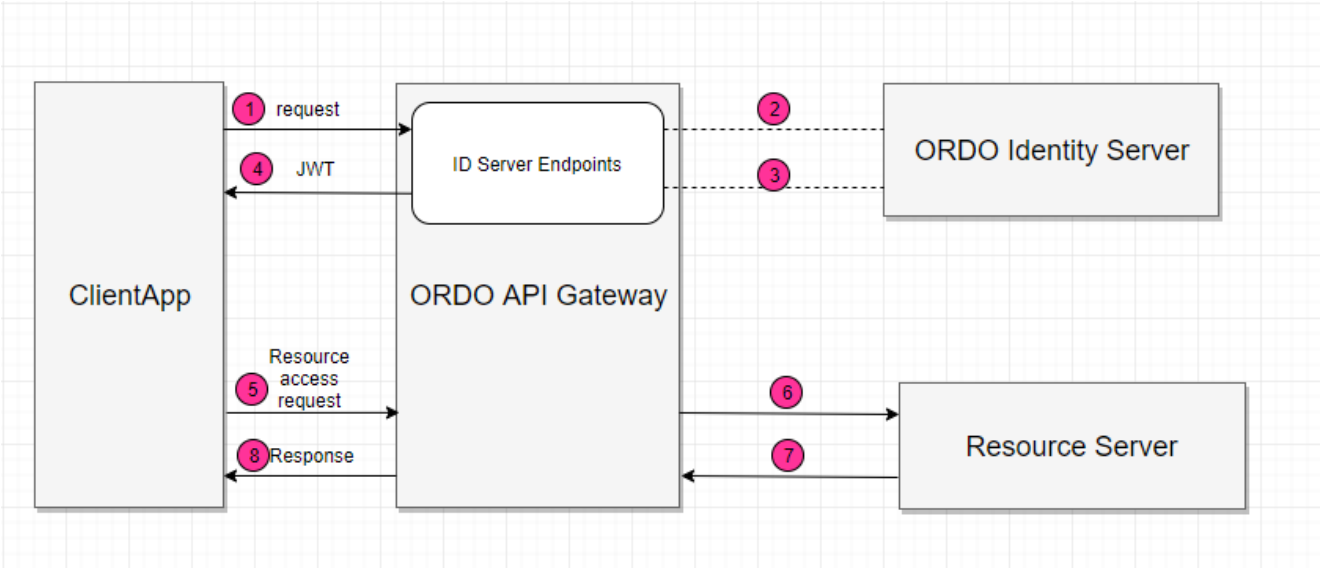
Configuration within the identity service database defines items such as API permissions for the client, allowed redirection URLs, Token lifespan, Cross-Origin Resource policy etc. The Identity Service will issue Access Tokens for use by client applications.

Each API will require an access token which has been issued by the Ordo Identity service.

The access token will be checked by each API to ensure

1. It has been issued by the Ordo identity service.
2. It has been digitally signed using the correct signing certificate.
3. The access token provides the correct permissions to access the API.

If any of these criteria are not met, then the access will not be granted to the API and a HTTP response code will be issued to indicate the action in not authorised



1.2 Audience

The audience for this document is any firm looking to integrate via API with Ordo, described through this document as 'the client'

1.3 Client Details

The client will share with us the information listed here:

Key	Value	Description	Required
ClientUri	Example; "www.msn.com"		Mandatory
ClientRedirectUri	Example: https://www.getpostman.com/oauth2/callback	The URIs that we will accept as destinations when returning authentication responses (tokens) after successfully authenticating users.	Mandatory
clientPostlogoutredirectUrl	Example: https://www.getpostman.com/oauth2/callback	Where the client wants to redirect user after click on logout link in login screen	Optional
Protocol Type	Example: "oidc" or "ouath2"	How the client will interact with the Ordo platform	Mandatory

The Ordo Admin will issue a Client Id and Client Secret for use with the Ordo Identity Service.

