Web Portal Session Management

Introduction

Session management deals with areas around maintaining a logged in state for an authenticated user.

By design, IDEA does not use **Cookies** or **Server Side session management** and all APIs are strictly stateless APIs.

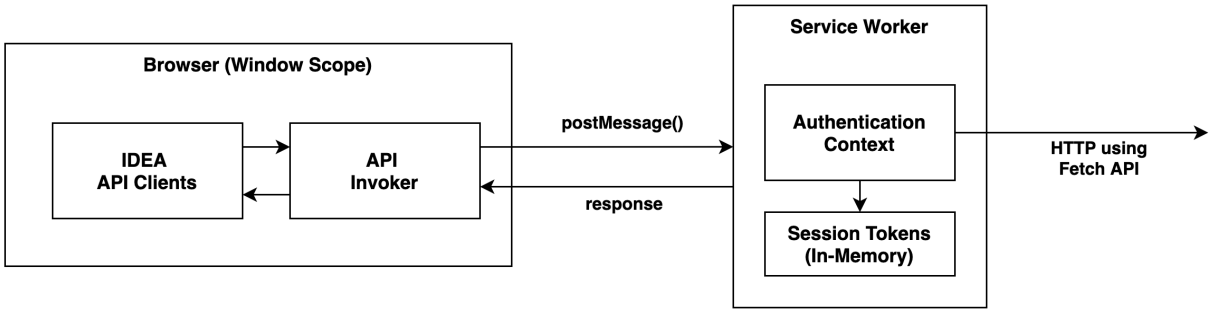
There are quite a few reason to not use cookies, but the primary ones are:

- Consistent API Authentication mechanism across all modules. We no longer need to worry about handling authentication and authorization separately (Authorization Header vs Cookies) from Web Portal vs API invocations from Native Mobile clients.
- Cookies are notoriously sticky and are passed around for every request back and forth from client to server, even for requests that do not necessarily need any Cookies. (eg. static assets such as javascript, images etc.)
- All APIs have to add and deal with additional logic around CSRF attacks or implement some sort of Secure Cookie mechanisms.

Server side session management does not help with scaling the infrastructure and we need to keep the APIs stateless. This document outlines the supported session management mechanisms implemented in the Web Portal so that user can remain logged in.

Schemes

Session Tokens saved in Service Worker (In-Memory)



(Source: session-management-service-worker.drawio)

In this mode, once the user signs in to the web portal, the user session will not expire until:

- all open tabs for the Web Portal have been closed; OR
- browser or computer is restarted; OR
- the validity of the refresh token expires (default: 30 days); OR
- user explicitly logs out from the Web Portal by clicking the logout CTA.

Closing Browser Tabs

If the user closes all active tabs for the Web Portal, the duration after which an active Service Worker is terminated depends upon the Browser being used. For Chrome, this duration is 30 seconds.

Below are some the key points summarizing the service worker based implementation for Session Management:

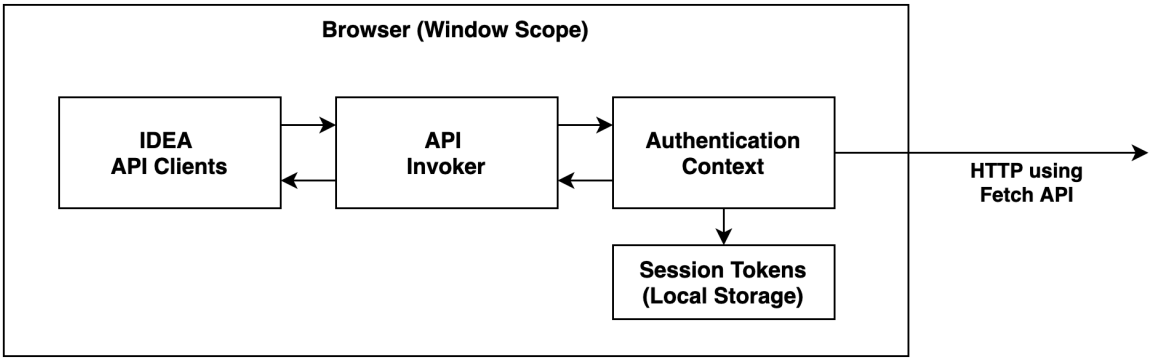
- At any point in time, single AuthenticationContext object may exists associated to an active Service Worker.
- All IDEA API invocations are routed via a Service Worker using the postMessage() interface. Request/Response is achieved via the MessageChannel interface in Channel Messaging API.
- Any script running in the Window object will not be able to access the session tokens saved in-memory within the active ServiceWorker. Communication with the service worker via the postMessage() interface is only possible via the scripts loaded over the current origin. Using this mechanism, the IDEA WebPortal implementation is free of any XSS attacks related to storing sensitive information in Local Storage.

- The refresh token is never exposed outside Service Worker's memory and access token renewals are handled automatically when the access token is nearing expiration.
- Clients can request for access token, if required and service worker may return the in-memory access token upon request. If the access token is expired, a renewed access token will be returned.
- This mode of operation is the default mode for Web Portal will be automatically activated for production deployments when a custom domain name and a valid certificate is configured for external ALB.
- In order for this mode to be activated, Web Portal must be served over a **valid** / **trusted** TLS context. (**Note:** External ALB configured using self-signed certificates will not work, unless the certificates are trusted by browser / operating system's trust store.)
- In the event Service Worker activation fails, the implementation falls back to local storage based session management.

For Development and Testing:

- Localhost is a considered as a secured origin by web browsers.
- The ALB endpoint must be configured using a valid ACM certificate and DNS.

