9. Authentication and Authorization Types in Postman - 2

OAuth (Open Authorization)

- **OAuth (Open Authorization)** is a secure way to allow one app to access specific information from another app without sharing your credentials — giving limited, controlled access to only what you approve.

OAuth 1.0 Real-Time Example: Twitter Login for a News App

Imagine you have a news app, and you want to allow users to log in using their Twitter account. Here's how the OAuth 1.0 process would work

1. **Request Token**
   a. Your news app asks Twitter for a temporary request token.
   b. Twitter responds with a request token and a token secret.
2. **User Authorization**
   a. Your app redirects the user to Twitter's authorization page, where they log in and approve access.
   b. After approving, Twitter sends the user back to your app with a special code (oauth_verifier).
3. **Access Token Request**
   a. Your app then exchanges the request token and oauth_verifier for a permanent access token and access token secret.
   b. Twitter validates this and provides the access token.
4. **Access Protected Resources**
   a. Now, your app can access the user's Twitter data (like their tweets, followers, etc.) using the access token.

This flow ensures that the user never has to share their Twitter password with your app, providing secure access to their data.

OAuth 2.0 Real-Time Example: News App Login with Google

Imagine you have a news app, and you want users to log in using their Google account. Here's how the OAuth 2.0 process would work

1. **Authorization Request**
   a. Your news app sends the user to Google's login page.
   b. The URL includes your client_id, redirect_uri, and scope (like access to the user's email and profile).
   c. For example
      i. https://accounts.google.com/o/oauth2/auth?client_id=YOUR_CLIENT_ID&redirect_uri=YOUR_REDIRECT_URI&scope=email+profile&response_type=code.

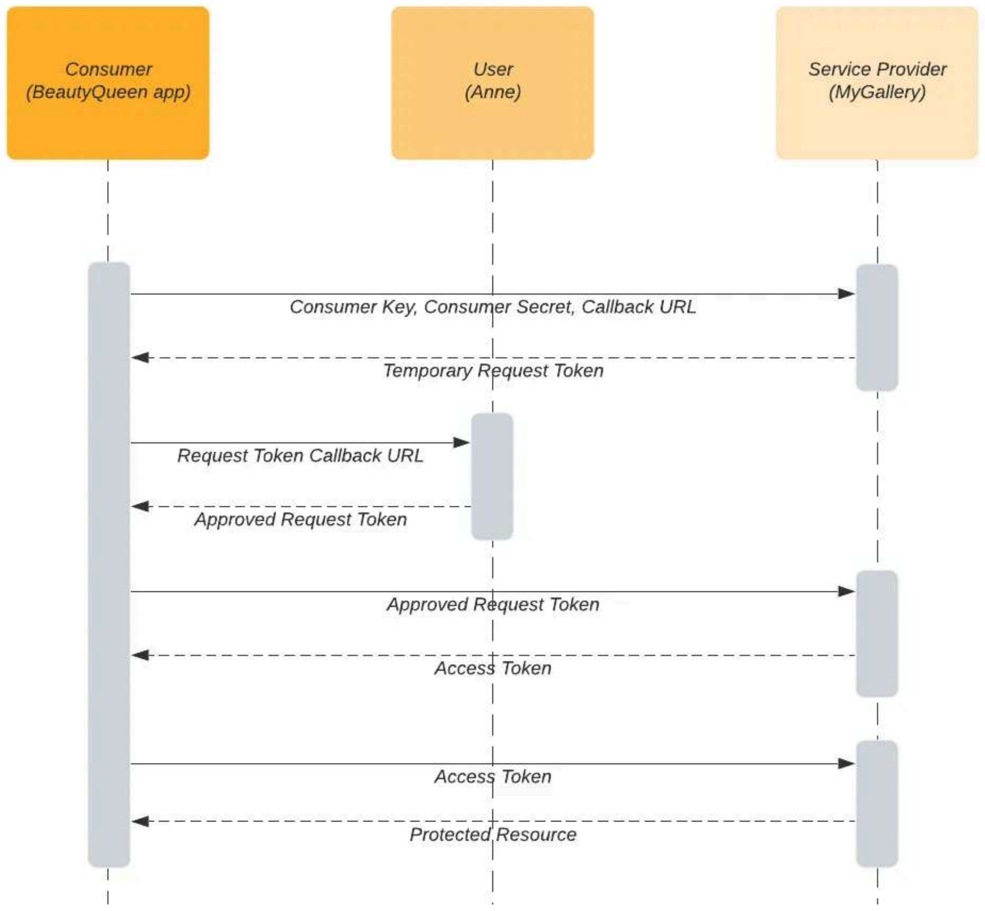
2. **User Consent**
    a. The user logs in and grants your app permission to access their Google account data.
    b. After approval, Google redirects the user back to your app with an authorization code.
3. **Exchange Authorization Code**
    a. Your app sends the authorization code to Google's token endpoint, along with your client_secret.
    b. Google verifies this and responds with an access token and a refresh token.
4. **Access Protected Resources**
    a. Your app uses the access token to access the user's Google data, like their email, profile picture, or contacts.
5. **Token Refresh (Optional)**
    a. If the access token expires, your app can use the refresh token to get a new one without requiring the user to log in again.