1.4 Ordo Platform Details

Details for the Ordo Sandbox Environment:

Parameter	Value
Auth/ID Server URL	https://test.identity.ordopay.com
Gateway URL	https://test.api.ordopay.com
ID Server Discovery Url	https://test.identity.ordopay.com/.well-known/openid-configuration
Authorize Endpoint Url	Client can obtain from “Discovery Document” link on identity server. It looks like this: https://test.identity.ordopay.com/connect/authorize
Token Endpoint Url	Client can be obtained from “Discovery Document” link on identity server. It looks like this; https://test.identity.ordopay.com/connect/token

NOTE: Clients must **NOT** attempt to connect to the Live APIs as part of any testing or create real bank accounts in the Sandbox environment.

Details for the Ordo Production Environment:

Parameter	Value
Auth/ID Server URL	https://live.identity.ordopay.com
Gateway URL	https://live.api.ordopay.com
ID Server Dicovery URL	https://live.identity.ordopay.com/.well-known/openid-configuration
Authorize Endpoint Url	https://live.identity.ordopay.com/connect/authorize
Token Endpoint Url	https://live.identity.ordopay.com/connect/token

NOTE: Clients must **NOT** attempt to connect to the Live APIs as part of any testing or create real bank accounts in the Sandbox environment.

1.5 Grant Types

The Ordo platform supports Authorization Code, Implicit, Resource Owner Password and Client Credentials grant types. Clients have to register/inform the Ordo Platform Admin in advance of the required grant type they want to use in their applications.

2 Client Configuration

2.1 Pipeline configuration for ASP.NET Core applications

Clients will configure the pipeline for Direct Injection to implement authentication and authorization in their application like below. This piece of code must be added to the "**ConfigureServices**" method. Below is sample code for a client application to consume the Ordo Identity Service.

```
JwtSecurityTokenHandler.DefaultInboundClaimTypeMap.Clear();

services.AddAuthentication(options =>
{
  options.DefaultAuthenticateScheme =
  CookieAuthenticationDefaults.AuthenticationScheme;
   options.DefaultScheme = "Cookies";
   options.DefaultChallengeScheme = "oidc";
})
.AddCookie("Cookies")
.AddOpenIdConnect("oidc", options =>
{
  options.SignInScheme = "Cookies";
  options.Authority = [Identity_Server_Url];
  options.RequireHttpsMetadata = false;
  options.ClientId = "mywebclient";
  options.ClientSecret = "thesecret";
  options.ResponseType = "code"; //This force for authorization code flow
  options.SaveTokens = true;
  options.GetClaimsFromUserInfoEndpoint = true;
  options.Scope.Add("openid");
  options.Scope.Add("profile");
  options.Scope.Add("offline_access");
   options.Scope.Add("users"); //Client can add/remove other scope as per
                               //need


 });
```

Finally call **UseAuthentication()** in **Configure()** method to implement authentication.

2.2 Required Scope

The below table gives the scopes required to get the access tokens to access exposed methods under respective endpoints. The API endpoints URI will change according to environment but the required scope will remain same.

API Endpoint	Scope
https://test.api.ordopay.com/usermanager/user	users
https://test.api.ordopay.com /smartrequestmanager/smartRequests	smartrequests
https://test.api.ordopay.com /registrymanager	registry
https://test.api.ordopay.com /documentmanager/documents	Documents

In addition the following scopes are also provided:

- openid, which tells the provider to return the sub (subject id) claim in the identity token.

- offline_access, which tells the provider to provide a refresh token. This is supplied by default if it has been configured by Ordo for the client and is not required.

2.3 Tokens Sample

The Ordo Identity Service will by default return a JSON Web Token (JWT) token. A JWT is represented as a sequence of base64url encoded values that are separated by a 'dot' character. The ideal format is **"Header.Payload.Signature"**, where Header keeps the metadata for the token, the Payload is the claims of the entity (typically user) and the Signature being the signed token. The Signed token is generated by combining the encoded JWT header and Payload and it is signed by using an encryption algorithm like HMAC SHA–256. The signature private key always held by server so it will be able to verify existing tokens as well as sign new tokens.

Access Token:

This token is generally a short-lived token. To check the validity of token we check the "*expires_in*" attribute within the token.

