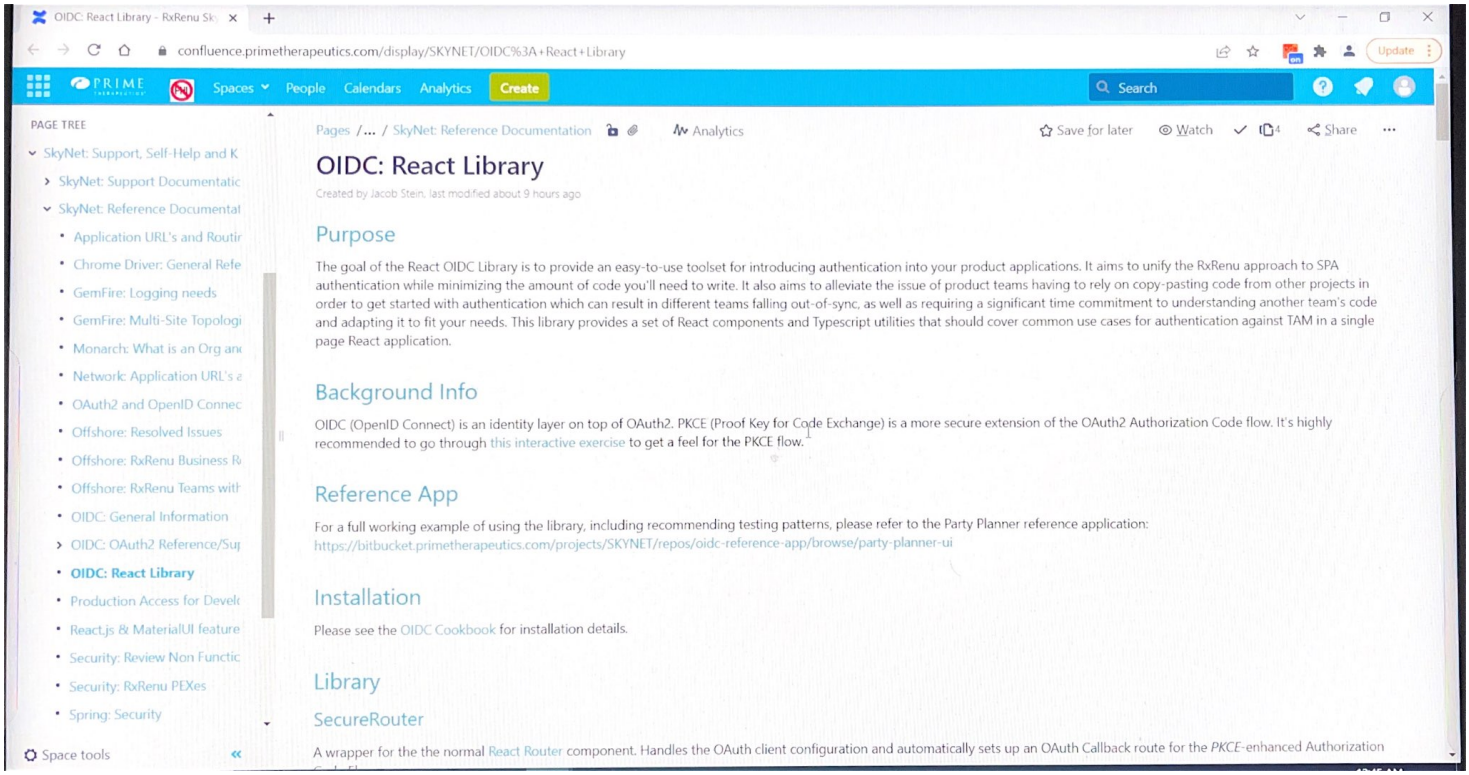
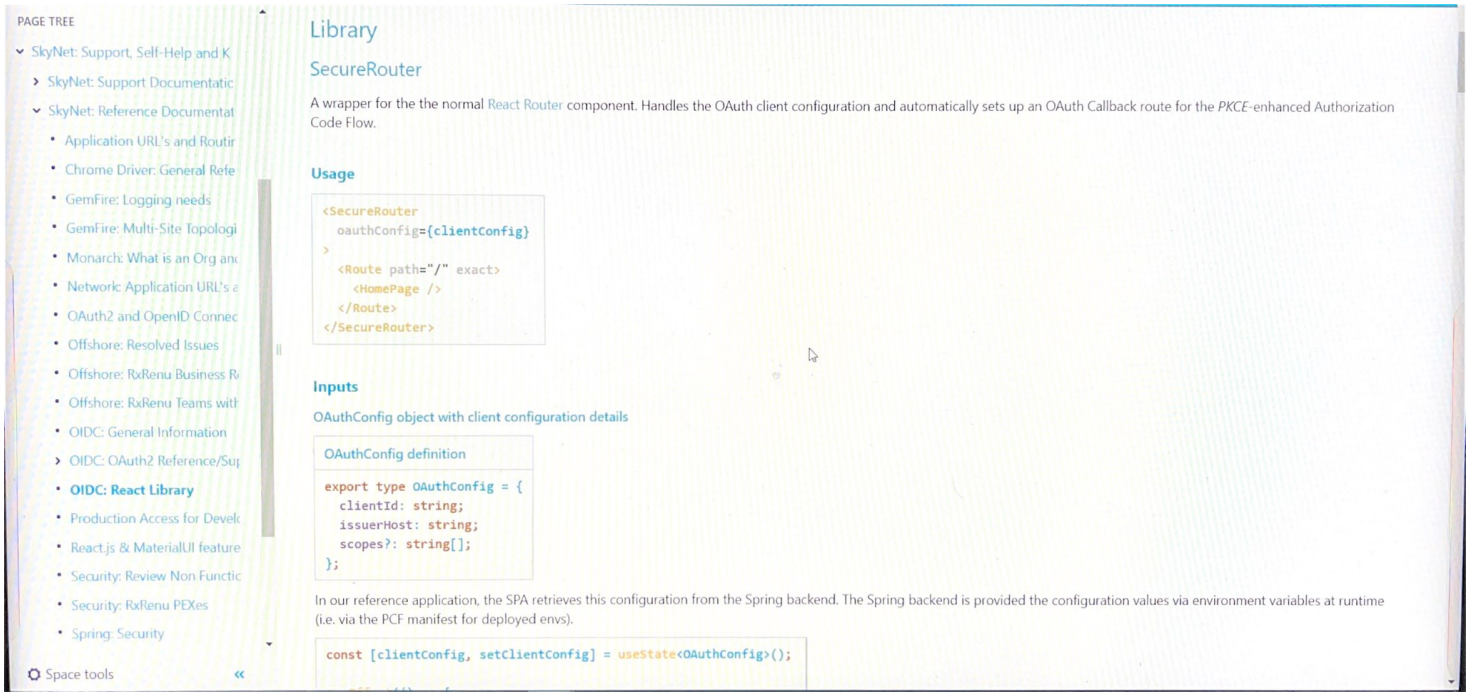
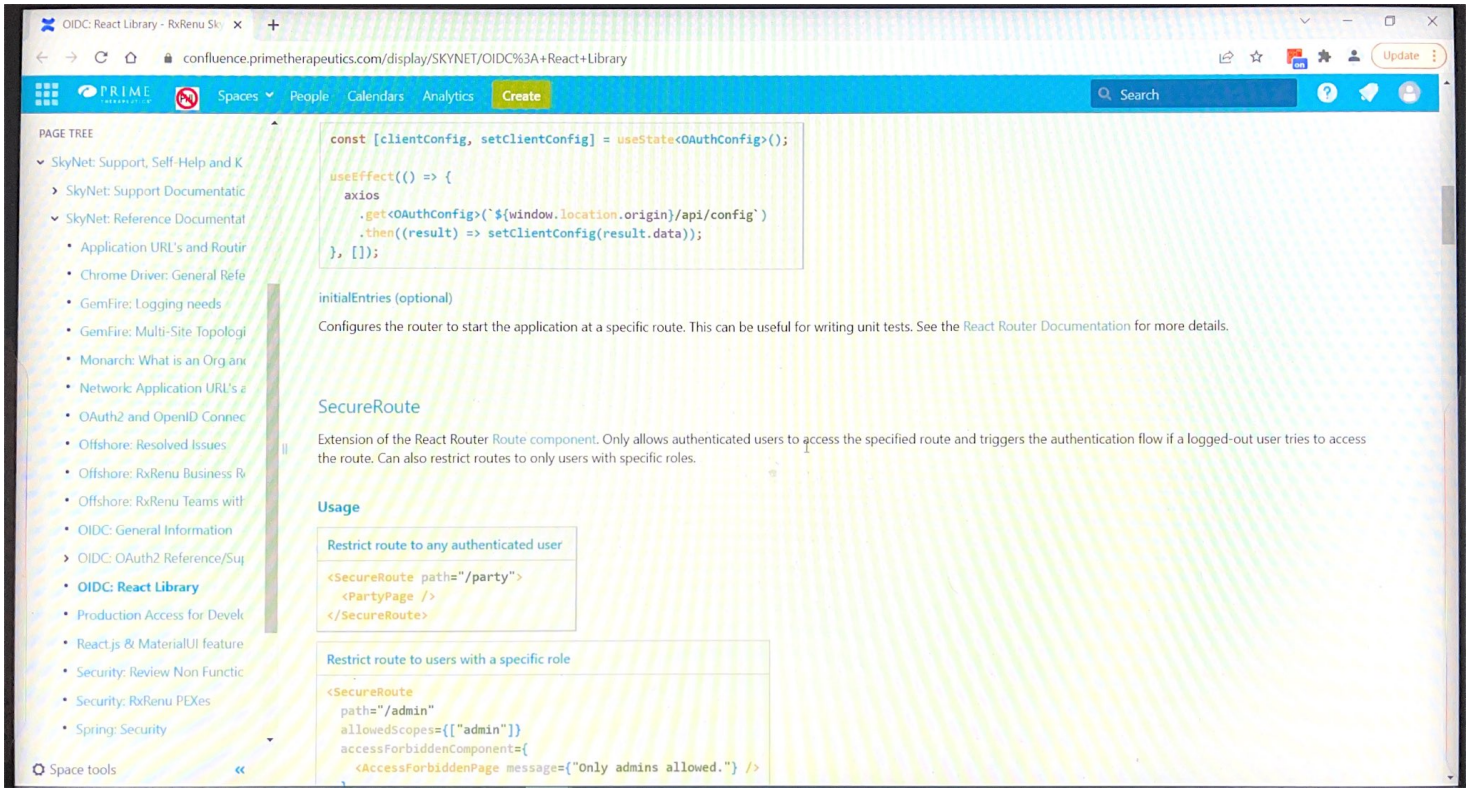
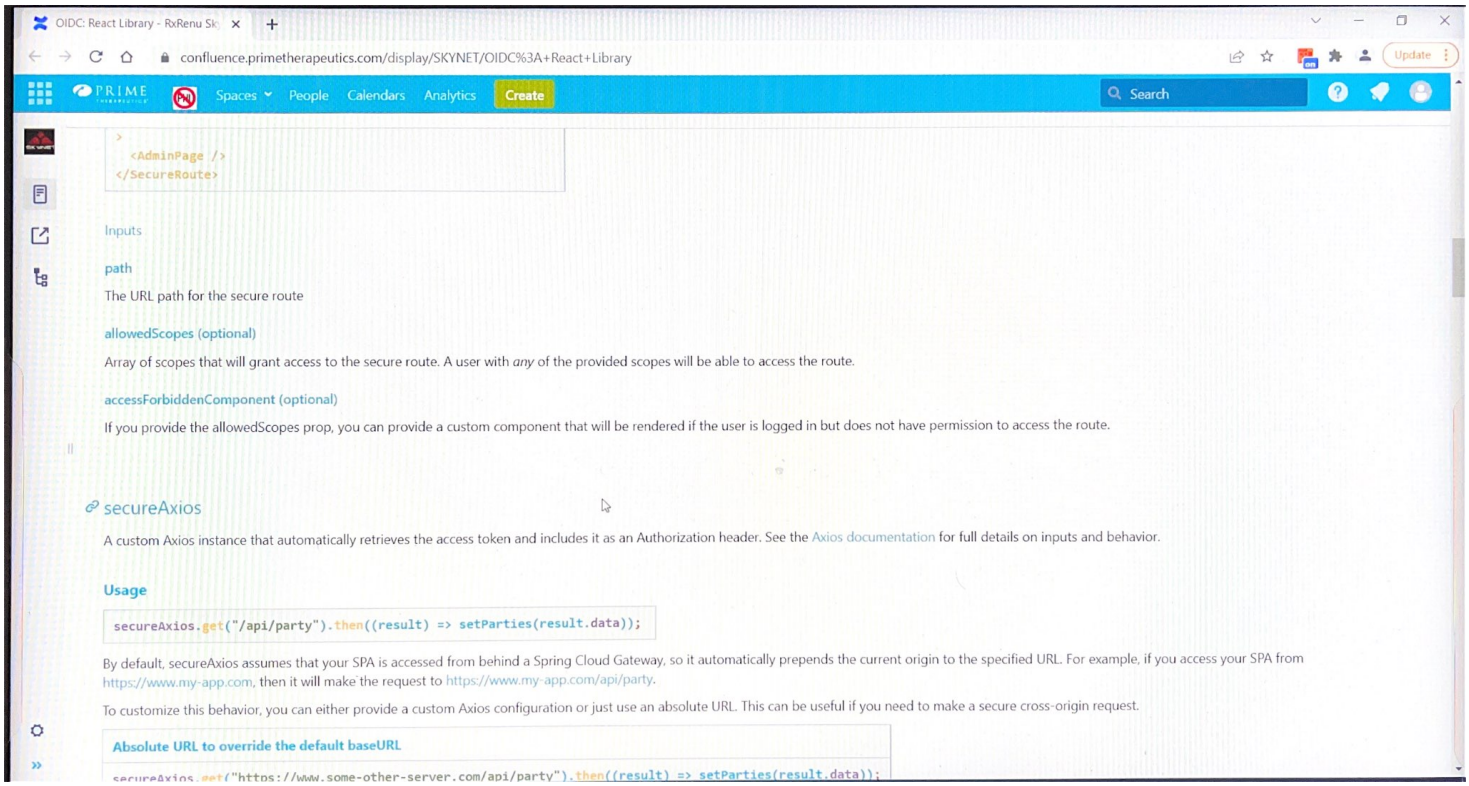
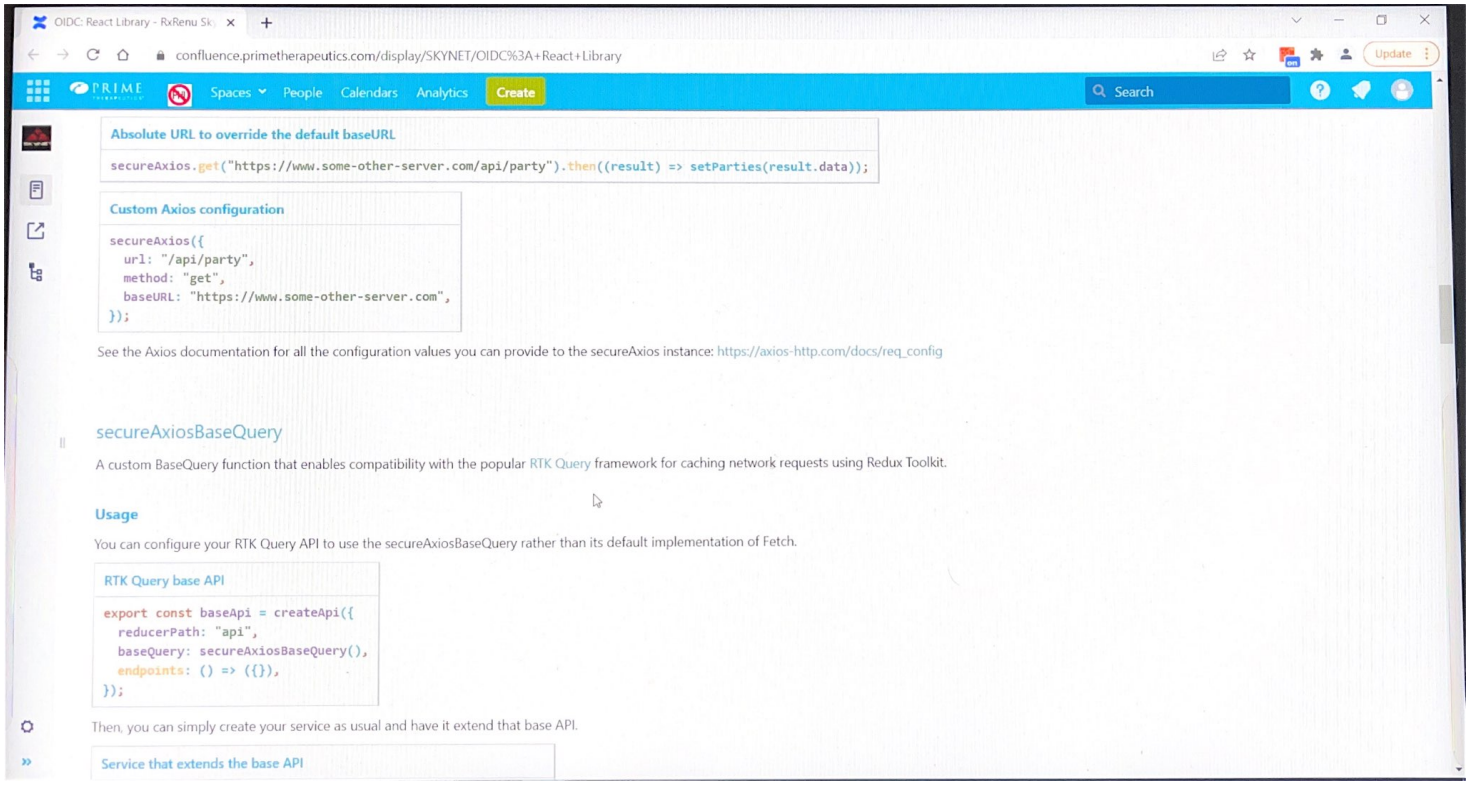
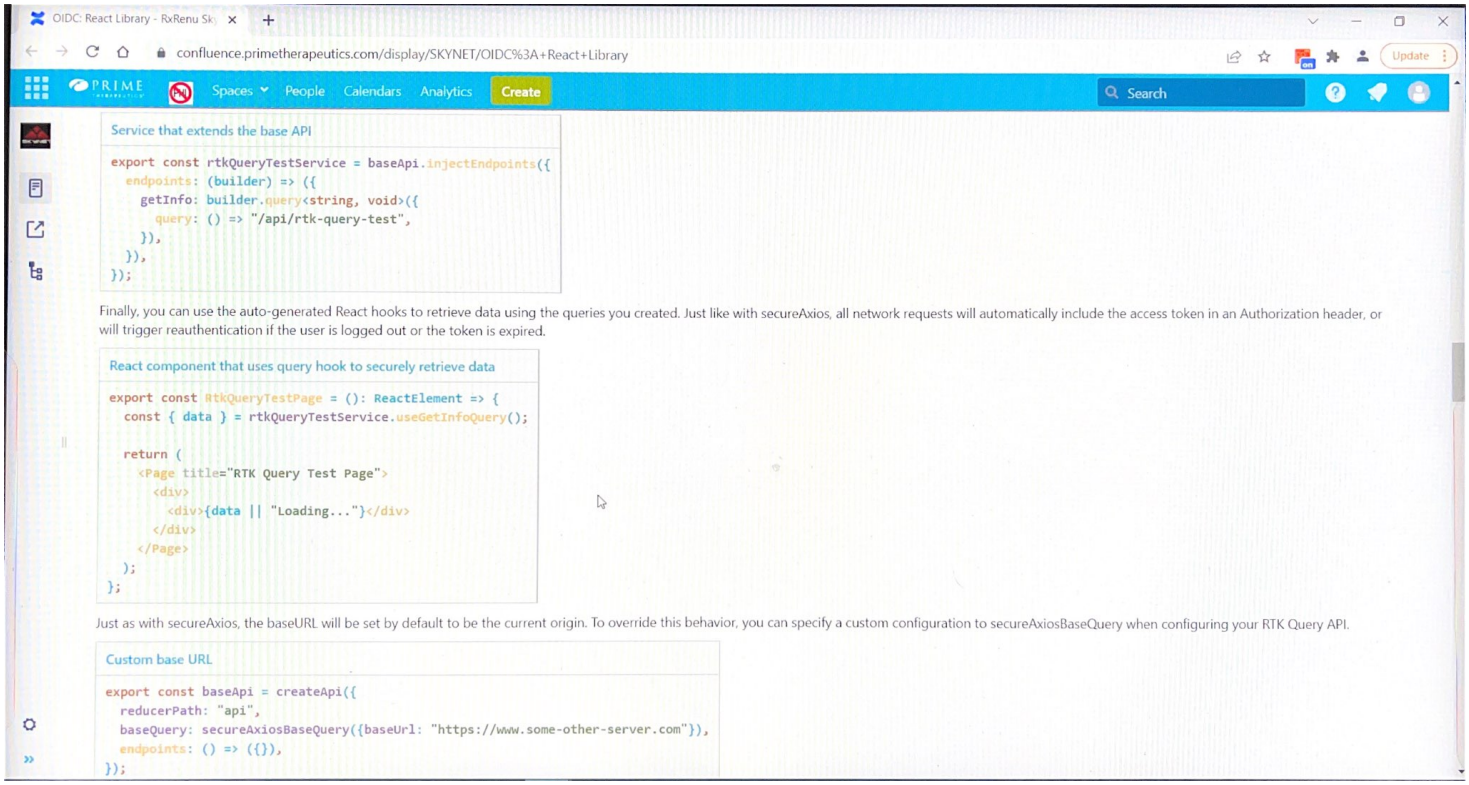

OIDC: React Library - RxRenu Sk
confluence.primetherapeutics.com/display/SKYNET/OIDC%3A+React+Library
