Hands-On Guide: Implementing OAuth 2.0 (Authorization Code Flow)

This guide provides the required endpoints and parameters for a server-side web application.

I. Facebook (Meta) Login Implementation

Facebook uses the standard **Authorization Code Flow**. You will need your `App ID` (Client ID) and `App Secret` (Client Secret).

Step 1: Initiate Authorization (Client-Side Redirect)

Your application redirects the user's browser to the following URL.

Parameter	Value (Example)	Description
`client_id`	`YOUR_FACEBOOK_APP_ID`	Your app's public ID.
`redirect_uri`	`https://yourdomain.com/facebook/callback`	Must be an exact match to the one configured in the Meta Developer settings.
`response_type`	`code`	Requests an Authorization Code.
`scope`	`public_profile,email`	The minimum required permissions.
`state`	`unique_csrf_token_12345`	Random, generated string for CSRF protection.

Full Authorization URL Structure:

HTTP
```
https://www.facebook.com/v20.0/dialog/oauth
  ?client_id=YOUR_FACEBOOK_APP_ID
  &redirect_uri=https://yourdomain.com/facebook/callback
  &response_type=code
  &scope=public_profile,email
  &state=unique_csrf_token_12345
```

Step 2: Exchange Code for Token (Server-to-Server POST)

Upon redirect to your `/facebook/callback` endpoint, your server receives the `code` and `state`. After verifying the `state`, your server makes a POST request to the token endpoint.

Parameter	Value (Example)	Description
`client_id`	`YOUR_FACEBOOK_APP_ID`	Your app's public ID.

Parameter	Value (Example)	Description
`client_secret`	`YOUR_FACEBOOK_APP_SECRET`	**Your secret key (NEVER in client code).**
`redirect_uri`	`https://yourdomain.com/facebook/callback`	Must match the original.
`code`	`AUTHORIZATION_CODE_FROM_URL`	The code received from Step 1.

Full Token Exchange Endpoint (POST Request):

HTTP
```
https://graph.facebook.com/v20.0/oauth/access_token
```

Step 3: Fetch User Data (Server-to-Server GET)

Use the received `access_token` to retrieve the user's profile.

Parameter	Value (Example)	Description
`fields`	`id,name,email`	Specify the data fields you want to fetch.
`access_token`	`ACCESS_TOKEN_FROM_STEP_2`	The token obtained from the previous step.

Full User Data Endpoint (GET Request):

HTTP
```
https://graph.facebook.com/v20.0/me
  ?fields=id,name,email
  &access_token=ACCESS_TOKEN_FROM_STEP_2
```

II. LinkedIn Login Implementation

LinkedIn also uses the standard **Authorization Code Flow**. You will use your `Client ID` and `Client Secret`.

Step 1: Initiate Authorization (Client-Side Redirect)

Parameter	Value (Example)	Description
`client_id`	`YOUR_LINKEDIN_CLIENT_ID`	Your app's public ID.
`redirect_uri`	`https://yourdomain.com/linkedin/callback`	Must be an exact match to the one configured in the LinkedIn Developer settings.
`response_type`	`code`	Requests an Authorization Code.

Parameter	Value (Example)	Description
`scope`	`r_liteprofile r_emailaddress`	Permissions for basic profile and primary email. Separate scopes with a space.
`state`	`unique_csrf_token_12345`	Random string for CSRF protection.

Full Authorization URL Structure:

HTTP
```
https://www.linkedin.com/oauth/v2/authorization
  ?response_type=code
  &client_id=YOUR_LINKEDIN_CLIENT_ID
  &redirect_uri=https://yourdomain.com/linkedin/callback
  &state=unique_csrf_token_12345
  &scope=r_liteprofile%20r_emailaddress
```

Step 2: Exchange Code for Token (Server-to-Server POST)

Upon redirect to your `/linkedin/callback` endpoint, your server verifies the `state` and makes a POST request.

Parameter	Value (Example)	Description
`grant_type`	`authorization_code`	Specifies the type of flow.
`client_id`	`YOUR_LINKEDIN_CLIENT_ID`	Your app's public ID.
`client_secret`	`YOUR_LINKEDIN_CLIENT_SECRET`	**Your secret key (NEVER in client code).**
`redirect_uri`	`https://yourdomain.com/linkedin/callback`	Must match the original.
`code`	`AUTHORIZATION_CODE_FROM_URL`	The code received from Step 1.