Header:


```
{
  "alg": "RS256",
  "kid": "c093c071bac6f98ab8702becc1011c38",
  "typ": "JWT"
}
```


PayLoad:


```
{
  "nbf": 1560529738,
  "exp": 1560530038,
  "iss": "null",
  "aud": "null/resources",
  "client_id": "postman",
  "sub": "ab9f074b-42be-4949-9958-c3722667210d",
  "auth_time": 1560529724,
  "idp": "local",
  "name": "DemoUser",
  "email": [
    "demouser@cgi.com",
    "demouser@cgi.com"
  ],
  "email_verified": [
```

```
    "true",
    false
  ],
  "scope": [
    "openid",
    "offline_access"
  ],
  "amr": [
    "pwd"
  ]
}
```


Signature:

$encodedContent = base64UrlEncode(header) + “.”+base64UrlEncode(payload)

$signature = hashhmacSHA256($encodedContent)

This gives us the final part of JWT.

ID Token:

Header:


```
{
  "alg": "RS256",
  "kid": "c093c071bac6f98ab8702becc1011c38",
  "typ": "JWT"
}
```


PayLoad:


```
{
  "nbf": 1560529743,
  "exp": 1560530043,
  "iss": "null",
  "aud": "postman",
  "iat": 1560529742,
  "at_hash": "Zzn-dkbMyN8mpWVPQnW9QQ",
  "sid": "c610f1e6591ed39d99534f75c2c6eff6",
  "sub": "ab9f074b-42be-4949-9958-c3722667210d",
  "auth_time": 1560529724,
  "idp": "local",
  "name": "DemoUser",
  "email": "demouser@cgi.com",
  "email_verified": [
    "true",
```

```
    “false”
  ],
  "amr": [
    "pwd"
  ]
}
```


Signature:

$encodedContent = base64UrlEncode(header) + “.”+base64UrlEncode(payload)

$signature = hashhmacSHA256($encodedContent)

This gives us final part of the JWT.

Refresh Token:

This token is a long-lived token compared to the access token and is used to request a new access token in case it is expired. You will receive this in an encoded format. An example is shown below

494c427ace9e04dea03c7234cea96c5ca53e0ce4ea95147e961fd9ebcf8feb84

2.4 Endpoints accessibility

All public/partner APIs endpoints are protected with an access token. Clients can get the access token from the Ordo Identity Service token issuing authority (for example: Test environment Identity Service discovery endpoint url http://test.identity.ordopay.com/.well-known/openid-configuration) and must submit the access token in each request header under the “**Authorization**” attribute.

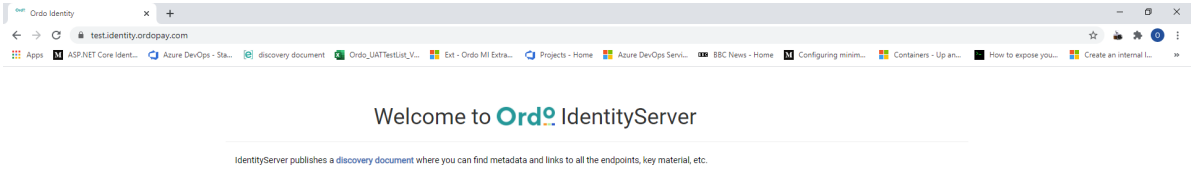
The Ordo Identity Service will issue a token for each client’s app using the authorization code or client credential flow based on the configuration done for clients in Ordo system. The token received using authorization code flow will be issued against the authenticated users and will have user specific claims also.

An API which requires client context and requires the “**UserPID**” to complete the request must receive an access token generated from the Ordo Identity Service using the authorization code flow.

An API which does not require client context to complete the request must receive an access token generated from the Ordo Identity Service using the client credential flow. The token issued using the client credential flow will be issued on-behalf of clients, in this case the token will not have any user specific information. This token will be used in case to access those API endpoints when the user has not been validated, but if the endpoints need “**UserPID**” to complete the request then it must be passed in the request body.

Client applications will have to get the access token with correct scope which is applicable to relevant product/endpoints, then only the respective endpoints will be accessible to them. Clients can get tokens that will be valid for multiple scopes.