Update
PRIME
Spaces
People
Calendars
Analytics
Create
Search
PAGE TREE
SkyNet: Support, Self-Help and K
SkyNet: Support Documentatic
SkyNet: Reference Documentat
Application URL's and Routir
Chrome Driver: General Refe
GemFire: Logging needs
GemFire: Multi-Site Topologi
Monarch: What is an Org anc
Network: Application URL's a
OAuth2 and OpenID Connec
Offshore: Resolved Issues
Offshore: RxRenu Business R
Offshore: RxRenu Teams with
OIDC: General Information
OIDC: OAuth2 Reference/Sup
OIDC: React Library
Production Access for Develc
React.js & MaterialUI feature
Security: Review Non Functic
Security: RxRenu PEXes
Spring: Security
Space tools
Pages / ... / SkyNet: Reference Documentation
Analytics
Save for later
Watch
Share
OIDC: React Library
Created by Jacob Stein, last modified about 9 hours ago
Purpose
The goal of the React OIDC Library is to provide an easy-to-use toolset for introducing authentication into your product applications. It aims to unify the RxRenu approach to SPA authentication while minimizing the amount of code you'll need to write. It also aims to alleviate the issue of product teams having to rely on copy-pasting code from other projects in order to get started with authentication which can result in different teams falling out-of-sync, as well as requiring a significant time commitment to understanding another team's code and adapting it to fit your needs. This library provides a set of React components and Typescript utilities that should cover common use cases for authentication against TAM in a single page React application.
