## Background: Current State of Vets.gov / EVSS Integration

Currently the Enterprise Veteran Self Service (EVSS) platform rest services are hosted behind an https reverse proxy (eauth.va.gov). HTTP web requests to these end points have all identity information appended into IAM-originated http request headers. IAM is the provider of these headers as documented in the IAM Portal Strategy design. All EVSS web services read these headers to establish the identity and coarse grain authorization for the user.

In the current integration between the Vets.gov Platform (Vets.gov) and EVSS, the objective was to temporarily reuse existing EVSS service capabilities with as a few changes as possible to either the code or transport layer security (TLS). To accomplish this reuse, the chosen approach was to integrate the two platforms without going through the reverse proxy. This choice was based on the architectural pattern for Vets.gov identity (using ID.me as an identity provider (IDP)) and that an integration between the IAM IDP infrastructure and ID.me was not present.

The current configuration between EVSS and Vets.gov is configured as follows: TLS is present using mutual authentication leveraging secure sockets layer (SSL) certificates issued from the VA certificate authority. Identity is accomplished by Vets.gov using claims based user attributes originating from ID.me (via SAML) to call MVI correlation services for VA identity attributes (example, EDIPI, participantId). Vets.gov then makes resource requests to EVSS adding the necessary information to the headers in the same format that the secure junction would have used. This means that existing EVSS services retrieve identity agnostic to the path of the request. Furthermore, in both cases these headers (and associated data) are only passed in server to server communication, and thus remain immutable to an end user.

## Intermediate State

A decision has been made to maintain the EVSS service layer to provide composite services for integration to Vets.gov. Given this, the objective is to establish a new trust-based security pattern that does not leverage, or mimic, the IAM portal strategy headers. The pattern for trust will be that Vets.gov will assert user identity to EVSS through a secure token. Vets.gov will create a secure token that will contain the identity information sourced from the results of an MVI correlation and the ID.me SAML token and transmit this with each resource request to EVSS. This trust relationship will leverage an RFC standards-based approach.

### One-Way Direct Trust Design

The token will use the JSON-based web token (JWT) which is an open standard (RFC-7519). A JWT token is a self-contained and compact mechanism for securely transmitting the identity claim between Vets.gov and EVSS. The token will be generated on each request by Vets.gov and will be sent in an http Authorization header (example: “Authorization: Bearer <token>”). Like the current implementation, this token is only passed server-to-server.