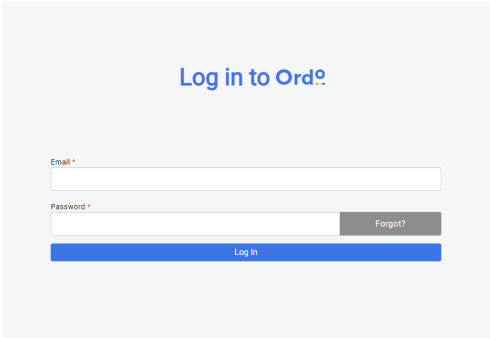
Each issued token has its own self expiry time based on the system configuration.



3 Authorisation Code Grant Flow

Clients will have to register for this flow with the Ordo platform if they want to use the Ordo Identity Service to generate access tokens for APIs.

The Authorization code grant flow is a two-leg process where the client will request an authorization code from the Ordo Identity Service via the Authorization Endpoint. This will redirect the client to the Login Endpoint and a logon screen will be presented for authentication. After successful authentication of the user an authorization code will be returned to the client via a redirect uri.



In the next leg the client will request an access token from the Ordo Identity Service via the Token Endpoint with submission of the client Id, client secret and authorization code received in first leg.

Once the authorization code is validated, three access tokens will be returned

1. Access Token
2. ID Token
3. Refresh Token

Step 1:

This is the request to fetch an authorization ‘code’ for a successfully authenticated user from the Ordo Identity service and grant the consent based on the requested scope from the client app. It is always recommended to send the ‘request’ parameter along with other parameters that help to verify integrity of data. The Ordo platform will verify the signature in the ‘request’ parameter and decode the base64 encoded body from JWS to match the corresponding fields value received in the request.

Parameter	Description	Required
Client_id	Client unique identifier	Mandatory
redirect_uri	must match with one of the allowed redirect URIs for that client	Mandatory
Scope	Must provide one or more registered scope	Mandatory

response_type	supported response_types are: • code • token • id_token multiple response_types are supported using space separation e.g. “code id_token token”.	Mandatory
State	This is sent to stop CSRF attack, ID server will echo back the same value	Recommended
Nonce	ID server will echo back the nonce value in ID token. **Note :** This is required for ID token via implicit grant	Optional
Prompt	This provide two value “login” and “none” . When value set to “none” then no UI will be shown during request.	Optional
code_challenge	Sends the code challenge for PKCE (if used)	Optional
code_challenge_method	If the value is set to “plain” it indicates the challenge is using plain text, if it is set to “S256” then it means challenge is hashed with SHA256. Required for PKCE (if used)	Optional
login_hint	This is used to pre-populate user login name field in login screen.	Optional
ui_locales	Gives a hint about the desired display language of the login UI	Optional
max_age	If the user login session exceed max age (in seconds), the login UI will be shown	Optional
acr_values	Allow passing additional authentication related informations. You can use the following proprietry acr values Idp:name_of_idp Tenant:name_of_tenant	Optional
Request	Instead to provide all as individual query string parameters, you can provide as subset or all of them as JWT.	Recommended

GET Request:

[IdentityServer_Authorize_Endpoint]**?**response_type=code&state=abcfd&scope=read write&redirect_uri=https://ANDDigital.com/callback&request=
yRQYnWzskCZUPxwawaQupWkiUzKELZ49eM7oWxAQK_ZXw
. yRQYnWzdKjHfZUxPwaQupWkiUzKELZ49eM7oWxAQK_JSw
. yRQYnWzskCZUxPwaQupWkiUzKELZ49eM7oWxAQK_ZAw

Non-Base64 Representation of ‘request’ object

{
"alg": "HS256",
"kid": "GxlIiwianVqsDuushgjE0OTUxOTk",
“typ” : “JWT”
} .
{
"iss": "https://ANDDigital.com”,
"aud": "https://test.identity.ordopay.com",

"response_type": "code”,
"redirect_uri": "https://ANDDigital.com/callback",
"scope": "read write",
"state": "abcfd"
} .<<signature>>

The Ordo Identity server will validate the authorization request and generates an Auth Code for the user and attach it to redirection url along with state, for example:

https://ANDDigital.com/callback#
code=BplxlOBeKQQYbYS6WxSbIA&expires_in=300&state= abcfd &response=gkJ0 ...
MiJ9.ezJ1c ... l8IjlifS0.DeRt4Qu ... ZXso

The User will be redirected back to the client app via the redirect uri provided in the request. The client app will use this code to request an access token. The ‘response’ parameter is signed by the Ordo Identity Service using the private key of the signing certificate. Client apps will verify this signature with a shared public key and match values with those present in response.

