

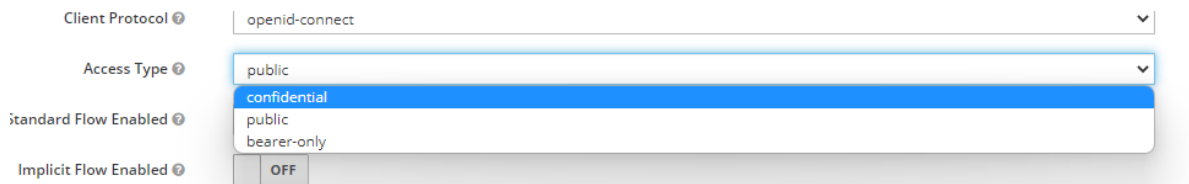
# Keycloak Plan

## Stage 1

1. Secure All pages that are meant to be secured in Order Online Website

Consideration – The Keycloak connection / adaptor code must be dynamic. Example given below:

The client in keycloak must be set as a confidential client. Therefore, you will also need to provide the secret set by the client on the code dynamically (not hard coded in the code)



The screenshot shows the Keycloak client configuration interface. The 'Client Protocol' is set to 'openid-connect'. The 'Access Type' dropdown menu is open, showing options: 'public', 'confidential' (highlighted in blue), 'public', and 'bearer-only'. The 'Standard Flow Enabled' checkbox is checked. The 'Implicit Flow Enabled' checkbox is unchecked, with a toggle switch set to 'OFF'.

The config for a confidential client should look like this:

```
{
  "realm": "demo-public",
  "auth-server-url": "https://identity.bettertechsolutions.net/auth/",
  "ssl-required": "external",
  "resource": "TestClient",
  "verify-token-audience": true,
  "credentials": {
    "secret": "06a76897-f10d-4c27-a600-7a79a23cd26c"
  },
  "use-resource-role-mappings": true,
  "confidential-port": 0
}
```

The above three variables in bold red will be different for each deployment as well. Global configuration like this must be stored outside the code so that it can be manipulated before every deployment without touching the code.

Typically for the Order Online Website the naming convention of the realm is **“organisation\_name-public”**

The URL naming convention for the keycloak host will be:

**“https://sso.the\_fully\_qualified\_domain\_for\_the\_organisation/auth/”**

2. After every Login Keycloak redirects to the initiator page with some information of the logged in user. The data must be captured on the application and processed accordingly.

For example:

- a. An existing user has logged in and the application DB is aware of the user. Use the data passed by keycloak to validate that user and show relevant info.
  - b. A user has registered and logged on for the first time. The Application DB is not aware of the user. Capture the data and create relevant profile / info for the user
  - c. In some cases, username, email address, first name and Surname can be changed at keycloak end. Upon a user login the new information will be passed on to the application against its unique user identifier. The system should capture the change and apply accordingly.
3. Implement Logout. The logout must be implemented for the corresponding session i.e. logout must happen for the relevant keycloak client only for this site. This is important as there is also a feature to log a user out globally for all SSO session that is opened in keycloak. But in this case, we want to log out only the relevant session.

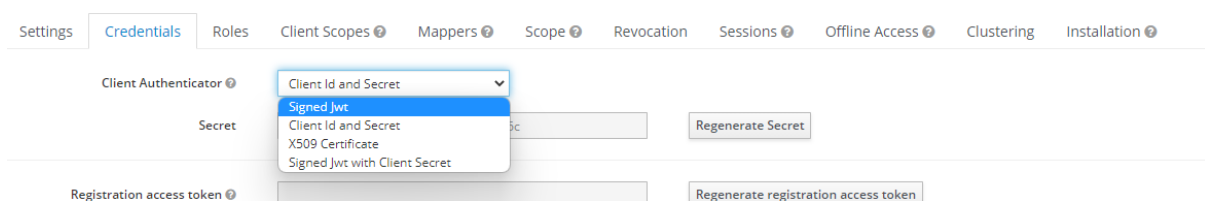
## Stage 2

1. Secure all Back-office Pages. For this site the keycloak client to be created in “otganisation\_name-local” realm. Similarly the realm name and keycloak server address will be dynamic for every deployment.
2. Implement a logout trigger for the relevant login in this site.
3. Implement a basic permission mapping for the logged in user for various functions in the back office. Such as; a standard user can only operate the order process (accept, decline amend etc.) and a privileged user can manage other tasks (create, amend menu, discounts and so on)

## Stage 3

1. Secure API Layer with Keycloak. This needs to be reviewed and the best approach must be discussed. It is possible to create a specific client in the realm for the API backend and that can be set as bearer only. It also provides different types of credentials therefore most relevant method should be used.