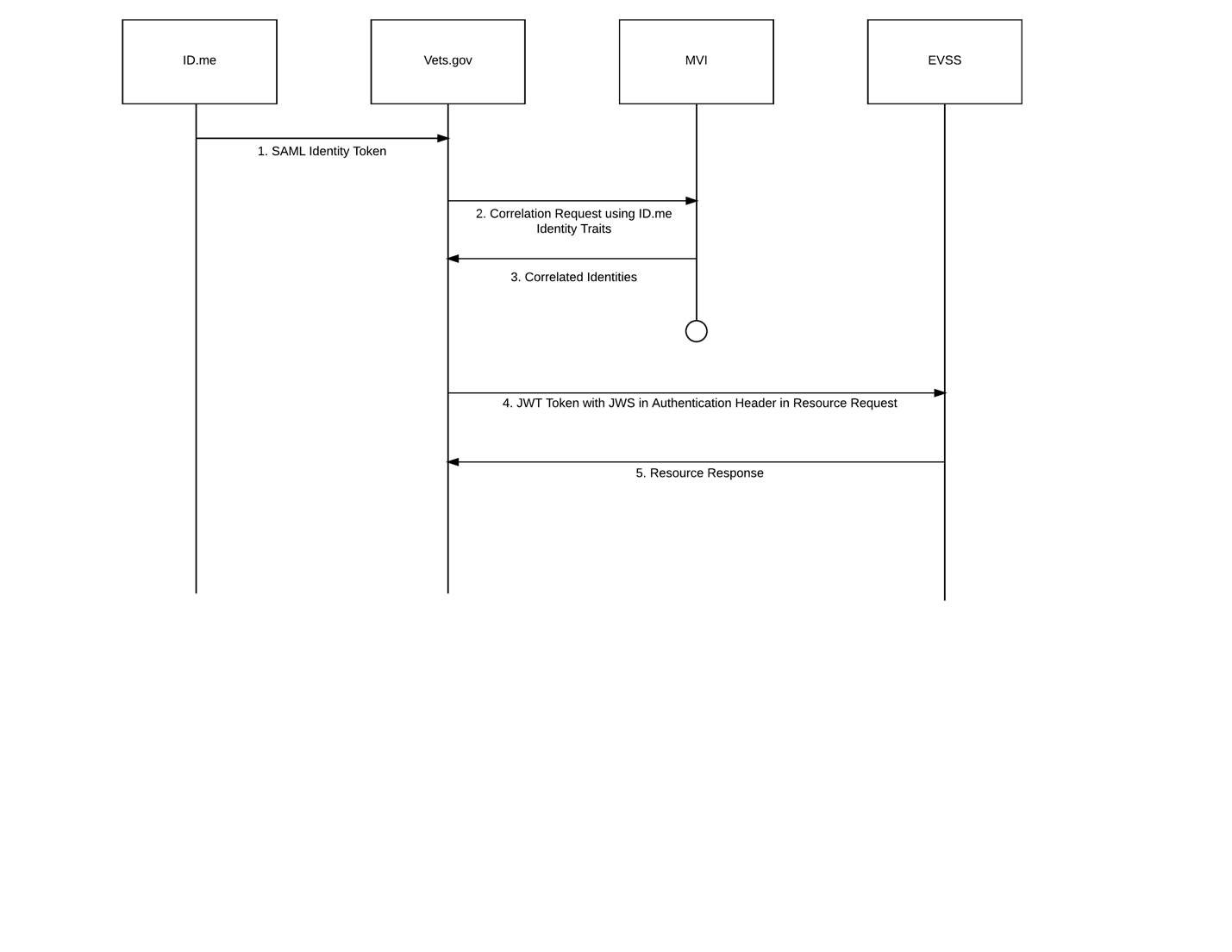
It is recommended that the JSON data structure also be signed by Vets.gov in compliance with JWS standards (RFC-7515) using a HMAC SHA256 algorithm. This will enable EVSS to verify that the sender was indeed Vets.gov and ensure that the message has not been tampered with and is valid.

Given that the JWT token must traverse the VA internal network using mutual authentication TLS and be signed, we are not recommending that the token itself also be encrypted. This will avoid the incremental time required on each request to encrypt and decrypt the token.

Why intermediate state versus final target state? In the proposed approach a new identity token is generated in one application and passed to another (Vets.gov->EVSS). Point being that the Vets.gov application is acting as an identity provider in the flow. Furthermore, EVSS does not have the ability to forward the token to downstream systems given they are not integrated with this approach for identity assertion.

### Logical Flow

As demonstrated in the diagram below, Vets.gov will accept the identity assertion provided by ID.me using SAML (1). It will use these attributes to obtain the Veteran’s VA identifiers by calling the Master Veteran Index (MVI) (2). Upon requesting a resource from EVSS, Vets.gov will generate a JWT token with JWS. EVSS will validate the signature of this token and leverage the identity information (3). Using the identity information, EVSS will respond to the resource request (4).