Non-Base64 representation of the ‘response’ object

{
"alg": "HS256",
"kid": " GxlIiwianVqsDuushgjE0OTUxOTk "
}
.
{
"code ": "BplxlOBeKQQYbYS6WxSbIA",
"state": abcfd,
"iss": “https://repository1.rtp.fasterpaymentsdemo.co.uk ",
"aud": "https://ANDDigital.com/callback",
"exp": 1311281970
}
.
<<signature>>

Step 2:

Client apps will initiate the request for an access token as soon as they receive the authorization code and the state is valid. Client apps will also send ‘client_assertion’ along with authorization code which signed by the private key of the app. The Ordo platform will verify the signature in the ‘client_assertion’ parameter and decode the base64 encoded body from JWS to match the ‘code’ value.

Parameter	Description	Required
client_id	Client unique identifier	Mandatory
client_secret	This can be pass either in post body or as basic authentication header	

redirect_uri	must match with one of the allowed redirect URIs for that client	Mandatory
grant_type	supported grant types are: • password • authorization_code • client_credentials • refresh_token • device_code	
Scope	Provide one or more registered scopes, otherwise a token will be issued for all explicitly allowed scope.	Mandatory
Code	This value is required for authorization_code grant type.	Mandatory
code_verifier	PKCE proof key. Required for PKCE (if used)	Optional
refresh_token	The refresh token is required for refresh_token grant type	
acr_values	Allow passing additional authentication related informations. You can use the following proprietary acr values Idp:name_of_idp Tenant:name_of_tenant	Optional
Request	Instead to provide all as individual query string parameters, you can provide as subset or all of them as JWT.	Recommended

POST Request:

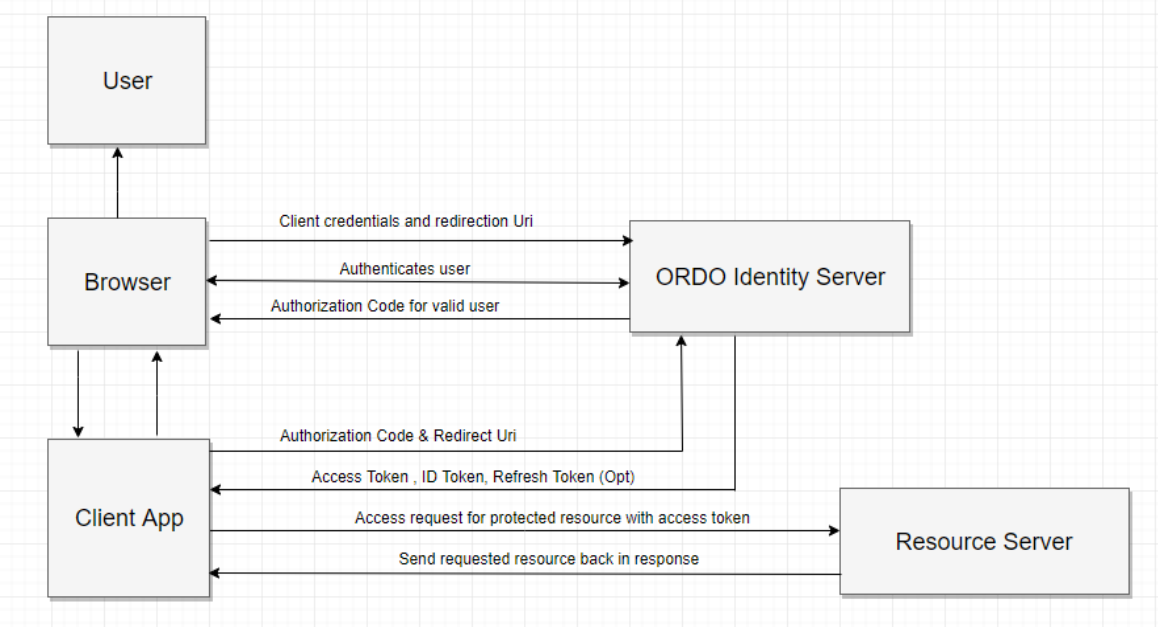
HOST URL : [ORDO_IdentityServer_Token_Endpoint]

