***MyHealtheVet to Mobile Single Sign-on***

***Date: August 3, 2017***

# Introduction

The VA Mobile Framework (VAMF) has been using VA’s Identity Access Management (IAM) services with an older junction (pre AccessVA and SSOe terminology) that only included the DSLogon credentials from DMDC. This limits the number of Veterans who will have a credential for use with the Mobile applications (apps). It was determined that a single-sign on approach could be taken to allow MyHealtheVet (MHV) users to access mobile applications which today are mobile-enabled web applications. The existing VAMF Authorization Services is a customized OAuth solution which limited the ability of other portals such as MHV to interface with mobile apps. The mobile apps validate the OAuth 2 token within its environmental shared services which makes it difficult for external applications to integrate with the VAMF under this older architecture.

Mobile is moving to a new Docker/container based architecture using micro-services that will enable mobile applications to reduce the footprint of their services. This new architecture is referred to as NextGen within the VAMF and has other aspects that increase flexibility and interoperability with outside applications. It is with this NextGen architecture that MHV is integrating with the VAMF to allow MHV users to log into the MHV portal (under Liferay) and be forwarded to a Mobile web-application. This action will not require the user to re-authenticate and will automatically supply the MHV user’s identity and roles to the mobile application.

# VAMF NextGen Architecture

The VAMF NextGen architecture centers around the API Gateway – Microservice pattern (<http://microservices.io/patterns/apigateway.html>). The API Gateway provides the routing and load balancing across micro-services that are patient-centric. There is a user-service coupled with the API Gateway to support multiple Identity Providers (IDPs). JSON Web Tokens (JWT) are generated by the user-service and validated by the Mobile micro-services. The benefit of this approach is that the services can validate the JWTs and retrieve user identification information as well as authorize the use of their services without reaching out to any other service. In OAuth, an external source typically must confirm the token received and thus there are a lot of calls made to just authorize the use of a service.

The following figure shows an example of the Mobile Blue Button architecture in our development environment. The red flows show flows for the user to login and before the user-service creates the JWT token. Once logged in with the IDP, the flows become authenticated and there is a JWT token that is set in the browser (for convenience) and the services are expected to validate the JWT including validating that the session has not expired. Services work with the Consul configuration to obtain the signing authority for the environment and from then on, they are able to detect whether the JWT has been signed by the user-service and valid (in terms of expiration time, roles, and other attributes). There are no callbacks for the service to validate the JWT it receives as in the OAuth2 implementation previously used.

The web application on the user’s device can use the JWT supplied in a cookie by the user-service to conveniently find information about the user logged in such as their first name, last name and VA identifiers such as the ICN and EDIPI.