Session Tokens saved in Browser Local Storage



(Source: session-management-local-storage.drawio)

Warning

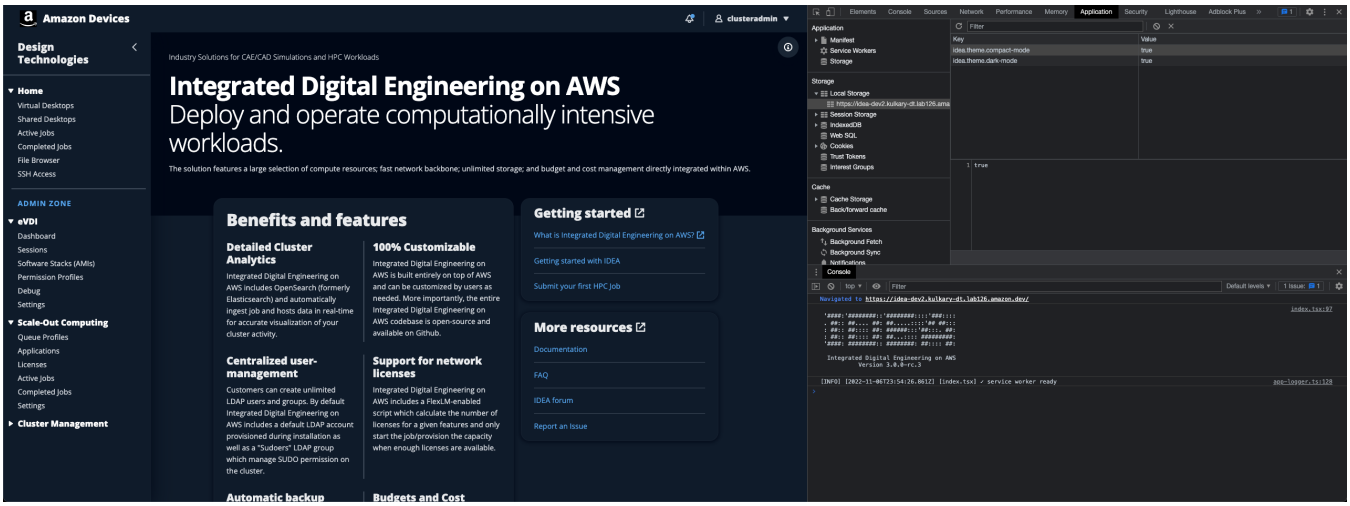
Any script that can be run in the Window scope, loaded from the same origin or different origin, can access the browser's local storage and read the access and refresh tokens, leading to possible XSS attacks.

Below are points summarizing the local storage based implementation for Session Management:

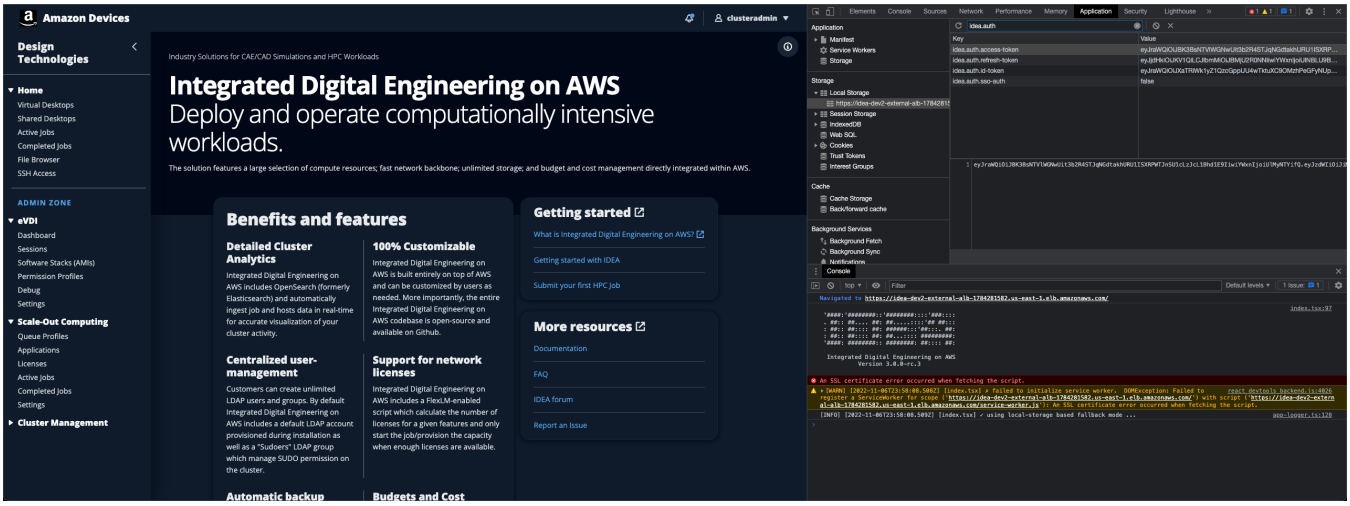
- The session tokens (AccessToken and RefreshToken) are saved in Browser's local storage.
- A new AuthenticationContext object is initialized upon page load for each browser tab by reading the session tokens from local storage.
- The session may remain active even when user closes all Web Portal tabs or across browser restarts.
- The session can expire only when user explicitly logs out from the web-portal or the refresh token expires.

Appendix

Appendix A) Service Worker based Session Management



Appendix B) Fallback to Local Storage based Session Management (In-Secure TLS Context)



Appendix C) Local Storage based Session Management (Local Development and Testing)