Content-Type: application/x-www-form-urlencoded
Accept: application/json
grant_type=authorization_code
&code= BplxlOBeKQQYbYS6WxSbIA
&client_id= BplxlOBeKQQYbYS6WxSbIA
&client_= BplxlOBeKQQYbYS6WxSbIA
&code= BplxlOBeKQQYbYS6WxSbIA
&client-assertion-type=jwt-bearer
&client_assertion=cyJhbKciOiJSUzI1MiIsInR5cCI6IkpYVCJ9.eyJpc3NiOiJodHRw…_QrOyM
0pONWKj9K4Mj7I4GPGvzyVqpaXUgjcOaZY_rlu_p9tnBlD781dDNuw

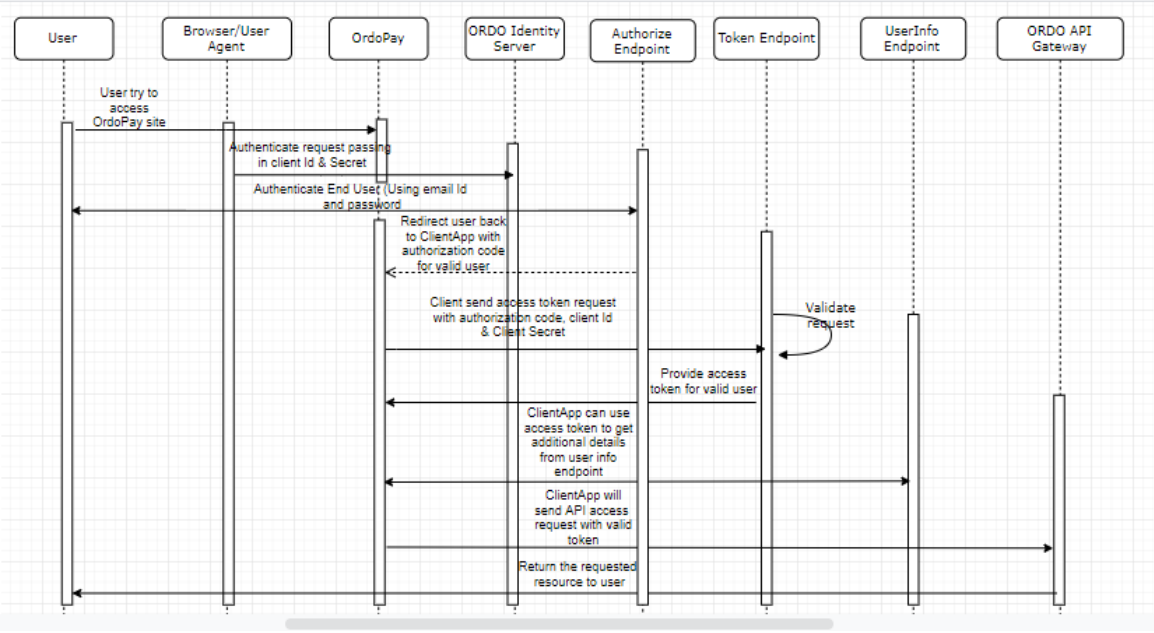
Client apps will receive response like after successful validation like below

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
"access_token": "WQAV32hkKG",
"id_token": "BKPD32hkMG",
"token_type": "Bearer",
"expires_in": 3600
"refresh_token": "p2jKX45hnzG"
}

3.1 Block Diagram



3.2 Sequence Diagram



Authorization Code Flow Details:

Step 1: ORDOPay app user tries to access the protected resource in app.

Step 2: ORDOPay will send this request to authorization server to authenticate with client credentials (client ID & Secret) and requested scope to the Authorize endpoint of the Ordo Identity Service.

Step 3: The Ordo Identity Service will launch the login screen and ask user to submit his/her credential (email & password) to authenticate for request.

Step 4: The Ordo Identity Server will Return an authorize code to client app via the redirect uri, if user is a valid for the request.

Step 5: ORDOPay app will send the access token request with client credentials and authorize code to the Token endpoint of the Ordo Identity Service.