Full Token Exchange Endpoint (POST Request - Parameters must be URL-encoded in the body):

HTTP
```
https://www.linkedin.com/oauth/v2/accessToken
```

Step 3: Fetch User Data (Server-to-Server GET)

Use the received `access_token` to fetch the user's basic profile and then their email address (two separate API calls).

A. Fetch Basic Profile:

HTTP
```
https://api.linkedin.com/v2/me
  (Requires: Header: Authorization: Bearer ACCESS_TOKEN_FROM_STEP_2)
```

B. Fetch Email Address (Requires `r_emailaddress` scope):

HTTP
```
https://api.linkedin.com/v2/emailAddress?q=members&projection=(elements*(handle~))
  (Requires: Header: Authorization: Bearer ACCESS_TOKEN_FROM_STEP_2)
```

III. Twitter (X) Login Implementation - PKCE Required

Twitter/X mandates the use of **PKCE** (Proof Key for Code Exchange) for web apps. This means the **Client Secret is NOT used in the final token exchange**.

Pre-Step: Generate PKCE Keys

Your server must generate two values:

1. **`code_verifier`**: A cryptographically random string (43-128 characters). **Keep this secret on your server.**
2. **`code_challenge`**: The URL-safe Base64 SHA256 hash of the `code_verifier`.

Step 1: Initiate Authorization (Client-Side Redirect)

You use the `code_challenge` in the redirect URL.

Parameter	Value (Example)	Description
`client_id`	`YOUR_TWITTER_CLIENT_ID`	Your app's public ID.
`redirect_uri`	`https://yourdomain.com/twitter/callback`	Must match the one configured in the Twitter/X settings.
`response_type`	`code`	Requests an Authorization Code.
`scope`	`users.read tweet.read offline.access`	Permissions, including `offline.access` for a Refresh Token.
`state`	`unique_csrf_token_12345`	Random string for CSRF protection.

Parameter	Value (Example)	Description
`code_challenge`	`CODE_CHALLENGE_FROM_PRE_STEP`	The hashed value of the `code_verifier`.
`code_challenge_method`	`S256`	Specifies the hashing algorithm used.

Full Authorization URL Structure:

HTTP
```
https://twitter.com/i/oauth2/authorize
  ?response_type=code
  &client_id=YOUR_TWITTER_CLIENT_ID
  &redirect_uri=https://yourdomain.com/twitter/callback
  &scope=users.read%20tweet.read%20offline.access
  &state=unique_csrf_token_12345
  &code_challenge=CODE_CHALLENGE_FROM_PRE_STEP
  &code_challenge_method=S256
```

Step 2: Exchange Code for Token (Server-to-Server POST)

Upon redirect to your `/twitter/callback` endpoint, your server verifies the `state` and makes a POST request. **Crucially, you send the `code_verifier` (not the `client_secret`)**.

Parameter	Value (Example)	Description
`grant_type`	`authorization_code`	Specifies the type of flow.
`client_id`	`YOUR_TWITTER_CLIENT_ID`	Your app's public ID.
`redirect_uri`	`https://yourdomain.com/twitter/callback`	Must match the original.
`code`	`AUTHORIZATION_CODE_FROM_URL`	The code received from Step 1.
`code_verifier`	**`CODE_VERIFIER_FROM_PRE_STEP`**	**The original secret random string.**

Full Token Exchange Endpoint (POST Request - Parameters must be URL-encoded in the body):

HTTP
```
https://api.twitter.com/2/oauth2/token
```

Step 3: Fetch User Data (Server-to-Server GET)

Use the received `access_token` to retrieve the user's profile.

Parameter	Value (Example)	Description
`user.fields`	`profile_image_url,created_at`	Optional fields to fetch beyond the default.

Full User Data Endpoint (GET Request):

HTTP
```
https://api.twitter.com/2/users/me
  (Requires: Header: Authorization: Bearer ACCESS_TOKEN_FROM_STEP_2)
```