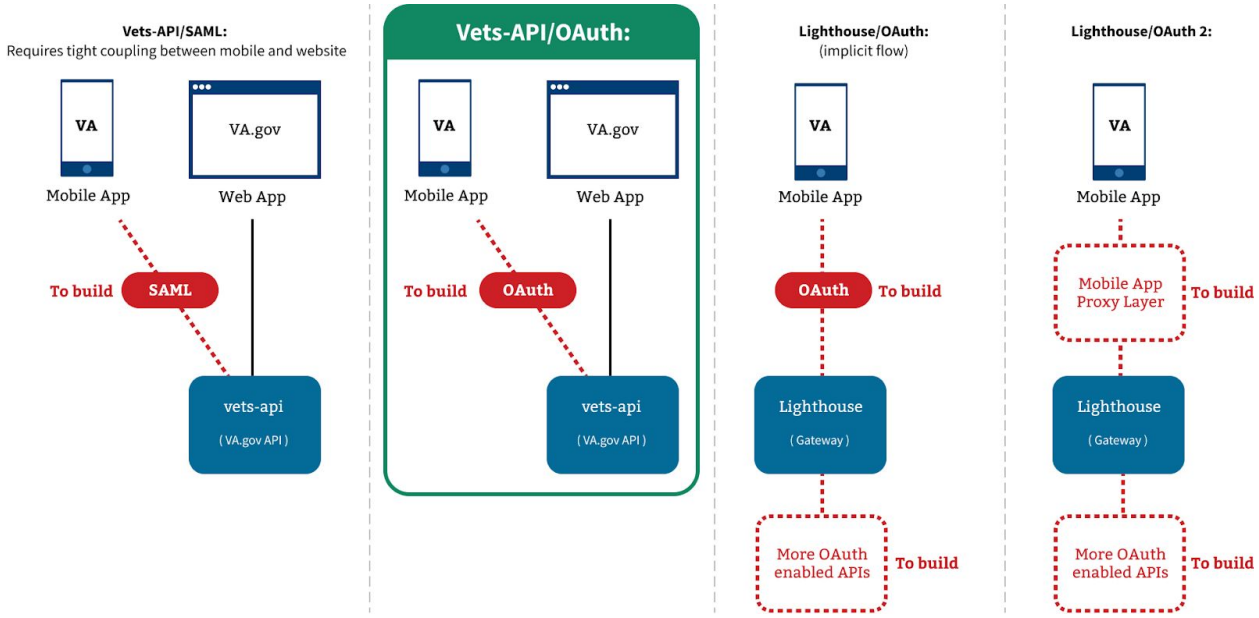
Ad Hoc

Mobile App Discovery Sprint - Login

TL;DR

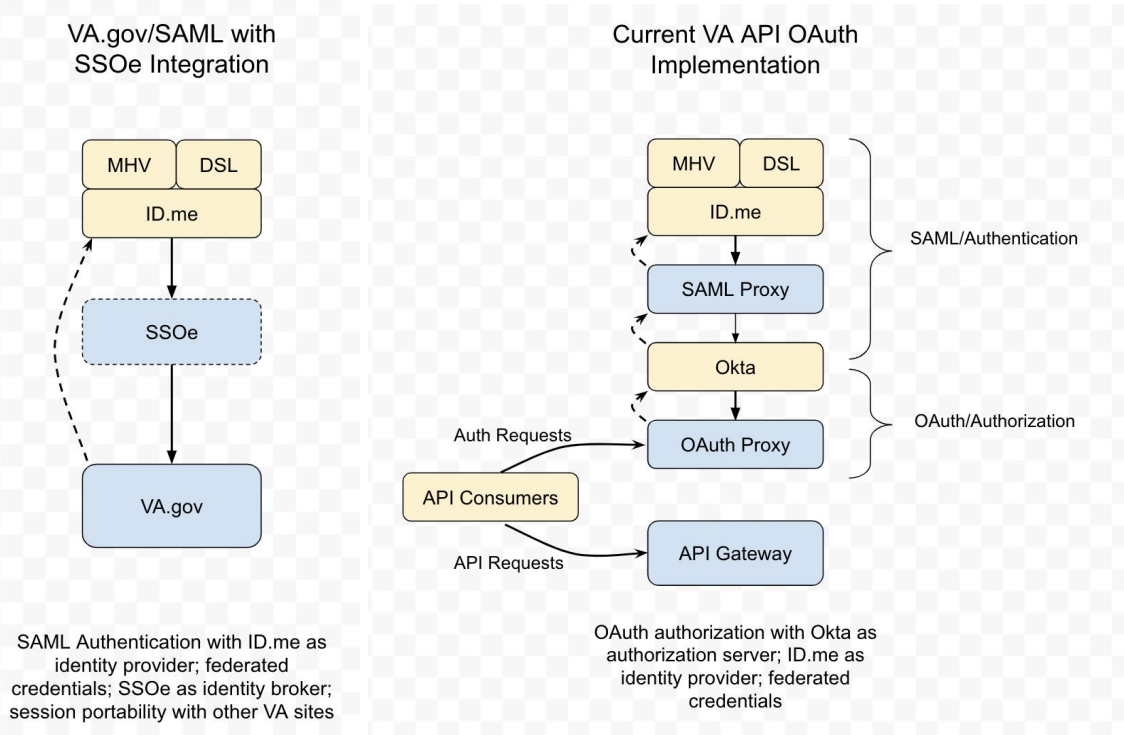
- With some development work, it would be technically feasible to use either VA.gov/SAML or Lighthouse/OAuth for login on a mobile app.
- However — based on the technical, Veteran experience, and logistical constraints and opportunities outlined in this document — our recommendation is to enable the VA.gov-centric part of vets-api to accept OAuth.