### JWT Token Attributes:

Attributes to be passed in token:

**Header**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Value | Description | Required/Optional |
| alg | HS256 | The "alg" (algorithm) Header Parameter identifies the cryptographic  algorithm used to secure the JWS. | Required |
| typ | JWT | Type- used by JWS applications to declare the media type [IANA.MediaTypes] of this complete JWS. | Required |

**Payload**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Value | Comments from Vets.gov | Required / Optional |
| **1305 REQUEST FOR MVI CALL** | | | |
|  |  | *Build a 1305 request object using First Name, Last Name, DOB, SSN Gender to call unattended MVI search. Depending on the traits one has available, a query can be constructed in a variety of ways. The 1305 transaction has the flexibility to support this variety through a paramaterList of traits.*  *HL7v3 1305 – Search Person Request Sample (Match criteria with person trait data)*  *<initialQuantity value="1" />*  *<parameterList>*  *<livingSubjectAdministrativeGender>*  *<value code="M" />*  *<semanticsText>LivingSubject.administrativeGender</semanticsText>*  *</livingSubjectAdministrativeGender>*  *<livingSubjectBirthTime>*  *<value value="19550512" />*  *<semanticsText>LivingSubject..birthTime</semanticsText>*  *</livingSubjectBirthTime>*  *<livingSubjectId>*  *<value root="2.16.840.1.113883.4.1" extension="111223333" />*  *</livingSubjectId>*  *<livingSubjectName>*  *<value use="L">*  *<given>Clark</given>*  *<given>Micheal</given>*  *<prefix>Dr.</prefix>*  *<suffix>IV</suffix>*  *<family>Kent</family>*  *</value>*  *<semanticsText>LivingSubject.name</semanticsText>*  *</livingSubjectName>*  *<patientAddress>*  *<value use="PHYS">*  *<streetAddressLine>Street1</streetAddressLine>*  *<streetAddressLine>Street2</streetAddressLine>*  *<streetAddressLine>Street3</streetAddressLine>*  *<streetAddressLine>Street4</streetAddressLine>*  *<city>Miami</city>*  *<state>FL</state>*  *<postalCode>33123</postalCode>*  *<country>USA</country>*  *</value>*  *<semanticsText />*  *</patientAddress>*  *</parameterList>* |  |
| **ID.me Proofed Attributes** | | | |
| firstName |  | Source: MVI  ***Schemas***: PRPA\_IN201305UV02.xsd (Request) and  PRPA\_IN201306UV02.xsd (Response)  ***Request***: Build a 1305 request object  ***Response***: Gets a 1306 response object  ***Transaction Traits***: MVI Search Person (Patient Registry Find Candidates Query – 1305/1306). Set the firstName using the following steps:  Step 1. Get the value of the Subject property.  (response1306.getValue (). getControlActProcess (). getSubject () returns “***List<PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1>***”)  Step 2. Iterate each subject to get patient person value of type  ***“PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01RegistrationEvent.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01Subject2.***  ***PRPA\_MT201310UV02Patient.***  ***PRPA\_MT201310UV02Person***”  Step 3. For each element of "***PRPA\_MT201310UV02Person.List<PN>***", read the value of Use property. If (Use Code == “L”) THEN  Set the First Name using “given” element value. First instance of this element is First Name.  **Name – Sample (MVI)**   |  | | --- | | <name use="L">  <given>First</given>  <given>Middle</given>  <family>Last</family>  </name> |   We have a proofed first name from ID.me SAML attributes (first\_name) |  |