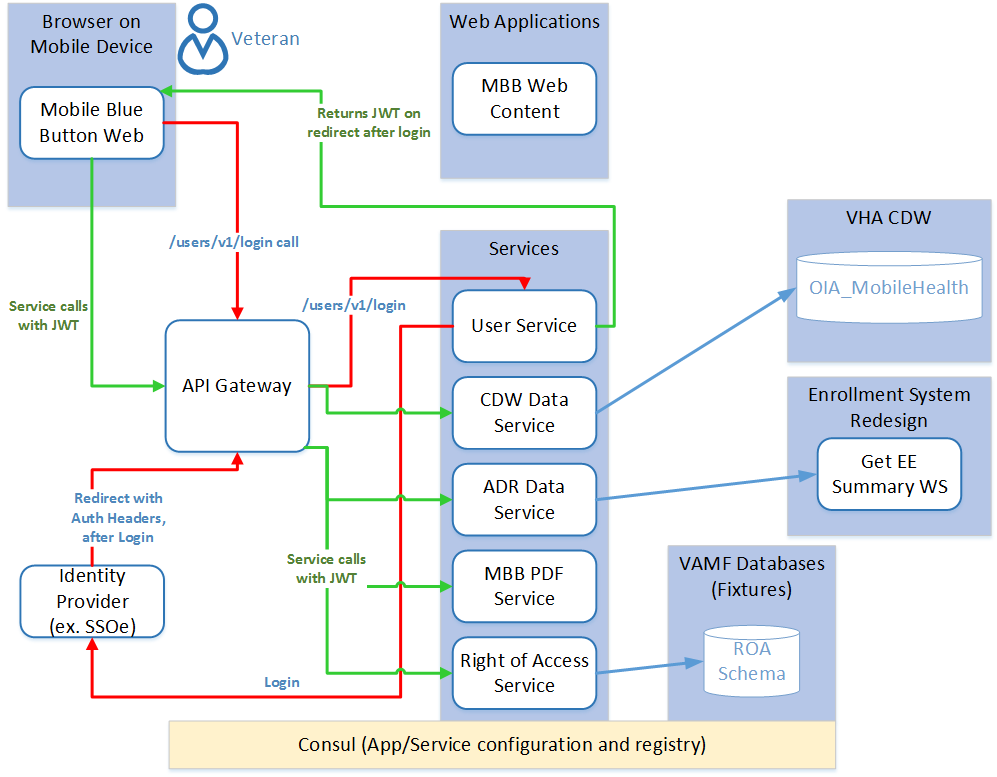


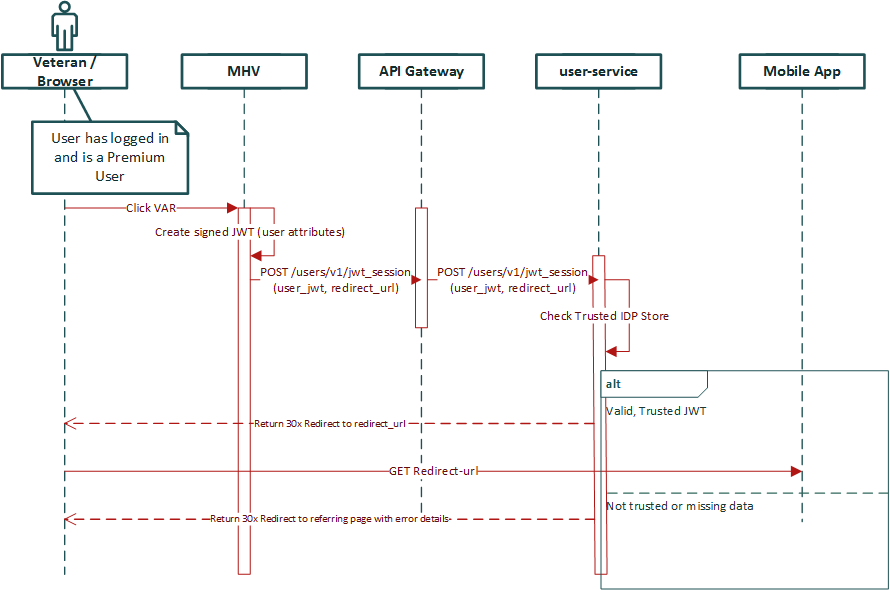
Figure 1: Example Application in NextGen Prior to MHV integration

This architecture was in place prior to being requested to implementing the integration with MHV. This would be used for SSOe integration with IAM’s IDPs.

# MHV Integration with VAMF NextGen

There are three phases to the integration with Mobile applications being performed. Only the first phase as described in the user stories in Section 5 are discussed in this section regarding the architecture. This is where a user logs into the MHV portal and wishes to then use a Mobile (web-based) application in the VAMF. MHV will present the Veteran Appointment Request (VAR) link only to users who have a Premium account (MHV may use other designations – but essentially a person who is confirmed to be a VA Patient).

The following logical diagram shows MHV as a black box to avoid having to go into details about MHV’s internal design. It shows the interactions that occur – which are very simple with the VAMF NextGen components.



Below is the documentation from the README.md file in the code.

**JWT Exchange / Security Token Service (STS)**

The user service supports exchanging JWT issued by other systems into tokens trusted by VAMF/NextGen services. This capability is valuable in several scenarios:

* **External Users**: A system not operating behind the APIGateway (i.e. outside of NextGen) wishes to enable their users to interact with NextGen apps/services.
* **Async Actions**: A system wishes to initiate an asynchronous or scheduled action on behalf of a user but the user is not currently logged in so a JWT is not available

To support these use cases, the user service supports HTTP POST actions on the /session endpoint for async / service-to-service requests and /jwt\_session endpoint for External User/browser actions. This endpoint takes a provided JWT signed by a trusted third party and re-signs/issues it for use within NextGen (after standard JWT validation occurs). Most importantly, it is important to not that the user service only allows trusted 3rd parties to preform this action- the issuer ID and the RSA public key of trusted third parties must be provided in the JWTISSUER\_CONFIG\_JSON environmental variable when the user service starts up:

{"rsaPublicKeyB64":

"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtxwSTqAC49xXxWIX/kpb4EWXjnpN8ii5FbjAOmubKA32ZVHC+OSj14uRkKJp5EgU0aOO6cbm7jTnGjOFq+KC540qtBMP6/bd/rcXNCb7t8ajb2eWC1PdvYCrpXM/IAZvdrOGhc9sU+8DeZj2dJg+WIlWkrnXsHMbAd5ePt1mCx6NPADFBRVSJg9MAN04yzBAcgmAeOgcZGYqHcihov3hIi0UnAwi0wtBHIquGcRKhUAG5clOY3YxyNJZ2HWglrCHtz1sHe7OGFBfGGR18C+S6CK7GDB2zm3IkJ16uqf/F3s+tF7u88mE3t6gr/Y/1U6OH99xm1Ta0Yqzie351qEXSwIDAQAB",

"iss": "gov.va.mhv.idp.v1",

"userType": "VETERAN"

},{

"rsaPublicKeyB64":

"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtxwSTqAC49xXxWIX/kpb4EWXjnpN8ii5FbjAOmubKA32ZVHC+OSj14uRkKJp5EgU0aOO6cbm7jTnGjOFq+KC540qtBMP6/bd/rcXNCb7t8ajb2eWC1PdvYCrpXM/IAZvdrOGhc9sU+8DeZj2dJg+WIlWkrnXsHMbAd5ePt1mCx6NPADFBRVSJg9MAN04yzBAcgmAeOgcZGYqHcihov3hIi0UnAwi0wtBHIquGcRKhUAG5clOY3YxyNJZ2HWglrCHtz1sHe7OGFBfGGR18C+S6CK7GDB2zm3IkJ16uqf/F3s+tF7u88mE3t6gr/Y/1U6OH99xm1Ta0Yqzie351qEXSwIDAQAB",

"iss": "external.idp.test1",

"userType": "VETERAN"

}

Also note that there is currently also a non-configurable requirement that all JWTs are signed with the RSA\_512 algorithm.

Once the JWT is validated and re-signed/issued, the service will respond differently depending on the HTTP Content-Type of the POST to reflect the expected usage:

* **External Users**: If the POST is to the /jwt\_session endpoint and is has a HTTP Content-Type of application/x-www-form-urlencoded, the user service assumes that this request is coming directly from a user/browser and will attempt to start a new authenticated user session. The body of the POST must include two parameters: user\_jwt and redirect\_url- the user\_jwt value is the signed [RFC7519](https://tools.ietf.org/search/rfc7519#section-3.1) / Base64-encoded JWT that includes the user's information, and the redirect\_uri value is the path of the NextGen app (e.g. /mbb/v1) the user should be redirected to once validation is completed.
* **Async Actions**: If the POST is to the /session endpoint and has a HTTP Content-Type of text/plain, the user service will assume the requesting service is attempting to retrieve a token to use on the user's behalf. The body of the POST should only be the signed, [RFC7519](https://tools.ietf.org/search/rfc7519#section-3.1) / Base64-encoded JWT from the external service.

For the External Users use case, external IDPs/systems should expect to send users to NextGen using a browser form POST/redirect - an example of a redirection:

<body onload='document.postRedirectForm.submit()'>

<div>

<div class="main-login">Login Success! Now redirecting…</div>

<form id="postRedirectForm" name="redirectForm" data-ajax="false" action="./users/v1/session" method="post">

<input type="hidden" name="user\_jwt" value="${user\_jwt}" />

<input type="hidden" name="redirect\_uri" value="${redirect\_uri}" />

</form>

</div>

</div>

This form can be set to post automatically using a javascript onload hook (e.g. body onload='document.postRedirectForm.submit()'). If any errors occur during the validation/redirection process, the user service will redirect the user back to the referring page (using the referrer header) with an URL query parameters of error\_code and error\_text (e.g. error\_code=403&error\_text=Error%20explaining%20why%20validation%20failed.)