How does login currently work?

VA.gov: Users can log in to VA.gov using one of three credential providers (CSP): ID.me, MHV, DS Logon. Each of these CSPs uses ID.me as the SAML identity broker.

- The end state of VA.gov sign-in is that the user's browser has a VA.gov session cookie containing a session token. When the vets-website frontend makes API requests, vets-api checks that the session token is valid and unexpired before allowing the API request. The expiry of the session token is extended with each API request, and expires after 30 minutes of inactivity.
- A VA.gov session cookie/token grants access to the entire surface area of vets-api, because of the implicit assumption that users are primarily veterans accessing their own data.



Lighthouse: Third party apps such as the Apple Health app or USA Jobs trigger an OAuth authorization request. The authorization server (powered by Okta) first authenticates the user (using the same three credential options and again brokered by ID.me), then requests consent from the user to allow the app to access specific pieces of their VA data.

- The end state of Lighthouse OAuth process is that the app holds an access token that can be passed with subsequent API requests. A given API endpoint will only accept a token that contains the right scope(s) for the data that it deals in. Access tokens have an expiry of 1 hour.
- An app may also request "offline access", in which case it will be issued a refresh token. This token is long-lived, potentially expiring after a year or more. The app can use the refresh token to request a new access token without requiring that the user re-authenticate.
- A lighthouse OAuth token only grants access to specific scopes of data; the assumption is that a third-party app will only need specific pieces of data (e.g. health data, or military service history) and shouldn't be given access to other unrelated pieces of veteran data.

API Surface Areas

The vets-api Rails application exposes API endpoints for both VA.gov and Lighthouse. The endpoints are similar in nature, but differ in intended audience and authorization mechanisms, and are managed by separate programs.

VA.gov endpoints

- API covers all resources exposed by VA.gov website.
- All resources with Veteran-specific data are accessed using the above cookie/token.

- API is not "private" per se, but is more closely coupled with the VA.gov front end in that there is a 1:1 mapping between VA.gov features and API endpoints.
- Operated by VSP/VSA programs.

Lighthouse endpoints

- API covers resources for which a likely external or internal consumer other than VA.gov was identified.
- APIs with Veteran-specific data are accessed using the above OAuth process.
- APIs with no Veteran-specific data, or with no online veteran interaction (e.g. VSO/call center use cases) are protected by a per-consumer API key.
- Operated by VA API program.

Lighthouse as a program *could* have taken the approach of enabling the entire VA.gov API surface area for OAuth. This was not done, in favor of prioritizing specific APIs with identified consumer bases, where each API could be thoughtfully enabled, documented, etc. This approach also minimized working conflict across programs

How could these login methods work on mobile?