| lastName |  | Source: MVI  ***Schemas***: PRPA\_IN201305UV02.xsd (Request) and  PRPA\_IN201306UV02.xsd (Response)  ***Request***: Build a 1305 request object  ***Response***: Gets a 1306 response object  ***Transaction Traits***: MVI Search Person (Patient Registry Find Candidates Query – 1305/1306). Set the lastName using the following steps:  Step 1. Get the value of the Subject property.  (response1306.getValue (). getControlActProcess (). getSubject () returns “***List<PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1>***”)  Step 2. Iterate each subject to get patient person value of type  ***“PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01RegistrationEvent.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01Subject2.***  ***PRPA\_MT201310UV02Patient.***  ***PRPA\_MT201310UV02Person***”  Step 3. For each element of "***PRPA\_MT201310UV02Person.List<PN>***", read the value of use property. If (Use Code == “L”) THEN  set the Last Name using “family” element value.  **Name – Sample (MVI)**   |  | | --- | | <name use="L">  <given>First</given>  <given>Middle</given>  <family>Last</family>  </name> |   We have a proofed last name from ID.me SAML attributes (last\_name) |  |
| middleName |  | Source: MVI  ***Schemas***: PRPA\_IN201305UV02.xsd (Request) and  PRPA\_IN201306UV02.xsd (Response)  ***Request***: Build a 1305 request object  ***Response***: Gets a 1306 response object  ***Transaction Traits***: MVI Search Person (Patient Registry Find Candidates Query – 1305/1306). Set the middleName using the following steps:  Step 1. Get the value of the Subject property.  (response1306.getValue (). getControlActProcess (). getSubject () returns “***List<PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1>***”)  Step 2. Iterate each subject to get patient person value of type  ***“PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01RegistrationEvent.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01Subject2.***  ***PRPA\_MT201310UV02Patient.***  ***PRPA\_MT201310UV02Person***”  Step 3. For each element of "***PRPA\_MT201310UV02Person.List<PN>***", read the value of use property. If (Use Code == “L”) THEN  set the Last Name using “family” element value. Second instance of this element is middleName.  **Name – Sample (MVI)**   |  | | --- | | <name use="L">  <given>First</given>  <given>Middle</given>  <family>Last</family>  </name> |   We have a proofed middle name from ID.me SAML attributes (middle\_name) |  |
| birthDate |  | Source: MVI  ***Schemas***: PRPA\_IN201305UV02.xsd (Request) and  PRPA\_IN201306UV02.xsd (Response)  ***Request***: Build a 1305 request object  ***Response***: Gets a 1306 response object  ***Transaction Traits***: MVI Search Person (Patient Registry Find Candidates Query – 1305/1306). Set the birthdate using the following steps:  Step 1. Get the value of the Subject property.  (response1306.getValue (). getControlActProcess (). getSubject () returns “***List<PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1>***”)  Step 2. Iterate each subject to get patient person value of type  ***“PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01RegistrationEvent.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01Subject2.***  ***PRPA\_MT201310UV02Patient.***  ***PRPA\_MT201310UV02Person***”  Step 3. For element "***PRPA\_MT201310UV02Person***", get the value of *birthTime* property and set the birthdate header in the format of “yyyymmdd”  **Date of Birth – Sample (MVI)**   |  | | --- | | <birthTime value="19821005" /> |   We have a proofed DOB from ID.me SAML attributes (birth\_date) |  |
| gender |  | Source: MVI  ***Schemas***: PRPA\_IN201305UV02.xsd (Request) and  PRPA\_IN201306UV02.xsd (Response)  ***Request***: Build a 1305 request object  ***Response***: Gets a 1306 response object  ***Transaction Traits***: MVI Search Person (Patient Registry Find Candidates Query – 1305/1306). Set the gender using the following steps:  Step 1. Get the value of the Subject property.  (response1306.getValue (). getControlActProcess (). getSubject () returns “***List<PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1>***”)  Step 2. Iterate each subject to get patient person value of type  ***“PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01RegistrationEvent.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01Subject2.***  ***PRPA\_MT201310UV02Patient.***  ***PRPA\_MT201310UV02Person***”  Step 3. For element "***PRPA\_MT201310UV02Person*** ", get the value of the administrativeGenderCode property and set the gender header.  **Gender – Sample (MVI)**   |  | | --- | | <administrativeGenderCode code="M" /> |   We have a proofed gender from ID.me SAML attributes (gender) |  |