This flow ensures that the user can securely access their Google account data without sharing their password with your app, providing a seamless and secure login experience.

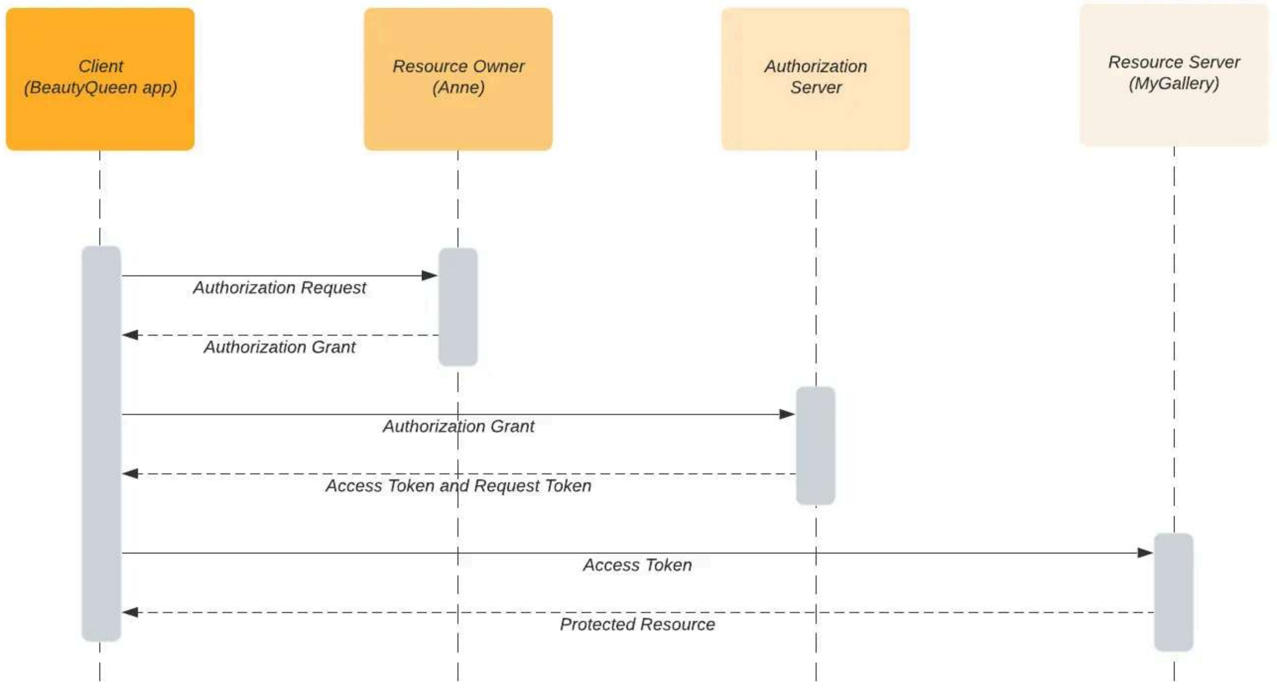
OAuth another Real-Time Example

➔ https://medium.com/identity-beyond-borders/oauth-1-0-vs-oauth-2-0-e36f8924a835

OAuth (Open Authorization) 1.0


Consumer (BeautyQueen app)
User (Anne)
Service Provider (MyGallery)
Consumer Key, Consumer Secret, Callback URL
Temporary Request Token
Request Token Callback URL
Approved Request Token
Approved Request Token
Access Token
Access Token
Protected Resource