Step 6: The Ordo Identity Service will send the access token to client via the redirect uri

Step 7: The ORDOPay app will use the access token and request access to protected API (resource) via the ORDO API gateway.

Step 8: The Ordo API Gateway will validate the access token against the Ordp Identity Service. If the token is valid it will pass the request to the requested resource (API) and the response will be returned back to the ORDOPay app through gateway.

Example calls:

1. **Obtain an authorisation code from the authorisation end point**

https://test.identity.ordopay.com/connect/authorize?client_id=***your_client_id***&scope=openid%20offline_access&redirect_uri=***your_redirect_uri***&response_type=code

GET /connect/authorize HTTP/1.1

Host: test.identity.ordopay.com

client_id=***your_clientid***&request_uri= ***your_redirect_url***&scope=openid offline_access&response_type=code

2. **Obtain an access token from the token end point**

Grant_type = authorization_code to obtain an access token

POST /connect/token HTTP/1.1

Host: test.identity.ordopay.com

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

client_id=***your_client_id***&client_secret=***your_client_secret***&redirect_uri=***your_redirect_uri***&grant_type=authorization_code&state=14518944&code=***authorisation code received from authorisation end point***

3. **Obtain an refresh token from the token end point**

Grant_type = refresh_token to obtain an refresh token

Refresh token request

POST /connect/token HTTP/1.1

Host: test.identity.ordopay.com

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

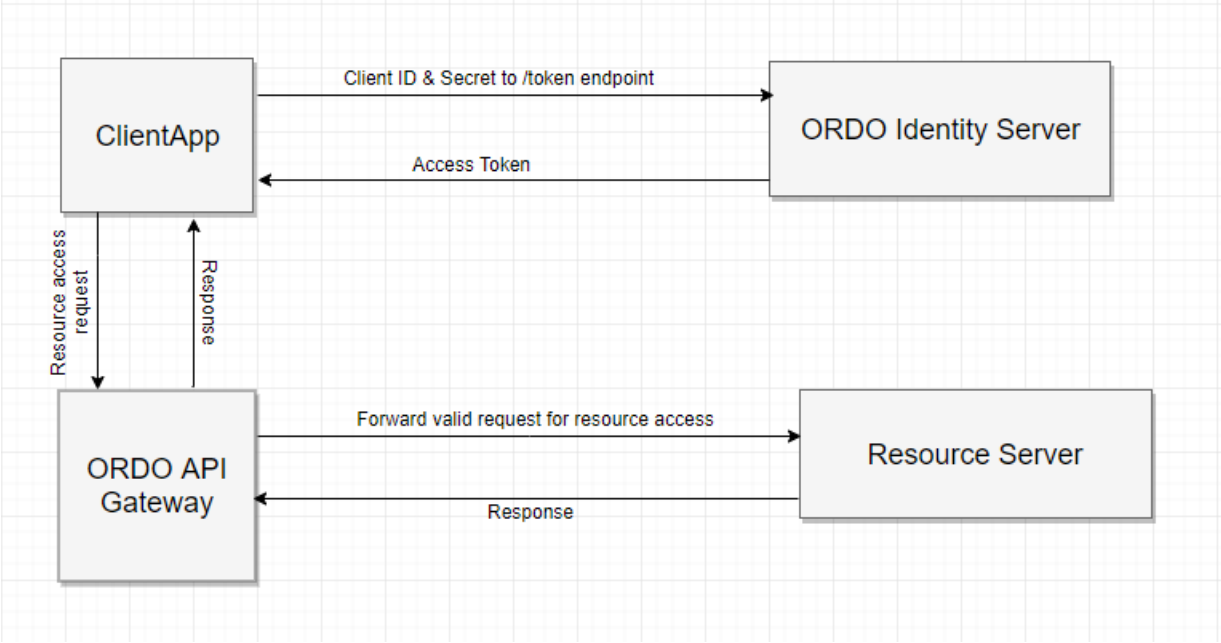
client_id=***your_client_id***&client_secret=***your_client_secret***&redirect_uri=***your_redirect_uri***&grant_type=refresh_token&refresh_token=***refresh token previously provided from token endpoint (step 2)***