| assuranceLevel | {Vets.gov: Source ID.Me} | Source: ID.me Attribute: <TBD>  The user's LOA at the time of authentication to the CS (Values of “1”, “2”, “3”, “4”) |  |
| email | Vets.gov Source: ID.me | Source: ID.me Attribute: <TBD>  Proofed email address. |  |
| iss | Vets.gov | Issuer - claim identifies the principal that issued the JWT.  *Example Header Value: Vets.gov* | Required |
| exp | {Vets.gov Compute} Current time + 5 minutes | Expiration time - claim identifies the expiration time on or after which the JWT MUST NOT be accepted for processing.  *Example Header Value: 1495640499*  *Value to be passed in seconds using UTC timezone* | Required |
| iat | {Vets.gov Compute} Current Time | Issued At - The "iat" (issued at) claim identifies the time at which the JWT was issued.  *Example Header Value: 1495640199*  *Value to be passed in seconds using UTC timezone* | Required |
| jti | {Vets.gov Compute} | JWT ID - The "jti" (JWT ID) claim provides a unique identifier for the JWT.  *Example Header Value*: 3056b3c0-35e4-46bc-9825-33df7e5a40ae  Unique and random identifier generated by the system (UUID) | Required |
| prefix |  | Source: MVI  ***Schemas***: PRPA\_IN201305UV02.xsd (Request) and  PRPA\_IN201306UV02.xsd (Response)  ***Request***: Build a 1305 request object  ***Response***: Gets a 1306 response object  ***Transaction Traits***: MVI Search Person (Patient Registry Find Candidates Query – 1305/1306). Set the prefix using the following steps:  Step 1. Get the value of the Subject property.  (response1306.getValue (). getControlActProcess (). getSubject () returns “***List<PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1>***”).  Step 2. Iterate each subject to get patient person value of type  ***“PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01RegistrationEvent.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01Subject2.***  ***PRPA\_MT201310UV02Patient.***  ***PRPA\_MT201310UV02Person***”  Step 3. For each element of "***PRPA\_MT201310UV02Person.List<PN>***", read the value of Use property. If (Use Code == “L”) THEN  Set the Prefix using “prefix” element value.  **Prefix – Sample (MVI)**   |  | | --- | | <name use="L">  <given>First</given>  <given>Middle</given>  <family>Last</family>  **<prefix>Dr.</prefix>**  <suffix>IV</suffix>  </name> | |  |
| suffix |  | Source: MVI  ***Schemas***: PRPA\_IN201305UV02.xsd (Request) and  PRPA\_IN201306UV02.xsd (Response)  ***Request***: Build a 1305 request object  ***Response***: Gets a 1306 response object  ***Transaction Traits***: MVI Search Person (Patient Registry Find Candidates Query – 1305/1306). Set the suffix using the following steps:  Step 1. Get the value of the Subject property.  (response1306.getValue (). getControlActProcess (). getSubject () returns “***List<PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1>***”).  Step 2. Iterate each subject to get patient person value of type  ***“PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01RegistrationEvent.***  ***PRPA\_IN201306UV02MFMI\_MT700711UV01Subject2.***  ***PRPA\_MT201310UV02Patient.***  ***PRPA\_MT201310UV02Person***”  Step 3. For each element of "***PRPA\_MT201310UV02Person.List<PN>***", read the value of Use property. If (Use Code == “L”) THEN  Set the Prefix using “suffix” element value.  **Suffix – Sample (MVI)**   |  | | --- | | <name use="L">  <given>First</given>  <given>Middle</given>  <family>Last</family>  <prefix>Dr.</prefix>  **<suffix>IV</suffix>**  </name> | |  |