OAuth (Open Authorization) 2.0



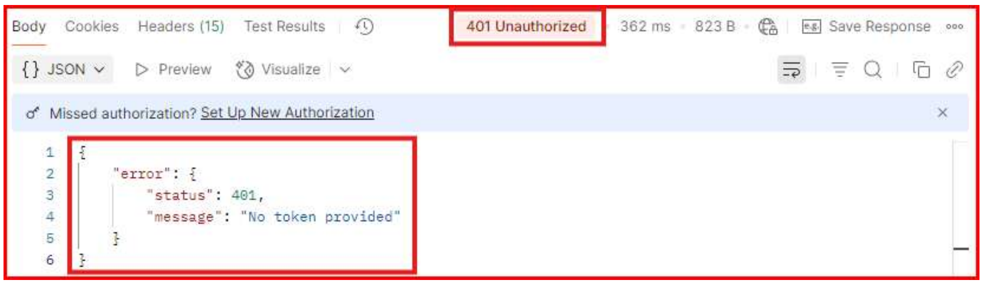
➔ **Use Case:** Used in platforms like Google, Facebook, or Imgur for user authentication.

Aspect	OAuth 1.0 (Twitter Login)	OAuth 2.0 (Google Login)
Token Type	Request token, Access token (with secrets)	Authorization code, Access token, Refresh token
Signing Method	Requires signing each request with a secret	Uses Bearer tokens, no signing required
Complexity	More complex, with separate request token and access token steps	Simpler, uses a single authorization code flow
Security	More secure due to signature-based verification	More flexible but requires HTTPS for security
User Flow	Redirect to authorization URL, exchange request token, get access token	Redirect to authorization URL, exchange authorization code, get access token
Token Secret	Uses both token and token secret for verification	No token secret, relies on HTTPS for secure transport
Refresh Token	Not supported	Supported, allows long-term access without repeated logins
Scopes	Limited support for scopes	Strong support for fine-grained scopes (e.g., profile, email)

➔ **Example** → Fetching artist from your account in spotify

- **Request: GET** https://api.spotify.com/v1/artists/0TnOYISbd1XYRBk9myaseg
- **Auth Type:** OAuth 2.0

➔ Create a new Get Request with name **OAuth2_spotifyAPI** in Postman.



Missed authorization? Set Up New Authorization

```
{
    "error": {
        "status": 401,
        "message": "No token provided"
    }
}
```

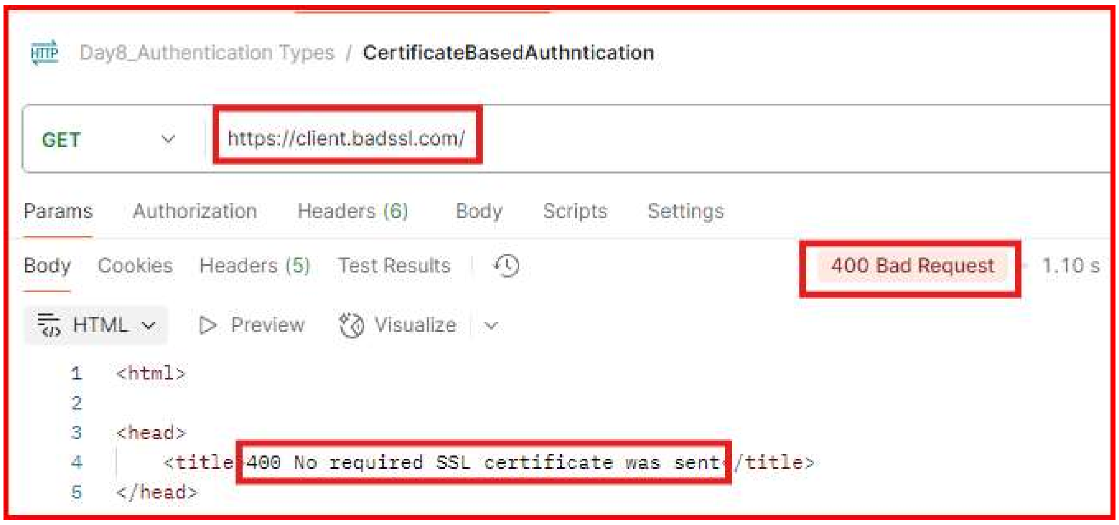
➔ Authentication error will occur and 401 Unauthorized status will be returned. This means that this endpoint requires Authentication to access the resource.