Example:



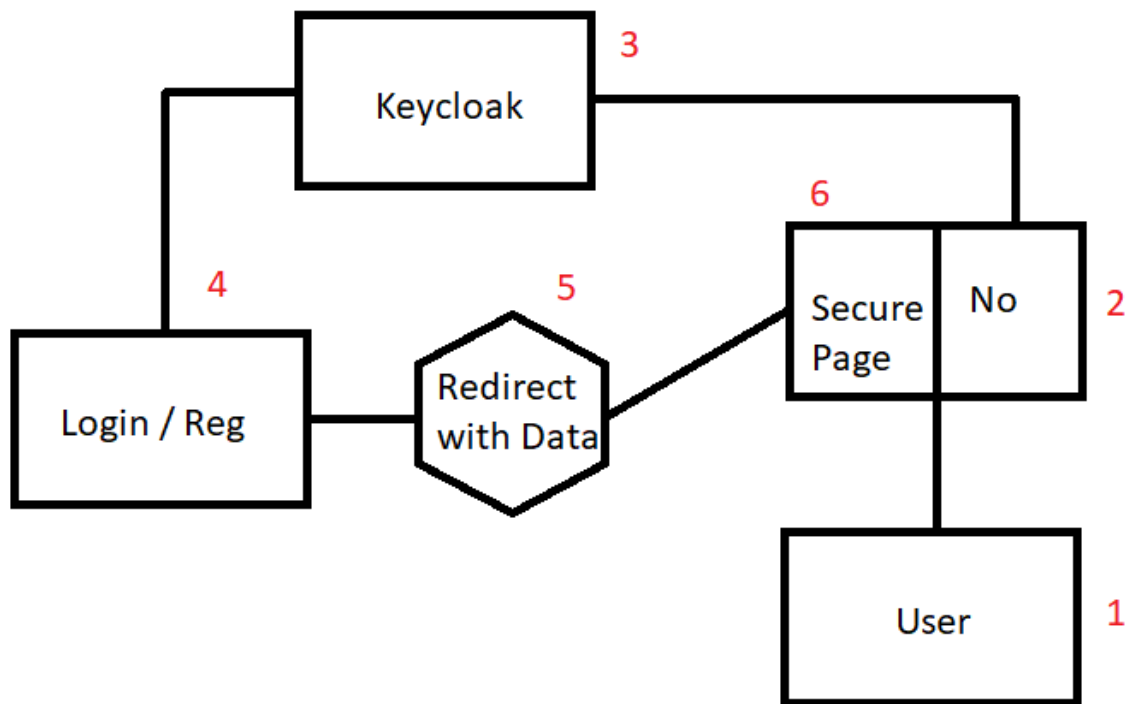
Again this must be noted that any reference to keycloak server URL and any realm or other variables will differ for each deployment therefore they must be dynamically picked by the code.

## Stage 4

1. Create Customised themes for the 4 Keycloak UI (Login, Account, Admin Console, Email) using HTML and CSS

## References

### Keycloak Standard Authentication Flow



### Secure React Routes & Component with Keycloak

<https://cagline.medium.com/authenticate-and-authorize-react-routes-component-with-keycloak-666e85662636>

### Securing Node.js Express REST APIs with Keycloak

<https://medium.com/devops-dudes/securing-node-js-express-rest-apis-with-keycloak-a4946083be51>

### Customising Keycloak Themes

<https://dev.to/sandeepbalachandran/how-to-add-custom-theme-in-keycloak-2854>