| correlationIds |  | Source: MVI  **Schemas**: PRPA\_IN201305UV02.xsd (Request) and  PRPA\_IN201306UV02.xsd (Response)  **Request**: Build a 1305 request object  **Response**: Gets a 1306 response object  **Transaction Traits**: MVI Search Person (Patient Registry Find Candidates Query – 1305/1306).  Set the **correlationIds** using the following rules:  **Step 1**. Get the value of the Subject property.  (response1306.getValue (). getControlActProcess (). getSubject () returns “**List<PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1>**”)  **Step 2**. For each Subject element get the patient id list  "**PRPA\_IN201306UV02MFMI\_MT700711UV01Subject1.**  **PRPA\_IN201306UV02MFMI\_MT700711UV01RegistrationEvent.**  **PRPA\_IN201306UV02MFMI\_MT700711UV01Subject2.**  **PRPA\_MT201310UV02Patient**.  List<II>"  Step 3. For each patient id element "II", get the value of the extension property & set it.  *Example Header Value as String Array (String [])*  *{*  *"correlationIds": ["1020212383V608753^NI^200M^USVHA^P", "100001412^PI^553^USVHA^A","20120203003^PI^500^USVHA^A","1025062341^NI^200DOD^USDOD^A","123456789^SS"]*  *}* |  |
|  |  | Technical Analysis for header “**va\_eauth\_authorization**”  JSON header, the value of which is stored in PersonDetail complex type object. JSON data specific to the CSP.  There are multiple properties in JSON that defines a person details. The fields are “id”, “idType”, “edi”, “firstName”, “middleName”, “lastName”, “cadencyName”, “gender“, “birthDate”, “deceased”, “deathDate”, “status”, “associationReason”, “personnel [”category”, “organization”, “rank”, “serviceBranchClassification”, “entitlementCondition”]”, “benefit”  Codebase occurrences of ***VaafiTraits.getPersonDetail()*** method are listed below: ***1) vii-app/src/main/java/gov/va/vii/app/service/personalinfo/PersonalInfoServiceImpl.java (VAPII)*** ***LOGIC****: If the Person Detail is available from VAAFI, then sets the personal information using the data fields, else sets it from a call to UserProfileService. getBeneficiaryProfile().getBeneficiary() which has attributes set from VAAFITraits. Personal information set is “First Name, Middle Name, Last Name, SSN, DOB and Gender“*  ***USAGE****: Personal information to be displayed on the front end*  *NO NEW HEADERS NEEDED AS FN, LN, SSN, DOB and GENDER ARE AVAILABLE IN SEPARATE HEADERS FROM VETS.GOV* ***2) wss-framework/src/main/java/gov/va/wss/framework/security/DependentAuthorityProvider.java (FRAMEWORK)*** ***LOGIC****: To populate DEPENDENT role based on the association reason mapping. Essentially used to fill granted authorities in VaafiTraits object.*  *The following roles expression values of the “Association Reason” are mapped to a role named DEPENDENT*  *SPOUSE, CHILD, FOSTER\_CHILD,PARENT,PARENT\_IN\_LAW, STEPCHILD,IN\_LOCO\_PARENTIS,EMERGENCY\_CONTACT,WARD,FORMER\_SPOUSE,JOINT\_MARRIAGE\_SPOUSE,OTHER\_HEALTH\_INSURANCE\_SUBSCRIBER,PRE\_ADOPTIVE\_CHILD,MEMBER\_OF\_HOUSEHOLD,UNKNOWN*  ***USAGE of ROLES (ENUM WSSRole.\*) across the EVSS codebase****:*     * *PCIU Service (PciuServiceHelperBean.java in common services) to get* ***EDIPI*** *for the Veteran vs Representative(POA) and load the profile information.* * *Audit Utility (AuditUtil.java in common services) to get requestor information based on Veteran vs Representative(POA)* * *Audit Utility (AuditUtil.java in common services) to get requestor information based on Veteran vs Representative(POA)* * *Forwarding Address Resource (ForwardingAddressResource.java in common services web) to get* ***EDIPI*** *for the Veteran vs Representative(POA) to eventually get the Participant ID. ID is used in* ***PCIU*** *methods for persisting and retrieving forwarding address and forwarding address effective date into the Corporate database.