➔ Tokens are generated through the OAuth process.

➔ Refer **Generate Access Token using OAuth 2 in Postman.pdf**

Certificate-based authentication in Postman

- Certificate-based authentication is used when high security is required to ensure that only trusted machines or users can access an API — not just anyone with a username and password.
- Unlike passwords that can be guessed or stolen, or API keys that can be copied, certificates are much harder to fake and serve as proof of identity.
- **CA (Certificate Authority)**
    - A Certificate Authority is a trusted entity that issues digital certificates to verify the identity of a website or client. It ensures secure communication by signing certificates with its private key, creating a chain of trust.
- **Client Certificates**
    - Client certificates are digital certificates used to authenticate a client (user, application, or device) to a server, ensuring that only authorized clients can access the server.
- **Why We Need Them**
    - **Authentication:** Client identity verification ensures only authorized users can connect.
    - **Encryption:** Enables secure, encrypted communication using SSL/TLS.
    - **Integrity:** Prevents data tampering by validating the integrity of messages.
- Files involved in CA and client certificates are CA and Client Certificate **(.crt/.pem)**, Private Key **(.key)**, Certificate Signing Request **(.csr)** and PKCS#12 File **(.pfx/.p12)**.
- **Who Provides CA and client certificates to Developers or Testers?**
    - **Security Teams or DevOps Engineers:** Usually handle the creation and distribution of these files.
    - **Project Admins or IT Teams:** May provide the client certificates for testing environments.
- **Example** → BadSSL provides a test endpoint that requires client certificate authentication
    - **Request: GET** https://client.badssl.com/
    - **Auth Type:** No Auth


Day8_Authentication Types / CertificateBasedAuthntication

GET https://client.badssl.com/

Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (5) Test Results — 400 Bad Request 1.10 s

HTML Preview Visualize

```
<html>

<head>
    <title>400 No required SSL certificate was sent</title>
</head>
```

Step-by-Step Guide to Configure Postman for Client Certificate Authentication

➔ **Obtain the Client Certificate**

- BadSSL offers a client certificate for testing purposes. we can download the certificate bundle from https://badssl.com/download/
- This .p12 file contains both the client certificate and the private key. The password for this file is
    - Password: **badssl.com**

➔ **Configure Postman with the Client Certificate**

- Open Postman and navigate to Settings (⚙ icon).
- Go to the Certificates tab. Click on Add Certificate.
- In the Host field, enter: **client.badssl.com**
- Leave the Port field blank (defaults to 443).
- Under PFX file, click Select File and choose the downloaded **badssl.com-client.p12.**
- Enter the Passphrase: **badssl.com**
- Click Add to save the certificate configuration.

➔ **Send a Request to the Protected Endpoint**

- **Request: GET** https://client.badssl.com/
- Click Send to make the request.

➔ If the client certificate is correctly configured and accepted by the server, you should receive a 200 OK response with a message indicating successful authentication.

➔ **Verifying Certificate Usage**

- To confirm that Postman is sending the client certificate:
    - Open the Postman Console (View > Show Postman Console).
    - Send the request again.
    - In the console, you should see details about the request, including the certificate used.

➔ Refer below link for **Managing CA and client certificates in Postman**

- https://learning.postman.com/docs/sending-requests/authorization/certificates/

Note

➔ In simple terms, a digital certificate is like an ID card — it lets your system securely enter a protected space (API), just like an ID gives access to a secure office.

➔ This method is commonly used in banking, government, and enterprise systems, where security is critical.

Summary

- Each authorization type serves a different purpose
    - **Basic Auth:** Username and password (simple and quick).
    - **Digest Auth:** More secure than Basic Auth.
    - **API Key:** A unique key to authenticate (public APIs).
    - **Bearer Token:** A token-based secure method (user-specific access).
    - **OAuth 2.0:** The most secure and widely used for user authentication.
    - **Hawk Auth:** HMAC-based authentication for message integrity and security.
    - **JWT (JSON Web Token):** Compact, self-contained token for secure, stateless authentication.
    - **Certificate Based Authentication:** Uses digital certificates for client-server authentication (highly secure).