Background Info
OIDC (OpenID Connect) is an identity layer on top of OAuth2. PKCE (Proof Key for Code Exchange) is a more secure extension of the OAuth2 Authorization Code flow. It's highly recommended to go through this interactive exercise to get a feel for the PKCE flow.
Reference App
For a full working example of using the library, including recommending testing patterns, please refer to the Party Planner reference application:
https://bitbucket.primetherapeutics.com/projects/SKYNET/repos/oidc-reference-app/browse/party-planner-ui
Installation
Please see the OIDC Cookbook for installation details.
Library
SecureRouter
A wrapper for the the normal React Router component. Handles the OAuth client configuration and automatically sets up an OAuth Callback route for the PKCE-enhanced Authorization


PAGE TREE
SkyNet: Support, Self-Help and K
SkyNet: Support Documentatic
SkyNet: Reference Documentat
Application URL's and Routir
Chrome Driver: General Refe
GemFire: Logging needs
GemFire: Multi-Site Topologi
Monarch: What is an Org an
Network: Application URL's a
OAuth2 and OpenID Connec
Offshore: Resolved Issues
Offshore: RxRenu Business R
Offshore: RxRenu Teams with
OIDC: General Information
OIDC: OAuth2 Reference/Sup
OIDC: React Library
Production Access for Develc
React.js & MaterialUI feature
Security: Review Non Functic
Security: RxRenu PEXes
Spring: Security
Space tools
Library
SecureRouter
A wrapper for the the normal React Router component. Handles the OAuth client configuration and automatically sets up an OAuth Callback route for the PKCE-enhanced Authorization Code Flow.
Usage
<SecureRouter
oauthConfig={clientConfig}
>
<Route path="/" exact>
<HomePage />
</Route>
</SecureRouter>
Inputs
OAuthConfig object with client configuration details
OAuthConfig definition
export type OAuthConfig = {
clientId: string;
issuerHost: string;
scopes?: string[];
};
In our reference application, the SPA retrieves this configuration from the Spring backend. The Spring backend is provided the configuration values via environment variables at runtime (i.e. via the PCF manifest for deployed envs).
const [clientConfig, setClientConfig] = useState<OAuthConfig>();


OIDC: React Library - RxRenu Sky
confluence.primetherapeutics.com/display/SKYNET/OIDC%3A+React+Library
Update
PRIME
Spaces
People
Calendars
Analytics
Create
Search
PAGE TREE
SkyNet: Support, Self-Help and K
SkyNet: Support Documentatic
SkyNet: Reference Documentat
Application URL's and Routir
Chrome Driver: General Refe
GemFire: Logging needs
GemFire: Multi-Site Topologi
Monarch: What is an Org and
Network: Application URL's a
OAuth2 and OpenID Connec
Offshore: Resolved Issues
Offshore: RxRenu Business R
Offshore: RxRenu Teams with
OIDC: General Information
OIDC: OAuth2 Reference/Su
OIDC: React Library
Production Access for Develc
React.js & MaterialUI feature
Security: Review Non Functic
Security: RxRenu PEXes
Spring: Security
Space tools
const [clientConfig, setClientConfig] = useState<OAuthConfig>();
useEffect(() => {
axios
.get<OAuthConfig>(`${window.location.origin}/api/config`)
.then((result) => setClientConfig(result.data));
}, []);
initialEntries (optional)
Configures the router to start the application at a specific route. This can be useful for writing unit tests. See the React Router Documentation for more details.
SecureRoute
Extension of the React Router Route component. Only allows authenticated users to access the specified route and triggers the authentication flow if a logged-out user tries to access the route. Can also restrict routes to only users with specific roles.