VA.gov/SAML:
A mobile application can embed a web view to trigger a SAML authentication flow, and obtain the resulting session token to use for subsequent API calls.

The current approach where the session token is maintained in a browser cookie would need to be modified slightly to return the token in a way that the mobile app could obtain it from the web view. This is feasible (VA.gov's frontend used to use this approach instead of a cookie), but would entail implementing a separate SAML callback endpoint in vets-api.

Given the current coupling between VA.gov frontend and API, mobile app development would need to overcome the organizational/communication barriers to keep the VA.gov-centric API synchronized with mobile use cases. As we've seen with Lighthouse, inertia tends towards separate teams forking API implementations.

Lighthouse/OAuth: Mobile applications likewise can embed a web view to enable the OAuth process, at the end of which the refresh and access tokens can be returned to the application for subsequent API calls. This is a well-trod pattern for mobile applications.

There are a few flavors of OAuth. The one currently supported by Lighthouse is not ideal for single-page applications or mobile apps, because it requires storing a client secret in browser-side Javascript or on-device. Current Lighthouse consumers work around this by having a server-side proxy that mediates between their app and the Lighthouse gateway. (For example, the Apple health app makes a request to an Apple server-side app, which makes OAuth-enabled API requests to Lighthouse FHIR APIs).

Lighthouse is geared towards supporting a broad base of API consumers and so its API surface area tends to be more general-purpose, better documented, etc. Whether this is important in practice for what would be an internal VA team implementing a mobile app, is unclear.

Token Duration and Mobile-Centric Authentication

One downside of the VA.gov-style cookie approach is that the cookie expires after 30 minutes of session inactivity. After expiring, a user must re-authenticate to establish a new session. This would mean a user of a mobile app would need to re-authenticate using username/password/2FA each time they use the application after inactivity.

The long-lived refresh token paradigm provided by OAuth presents the interesting possibility that an app can remain authorized for a long period of time without requiring re-authentication. This provides a viable approach for enabling biometric-controlled access to an app. The user would follow the OAuth flow *one time*, resulting in a long-lived refresh token issued to the mobile app. The mobile app would then enforce on-device biometric authentication before allowing any user interaction that required access to the refresh/access tokens.

Programs vs. APIs vs. Auth Mechanisms

Only a small subset of Veteran-data-related APIs are enabled for OAuth by the Lighthouse program. A mobile app that required a similarly broad range of data as VA.gov would warrant expanding this such that the entire VA.gov-facing surface area of vets-api could be enabled for OAuth. This is feasible, with concurrence of the VSP team. Authorization *should* be largely orthogonal to the data returned by an API resource, and access to a single API endpoint can potentially be authorized in multiple ways (session cookies, OAuth tokens, API keys, etc) if we desire it to be so.

In other words, the current coupling between VA.gov+SAML and Lighthouse+OAuth is not set in stone; those programs just happen to support two distinct authentication mechanisms. More likely, a mobile app wants to provide an API footprint similar to that used by VA.gov, but secured using OAuth as a technology choice to enable mobile development. A mobile application would primarily be a first-party (Veteran accessing their own data) scenario, and in that way a mobile application team likely has more affinity with other Veteran-Facing Service teams at VA than with external API consumers.

Risk: Current momentum is for VSP and Lighthouse to fork vets-api and evolve it independently. Lighthouse operates the Okta/OAuth implementation and trying to use it more widely across the VA.gov part of vets-api may be running counter to that momentum.

Prototype Blockers

VA.gov/SAML approach:
We're unable to fully test this approach at the moment, as it would require implementing a separate SAML callback endpoint in vets-api.

Lighthouse/OAuth approach:

Lighthouse does not currently support the "implicit" or PKCE OAuth flow that is preferred for native mobile development.

Implementation Blockers

For vets-api-wide OAuth:

- Implicit flow support in Okta/OAuth implementation
- Enable vets-api to accept/validate OAuth tokens in addition to the current session token-in-cookie mechanism.
- Define scopes suitable for all vets-api resources
- Figure out right granularity of consent for mobile app - does it request "wildcard" access to all veteran data, or piecemeal scopes for each part of the app?
- Implement scope-application whitelisting so that *only* the mobile app consumer could request broad access to data (other Lighthouse consumers should be restricted to scopes for which they've been approved).