* * *Health Care Resource (HealthCareResource.java in common services web) to get* ***EDIPI*** *for the Veteran vs Representative(POA). It is used in* ***PCIU*** *methods for persisting and retrieving forwarding address and forwarding address effective date into the Corporate database.* * *VA Profile Resource (VAProfileResource.java in common services web) to get* ***EDIPI*** *for the Veteran vs Representative(POA) to eventually get the Participant ID. The calls are made in Document services to get person profile.* * *SOJ Resource (SOJResource.java in common services web) to get* ***EDIPI*** *for the Veteran vs Representative(POA) to eventually get the Participant ID.* * *MVI Resource (MviResource.java in common services web) to get* ***EDIPI*** *for the Veteran vs Representative(POA) to retrieve person based on EDIPI.* * *Resource Helper (ResourceHelperBean.java in common services web) to get* ***EDIPI*** *for the Veteran vs Representative(POA) to eventually get the Participant ID.* * *My Dashboard displays “My Documents” link only if the user isn’t a dependent and that there are no 7332 restrictions (SEP System)* * *If a VETERAN, then chat URL is constructed with a specific entry point id and template* * *In the VRE controller a check is made if the user is dependent. Dependents cannot apply 'electronically' so a flag is set for the view to handle it.*   *BASED ON THE USAGE SERVICES USE ROLES (WSSROLE.\*) TO GET EDIPI AND PID INFORMATION. NEW HEADER WON’T BE NEEDED.*  *ON THE FRONT END, ROLES ARE USED TO SHOW / HIDE MY DOCUMENTS LINK, CHAT URL AND VRE APPLICATION APPLY CHECK*  *NEED TO UNDERSTAND TYPES OF USER ON VETS.GOV ALLOWED TO MAKE EVSS SERVICE CALLS. IF NOT A VETERAN, THEN WE MAY NEED A NEW HEADER TO IDENTIFY USER ROLE AS “VETERAN”, “DEPENDENT” ETC* **3) wss-framework/src/main/java/gov/va/wss/framework/security/VaafiTraits.java** ***LOGIC****: Returns a Person Detail complex type object created from header “va\_eauth\_authorization” value which is a JSON object.*  ***USAGE****: Class that returns the “PersonDetail” Object*  *NO NEW HEADER NEEDED* **4) wss-framework-web-security/src/main/java/gov/va/wss/framework/web/security/VaafiTraitsUtils.java** ***LOGIC****: Validates EDIPI for the user and sponsor, if the Person Detail and Sponsor (head of family) complex type objects are populated.*  *NEED TO UNDERSTAND TYPES OF USER ON VETS.GOV ALLOWED TO MAKE EVSS SERVICE CALLS. SPONSOR VS USER INFORMATION* **5) wss-common-services/src/main/java/gov/va/wss/common/services/service/lettergenerator/html/processor/AbstractBenefitSummaryLetterGeneratorHelper.java** ***LOGIC****: To populate data for attributes “headOfFamily”, “fileNumber” in the model map to be used in the front end.*  ***USAGE****: Data from these attributes are read in the letter generation FreeMarker Template Language (FTL) files.*  *“headOfFamily” is used to display First Name and Last Name of Head of family in the following letter on front end:*   * *Benefit Summary Letter Non-Veteran*   *“fileNumber” is used to display Claim Number in the following letters on front end:*   * *Benefit Summary Letter Non-Veteran* * *Benefit Summary Letter Veteran*   <#if fileNumber?length == 8>  <#assign claimNumber = "${fileNumber?substring(0,2)?html}-${fileNumber?substring(2,5)?html}-  ${fileNumber?substring(5,8)?html}">  <#elseif fileNumber?length == 9>  <#assign claimNumber = "xxx-xx-${fileNumber?substring(5,9)?html}">  </#if*>*  *USAGE IS ON THE FRONT END DURING LETTER GENERATION* **6) wss-common-services/src/main/java/gov/va/wss/common/services/service/lettergenerator/html/processor/LetterGeneratorDataModelBuilderImpl.java** ***LOGIC****: . If the Person Detail “Association Reason” value is SPOUSE then get the veteran information by file number response. The response is used to set the BIRLS and CORP records.