Usage
Restrict route to any authenticated user
<SecureRoute path="/party">
<PartyPage />
</SecureRoute>
Restrict route to users with a specific role
<SecureRoute
path="/admin"
allowedScopes={["admin"]}
accessForbiddenComponent={
<AccessForbiddenPage message={"Only admins allowed."} />


OIDC: React Library - RxRenu Sk
confluence.primetherapeutics.com/display/SKYNET/OIDC%3A+React+Library
Update
PRIME
Spaces
People
Calendars
Analytics
Create
Search
>
<AdminPage />
</SecureRoute>
Inputs
path
The URL path for the secure route
allowedScopes (optional)
Array of scopes that will grant access to the secure route. A user with any of the provided scopes will be able to access the route.
accessForbiddenComponent (optional)
If you provide the allowedScopes prop, you can provide a custom component that will be rendered if the user is logged in but does not have permission to access the route.
secureAxios
A custom Axios instance that automatically retrieves the access token and includes it as an Authorization header. See the Axios documentation for full details on inputs and behavior.
Usage
secureAxios.get("/api/party").then((result) => setParties(result.data));
By default, secureAxios assumes that your SPA is accessed from behind a Spring Cloud Gateway, so it automatically prepends the current origin to the specified URL. For example, if you access your SPA from https://www.my-app.com, then it will make the request to https://www.my-app.com/api/party.
To customize this behavior, you can either provide a custom Axios configuration or just use an absolute URL. This can be useful if you need to make a secure cross-origin request.
Absolute URL to override the default baseURL
secureAxios.get("https://www.some-other-server.com/api/party").then((result) => setParties(result.data));


OIDC: React Library - RxRenu Sk
confluence.primetherapeutics.com/display/SKYNET/OIDC%3A+React+Library
Update
PRIME
Spaces
People
Calendars
Analytics
Create
Search
Absolute URL to override the default baseURL
secureAxios.get("https://www.some-other-server.com/api/party").then((result) => setParties(result.data));
Custom Axios configuration
secureAxios({
url: "/api/party",
method: "get",
baseURL: "https://www.some-other-server.com",
});
See the Axios documentation for all the configuration values you can provide to the secureAxios instance: https://axios-http.com/docs/req_config
secureAxiosBaseQuery
A custom BaseQuery function that enables compatibility with the popular RTK Query framework for caching network requests using Redux Toolkit.
Usage
You can configure your RTK Query API to use the secureAxiosBaseQuery rather than its default implementation of Fetch.
RTK Query base API
export const baseApi = createApi({
reducerPath: "api",
baseQuery: secureAxiosBaseQuery(),
endpoints: () => ({}),
});
Then, you can simply create your service as usual and have it extend that base API.
Service that extends the base API


OIDC: React Library - RxRenu Sk
confluence.primetherapeutics.com/display/SKYNET/OIDC%3A+React+Library
Update
PRIME
Spaces
People
Calendars
Analytics
Create
Search
Service that extends the base API
export const rtkQueryTestService = baseApi.injectEndpoints({
endpoints: (builder) => ({
getInfo: builder.query<string, void>({
query: () => "/api/rtk-query-test",
}),
}),
});
Finally, you can use the auto-generated React hooks to retrieve data using the queries you created. Just like with secureAxios, all network requests will automatically include the access token in an Authorization header, or will trigger reauthentication if the user is logged out or the token is expired.
React component that uses query hook to securely retrieve data
export const RtkQueryTestPage = (): ReactElement => {
const { data } = rtkQueryTestService.useGetInfoQuery();
return (
<Page title="RTK Query Test Page">

{data || "Loading..."}

</Page>
);
};
Just as with secureAxios, the baseURL will be set by default to be the current origin. To override this behavior, you can specify a custom configuration to secureAxiosBaseQuery when configuring your RTK Query API.
Custom base URL
export const baseApi = createApi({
reducerPath: "api",
baseQuery: secureAxiosBaseQuery({baseUrl: "https://www.some-other-server.com"}),
endpoints: () => ({}),
});