Advent eModal

eModal identity system

Integration Guide

Advent eModal

VERSION CONTROL SHEET

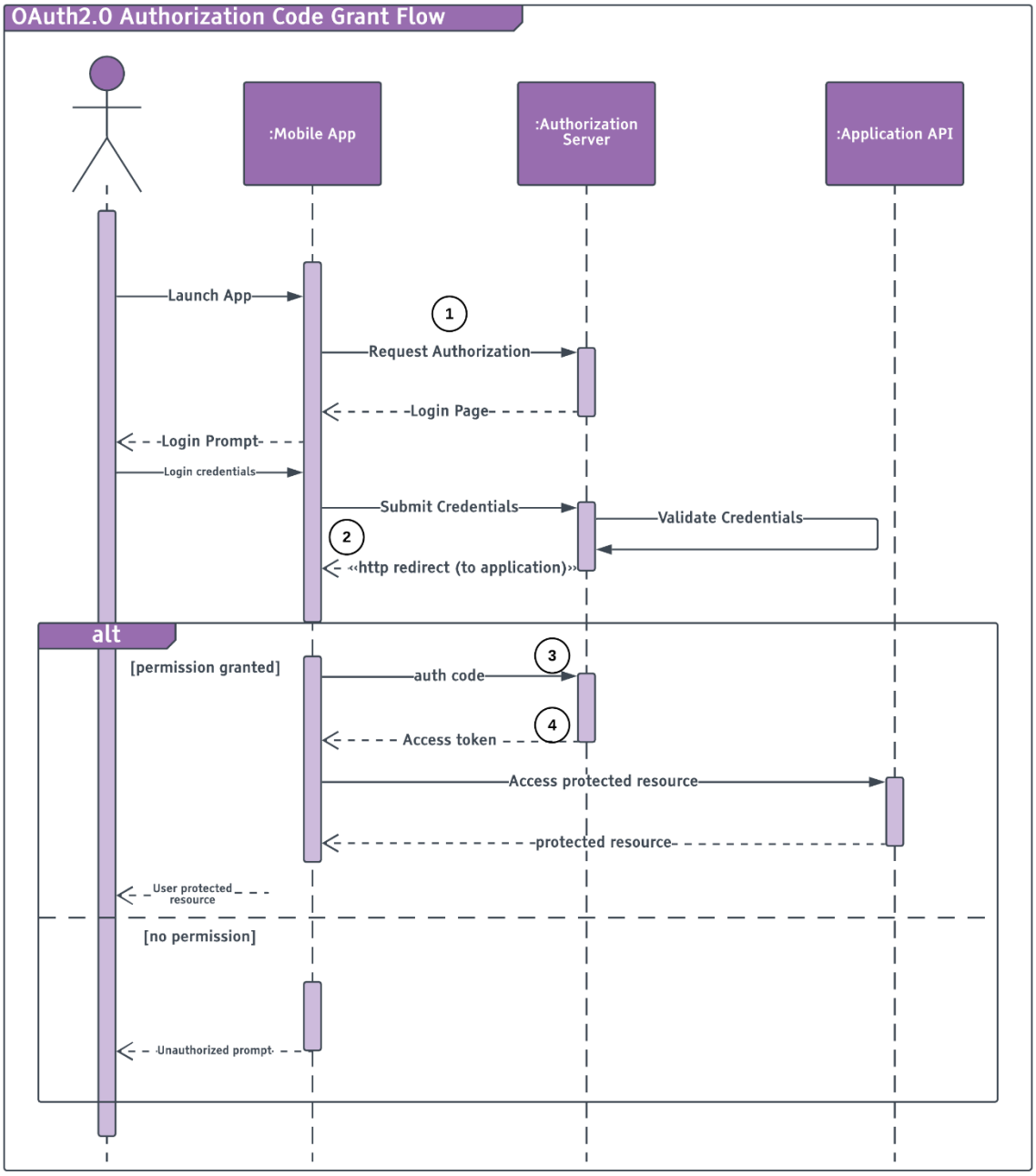
Document Title : eModal identity system integration guide
Description : This document provides a general guidance for integration with eModal identity system.
Issue Date : 16th October 2020
Originator : eModal DevOps

Version	Description of Change	Date	Revision Author
1.0	First draft	10/16/2020	Avinash Kanuganti
1.1	Updated user exp. Flow	10/21/2020	Avinash Kanuganti