Currently Supported Error Codes:

* **400**: JWT was incorrectly signed or did not include required parameters
* **403**: JWT was issued from an untrusted issuer
* **500**: An unexpected error occurred in the user service

# Appendix A – Development Steps / Tasks

| # | Description | Team | One-time / each environment |
| --- | --- | --- | --- |
|  | Implement JWT signing in eBenefits for Mobile apps – SEE other requirements in Section 5 | eBenefits team | One-time |
|  | Exchange public key with VAMF team | eBenefits team | Each environment |
|  | Submit requests to configure public key to trust eBenefits in VAMF | Mobile team | Each environment |
|  | Test | Both teams | Each environment |

# Appendix B – User Stories

The following User Stories were provided to MHV to inform their initial development and assessment of the overall effort involved in this integration.

## Phase 1. MHV User accesses VAMF App through MHV Portal

### MHV User accesses VAMF App via MHV Portal

**Story:** As a MHV user, I want to be able to use VAMF NextGen apps using my MHV account without having to log in again.

**Flow:**

1. MHV user accesses MHV portal
2. MHV user logs in with MHV or DS-Logon account
3. MHV user selects a VAMF app link (e.g. Mobile Blue Button (MBB))
4. MHV portal opens a new window/tab with the VAMF app application
   1. *(Optional)* If MHV user has not previously agreed to VAMF RoA, user is redirected to RoA to complete agreement
5. MHV user can access personal health information via VAMF app

**Assumptions**:

* VAMF has provided MHV a list of applications, icons, descriptions, and URLs for veteran VAMF apps
* *Technical:* MHV can issue a RSA-signed, *RS512* JWT-based token with MHV user information
* *Technical:* MHV and VAMF have pre-shared RSA public keys used to sign and verify tokens

**Acceptance Criteria:**

* MHV user can select a link to a VAMF app; app opens in a new window without additional logins or user intervention
* MHV user can access the full functionality of any veteran-facing VAMF NextGen app
* The MHV portal provides a JWT for the user including minimum set of identity attributes (claims):
  + *Name*: firstName, lastName, middleName
  + *Identifiers*: ICN, EDIPI, MHV ID
  + *Date of Birth*: DoB (yyyy-mm-dd)
  + *Gender*: Male, Female
  + *AuthMethod*: DSLogon, MHV
  + MHV Session ID
* The MHV provides an optional set of extended attributes (claim):
  + *Contact Info*: Email, PrimaryPhone, SMSPhone
* The VAMF application appears in a new tab/window; the primary window displaying the MHV portal remains open in the background
* If the MHV portal request is rejected, VAMF will redirect the user back to the referring MHV page with an error describing the problem
* *Technical*: The MHV portal instigates the user's browser to perform a HTTP POST to the VAMF/user/v1/jwt\_session endpoint with a MHV-signed JWT via a URL-encoded HTML form
  + Includes the JWT RFC7519 (Base 64-encoded) identity token (*vamf\_jwt*)
  + Includes the relative path to the VAMF destination app (*redirect\_url*)
  + Uses a HTTPContent-Type of *application/x-www-form-urlencoded*

### MHV User interacts with the VAMF application

**Story**: As a MHV user, I want my VAMF session activity to be synced with MHV so that my MHV session does not timeout due to inactivity while working within the VAMF app.

**Flow**:

1. Setup: Flow #1.1
2. MHV user interacts with the VAMF app and services
3. The user activity refreshes the VAMF session timeout
4. The VAMF activity is relayed to MHV in order to maintain MHV user's session

**Assumptions**:

* MHV has provided VAMF with a MHV session ID
* VAMF can identify a MHV user's activity through the API Gateway and make callbacks
* MHV does not need to maintain a VAMF session

**Acceptance Criteria**:

* MHV user actively working within a VAMF app will maintain an active MHV session
* *Technical*: MHV offers a 'session heartbeat' endpoint that renews a given a MHV Session ID
* *Technical*: VAMF user service invokes the MHV session heartbeat service with each unique request from a MHV user
  + VAMF user service performs a HTTP GET on the MHV URL */mhv-portal-web/session-heartbeat?MHV\_JESSIONID=[sessionID]*

### MHV User closes/navigates away from VAMF application

**Story**: As a MHV user, I want to be able to close/abandon a VAMF app (browser) window/tab and return to the MHV portal at any time.