4 Client Credential Grant Flow

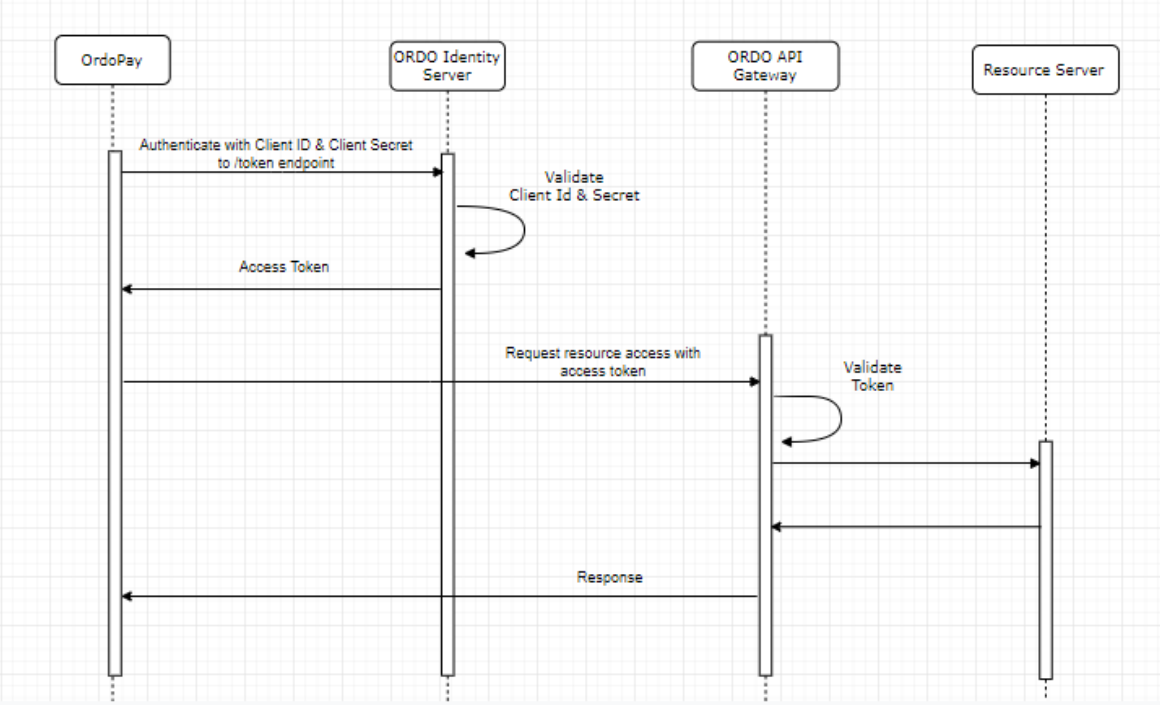
This grant type is used by clients to obtain an access token outside the context of a user. In this flow the Ordo Identity Service will authorize an application rather than a user. This is ideally used by clients if they want to access resources about themselves rather than to access user specific resource. This is particularly required for M2M (Machine-to-Machine) applications.

Parameter	Description	Required
client_Id	A unique id provided by ORDO admin to the client	Mandatory
client_secret	A unique secret provided by ORDO admin to the client	Mandatory
scope	This is required to provide API access to access client app based on the read and write.	Mandatory
grant_type	Pass the value “client_credentials” for this field	Mandatory

4.1 Block Diagram



4.2 Sequence Diagram



Client Credential Flow Details:

Step 1: ORDOPay app user tries to access the protected resource in app.

Step 2: ORDOPay app will send the access token request with client credentials and client secret to the Token endpoint of the Ordo Identity Service.

Step 3: The Ordo Identity Service will validate send the access token to client via the redirect uri

Step 4: The ORDOPay app will use the access token and request access to protected API (resource) via the ORDO API gateway.

Step 5: The Ordo API Gateway will validate the access token against the Ordo Identity Service. If the token is valid it will pass the request to the requested resource (API) and the response will be returned back to the ORDOPay app through gateway.

Example calls:

1. **Obtain an access token from the token end point**

Grant_type = client_credentials to obtain an access token

POST /connect/token HTTP/1.1

Host: test.identity.ordopay.com

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

client_id=***your_client_id***&client_secret=***your_client_secreti***&grant_type=client_credentials&scope=***required scope e.g. user.***