Advent eModal

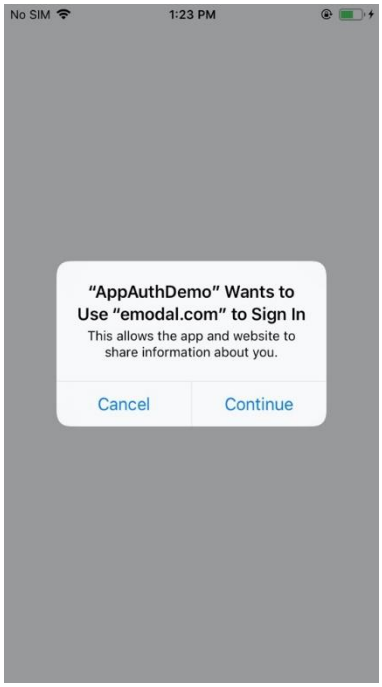
Contents

Process Flow



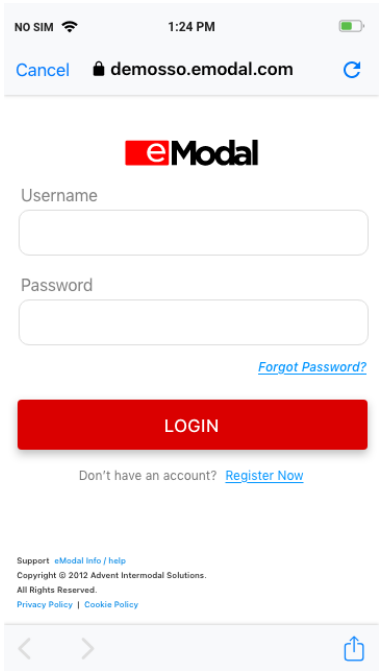
API Definitions and user flow

1 User Launches app and initiates login process. The user is prompted with a message to confirm the use of eModal account to login to the app.




```
GET https://demosso.emodal.com/connect/authorize?
client_id=PROPASS&
redirect_uri=https%3A%2F%2Fmyapp.com%2Fsignin&
response_type=code&
scope=openid%20profile%20sso_auth_api%20offline_access&
nonce=5dabf0734621d8cf607bc7c55302455746S7hFINR&
state=bc3c34ce4d0daafb9a31f9edfa1f193141PkKiSYI&
code_challenge=JA4TFkQf4ji_1JP034qkiHMMmgTh&
code_challenge_method=S256
```

If user approves the prompt, the app launches an in-app web view and the user is redirected to the eModal login page. The user has to enter the eModal credentials and submit.



2 If the login credentials are valid, eModal SSO redirects the user with the authorization code in the response as a query string parameter. The app must capture this response and present the authorization code in the next step.

```
HTTP/1.1 302 Found https://myapp.com/signin-oidc?
code=hQlwGfueKl0FFW-uGqaVyjqP8mJVcadqrvRXNR2_Wac&
scope=openid%20profile%20sso_auth_api%20security_web_auth_api%20sso_custom_endpoint%20offline_access&
state=bc3c34ce4d0daafb9a31f9edfa1f193141PkKiSYI&
session_state=Z5mvghG254byVoNMJzzU4GqWPsmjiaJpl5C3sUdyb6c.gNTvMRl4kDwkmIdzFXcvTA
```

3 The app uses the authorization code from the above response and calls the token end point to get an access token.

```
POST https://demosso.emodal.com/connect/token?
grant_type=authorization_code&
client_id=PROPASS&
code_verifier=aed231f018be0fbec256ea6642db4b0f52a03ce2a2ebe077306ffe30f09015M6Zqv&
code=hQlwGfueKl0FFW-uGqaVyjqP8mJVcadqrvRXNR2_Wac&
redirect_uri=https://myapp.com/signin-oidc
```

4 eModal SSO validates the authorization code and code verifier token sent by the app and if valid, issues a bearer token and a refresh token in the response.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
        "id_token":"eyJhbGciOiJSUzI1NiIsImtpZCI6IkM3M",
        "access_token":"eyJhbGciOiJSUzI1NiIsImtp",
        "expires_in":432000,
        "token_type":"Bearer",
        "refresh_token":"KJP0JmB84Wlbwl",
        "scope":"openid profile sso_auth_api offline_access"
}
```

The app must store the access token and refresh token locally and any secure API calls made by the app must include the access token in the authorization header of the request. The access token has an expiry and can be re-used until it expires. The application must renew the access token before it expires by calling the token endpoint by passing the refresh token in the request. This renewal can be done in the background on the app without the need for any action from the user. If the access token expires and the app does not renew the token, the calls to secure API's will return unauthorized response. The user must reinitiate the login process in this situation.

Integration recommendation

There are several SDKs that provide an abstracted implementation of the OAuth 2.0 flow into your mobile and web applications. One popular example is AppAuth (https://appauth.io/). The opensource project at appauth.io provides examples and interfaces for ios, android and js applications. It is recommended to use these SDK's to simplify the integration.