**Flow**:

1. *Prerequisite*: Story #1.1
2. MHV user clicks the close button the VAMF application browser window/tab (but does not log out)
3. MHV returns to the existing MHV browser session

**Assumptions**:

* The MHV portal remains open in a background tab/window for the duration of the VAMF app use
* The VAMF session is not terminated immediately; it will timeout automatically
* No maintenance or cleanup of VAMF sessions is required by the MHV portal (e.g. activity within the MHV portal will not affect the session timeout for VAMF applications)

**Acceptance Criteria:**

* The MHV portal allows the user to resume their activity from the point where the VAMF link was selected
* The VAMF application can be reopened (e.g. *ctrl+shift+t*) within the inactivity timeout without requiring re-authentication through MHV

### MHV User selects "logout" from the VAMF application

**Story**: As a MHV user, I need the ability to end my session with the VAMF application so that I can resume my MHV session.

**Flow**:

1. Prerequisite: Story #1.1
2. MHV user clicks the logout button on a VAMF application
3. VAMF application redirects the user to the VAMF's /logout page
4. VAMF logout page (User Service) terminates the VAMF JWT session and attempts to close the current tab/window
5. The MHV user the MHV Portal session without interruption

**Assumptions**:

* The selection of 'logout' within a VAMF application does not imply the termination of the underlying MHV session

**Acceptance Criteria**:

* The MHV portal allows the user to resume their activity from the point where the VAMF link was previously selected
* The VAMF application cannot be reopened (e.g. *ctrl+shift+t*)- Attempting to do so will display a 401/Unauthorized message

## Phase 2. Mobile App uses MHV Credentials to Authenticate

### User directly navigates to a VAMF App

**Story**: As a MHV user, I want to be able to use VAMF NextGen apps using my MHV account by accessing the VAMF application directly (i.e. not through the MHV Portal).

**Flow**:

1. MHV user navigates to the VAMF launch pad (e.g. *veterans.mobilehealth.va.gov*)
2. MHV user is presented with a dashboard with links to VAMF veteran-facing application
3. MHV user selects an application (e.g. Mobile Blue Button)
4. VAMF public-facing landing page is accessed
   1. VAMF application does not support a public-facing information page and immediately redirects the MHV user to the /login page
   2. VAMF application does support a public-facing information page with a 'login' or 'authenticate' button that causes the redirection to /login
5. VAMF displays a *'Where Are You From?*' (WAYF) web page at /wayf/v1/ with links to supported VA identity profiles/services
6. MHV user selects "*My HealtheVet*" from the VAMF WAYF selection screen
   1. MHV user can check a “Remember This Decision” checkbox to have the selected provider be their default identity provider
7. VAMF application redirects the user to the MHV Portal's external login page (within the same tab/window)
8. The MHV Portal prompts the user to log in using either their MHV or DS-Logon account
9. User authenticates successfully and the MHV Portal redirects the user back to the VAMF /landing page
10. VAMF validates the user and forwards the user back to the initiating VAMF application/page

**Assumptions**:

* JWT Exchange in this flow is the same as with the MHV Portal » VAMF flow (Story #1.1)
* In this VAMF-initiated flow, the MHV application does not create a new browser tab/window

**Acceptance Criteria**:

* MHV user can freely navigate within the VAMF application and access all available features
* MHV user can move between VAMF applications without requiring re-authentication
* The same public/private RSA signing keys, user attributes, and error handling are used as with the MHV-initiated flow
* The MHV portal instigates the user's browser to perform a HTTP POST to the VAMF /user/v1/session endpoint with JWT via a URL-encoded HTML form that includes:
  + The JWT RFC 7519 (Base64) encoded identity token (vamf\_jwt) and
  + The VAMF relative path to the destination app (redirect\_url)
* The VAMF application appears in a new tab/window; the primary window displaying the MHV portal remains open in the background
* If the token from the MHV portal is rejected by VAMF for any reason, VAMF will redirect the user back to the MHV page with an error describing the problem

### User interacts with the VAMF application

**Story**: As a MHV user, I want my VAMF session activity to be synced with MHV so that my MHV session does not timeout due to inactivity while working within the VAMF app.

**Flow:**

1. Setup: Story #2.1
2. MHV user interacts with the VAMF app and services
3. The user activity refreshes the VAMF session timeout
4. The VAMF activity is relayed to MHV in order to maintain MHV user's session

**Assumptions**:

* MHV has provided VAMF with a MHV session ID
* VAMF can identify a MHV user's activity through the API Gateway and make callbacks
* MHV offers a 'session heartbeat' endpoint that renews a session timeout given a MHV Session ID
* MHV does not need to maintain a VAMF session

**Acceptance Criteria:**

* MHV user actively working within a VAMF app will maintain an active MHV session
* *Technical*: MHV offers a 'session heartbeat' endpoint that renews a given a MHV Session ID
* *Technical*: VAMF user service invokes the MHV session heartbeat service with each unique request from a MHV user
  + VAMF user service performs a HTTP GET on the MHV URL */mhv-portal-web/session-heartbeat?MHV\_JESSIONID=[sessionID]*

### User closes VAMF application

**Story**: As a MHV user, I want to be able to close a VAMF app (browser) window/tab without affecting .

**Flow:**

1. Prerequisite: Story #2.2
2. MHV user clicks the close button the VAMF application browser window/tab (but does not log out)

**Assumptions**:

* The abandoned VAMF and MHV sessions are not terminated immediately; both will timeout automatically due to inactivity
* No additional activity messages will be sent to the MHV session heartbeat service
* No maintenance or cleanup of abandoned MHV session is needed or expected by VAMF

**Acceptance Criteria**:

* Both the MHV Portal and VAMF app can be reopened (e.g. *ctrl+shift+t*) within the inactivity timeout

### User selects "logout" from the VAMF application

**Story**: As a MHV user, I need the ability to end my session with the VAMF application and have the additional option log out of MHV completely.

**Flow**:

1. Prerequisite: Story #2.2
2. MHV user clicks the logout button on a VAMF application
3. VAMF application redirects the user to the VAMF's /logout page
4. VAMF logout page (User Service) terminates the VAMF JWT session and redirects the MHV user back to the MHV Portal
5. MHV Portal shows the VAMF logout confirmation
6. MHV user can choose to also close the MHV session

**Assumptions:**

* The selection of 'logout' within a VAMF application does not imply the termination of the underlying MHV session – **NOTE – I think this decision was later changed during implementation.**

**Acceptance Criteria**:

* The VAMF application cannot be reopened (e.g. *ctrl+shift+t*)- Attempting to do so will display a 401/Unauthorized message
* The MHV user can choose to continue with the MHV logout if desired

## Phase 3. MHV Portal / App accessing VAMF Services (On Behalf of a User)

### MHV User accesses VAMF Service through MHV Portal / App

**Story**: As a MHV user, I want to be able use MHV Portal applications that can access and update VAMF services on my behalf

**Flow:**

1. MHV user accesses MHV portal
2. MHV user logs in with MHV or DSLogon account
3. MHV user accesses a MHV app with a VAMF service dependency (e.g. PGD)
4. MHV user makes changes in the MHV app that requires VAMF service updates
5. The MHV app generates a VAMF JWT session for the user
6. The MHV app uses the JWT session to make VAMF service updates
7. The VAMF service processes the delegated request, applying the same logic and access controls used under direct-user access scenarios
8. The MHV app displays the updated information to the MHV user

**Assumptions:**

* JWT Exchange uses the same JWT-based identity exchange and parameters as described in Story #1.1
* No direct interaction between the MHV user and the VAMF service is expected or implied
* The MHV app will generate a unique JWT/session for each individual veteran/user
* The MHV app can reuse the JWT for each call on behalf of the veteran/user
* The MHV app will need to periodically refresh/regenerate tokens/JWT to prevent expiration (15m)

**Acceptance Criteria:**

* The MHV app, acting on behalf of a MHV user, sends the user's MHV identity and attributes (described in Story #1) to VAMF user service
* Creates a MHV-signed RFC7519 (Base 64-encoded) JWT with the same claims/attributes described in Story #1
* Sends JWT via a HTTP POST to the VAMF /user/v1/session URL with a HTTP Content-Type of text/plain
* The VAMF user service validates the token and returns a RFC7519 (Base 64-encoded) VAMF-signed JWT using a Content-Type of text/plain
* The MHV app includes the VAMF-signed JWT as a X-VAMF-JWT HTTP Header on every call
* The VAMF service performs normal JWT validation and processes the request in accordance with its Swagger/OpenAPI contract
* The VAMF service operates without regard to the calling system/service (only considers the asserted user information & claims)
* The MHV portal invalidates any still-valid VAMF JWTs upon user logout by triggering a HTTP DELETE on the VAMF /users/v1/session URL