*  ***USAGE****: The logic is executed for multiple letter generation if it's a dependent letter. To get the dependent's sponsor BIRLS and CORP information.*  *LIKELY A NEW HEADER NEEDED TO IDENTIFY “ASSOCIATION REASON” FOR THE USER* **7) wss-common-services/src/main/java/gov/va/wss/common/services/service/lettergenerator/LetterGeneratorServiceImpl.java** ***LOGIC****: If the Person Detail “Association Reason” value is SPOUSE then get the veteran information by file number response. The response is used to set the BIRLS and CORP records.*  ***USAGE****: The logic is executed during /GetLetterList service call to get the list of letters for a user.*  *LIKELY A NEW HEADER NEEDED TO IDENTIFY “ASSOCIATION REASON” FOR THE USER* **8) wss-common-services/src/main/java/gov/va/wss/common/services/service/lettergenerator/util/LetterGeneratorImplHelper.java** ***LOGIC****: Returns a Person Detail complex type object created from header “va\_eauth\_authorization” value which is a JSON object.*  ***USAGE****: Object that returns the “PersonDetail”* **9) wss-common-services/src/main/java/gov/va/wss/common/services/service/lettergenerator/util/PartnerServiceFactory.java** ***LOGIC****: Finds the Person Status from Person Detail object. Based on person status, sets the File Number to find the rating information*  ***USAGE****: To set the file number on the rating service (FindRatingInfoService) request object. Based on the status to be either DEPENDENT, OTHERS OR VETERAN, the file number is set accordingly.*  *NEED TO VERIFY IF THE HEADER VALUE FOR “****birlsfilenumber****” COULD BE SAFELY USED FOR THE SERVICE.*  *Possible values for “personStatus”: “VETERAN”,”DEPENDENT,”DECEASED”, “SPONSOR”, “UNKNOWN”* **10) wss-common-services/src/main/java/gov/va/wss/common/services/service/userprofile/UserProfileServiceImpl.java** ***LOGIC****: To find Veteran ID / File Number based on the Person Status and Detail object.*  ***USAGE****: The call to “findVeteranID” is made from Letter Generation Partner Service, Rating Service, Claimant General Service to set file number* **11) wss-common-services-api/src/main/java/gov/va/wss/common/services/api/serviceproperties/transfer/ServiceProperties.java** ***LOGIC****: Class that sets the Beneficiary and Sponsor information from VAAFITraits Person Detail and Head of Family Person Objects .*  ***USAGE****: The call to ”getServiceProperties()” is made from multiple places as mentioned below*  *- Letter Generation to set the letter service properties*  *- My Dashboard Service*  *- Request State Benefits Info*  *- PCIU Audit for Update*    *This is typically done to set the personal information such as First Name, Last Name, DOB* **12) wss-form526-services/src/test/java/gov/va/wss/form526/services/service/load/VeteranPersonalInfoMapperTestHelper.java** ***LOGIC****: Test Class that sets the DOB on Person Detail Object*  <pre>  <complexType name="person">  <complexContent>  <restriction base="{http://www.w3.org/2001/XMLSchema}anyType">  <element name="cadencyName" type="{http://www.w3.org/2001/XMLSchema}string" minOccurs="0"/>  <element name="deceased" type="{http://www.w3.org/2001/XMLSchema}boolean" minOccurs="0"/>  <element name="deathDate" type="{http://www.w3.org/2001/XMLSchema}dateTime" minOccurs="0"/>  <element name="status" type="{http://www.dmdc.osd.mil/identitymanagement/authorization}personStatus" minOccurs="0"/>  <element name="associationReason" type="{http://www.dmdc.osd.mil/identitymanagement/authorization}associationReason" minOccurs="0"/>  <element name="personnel" type="{http://www.dmdc.osd.mil/identitymanagement/authorization}personnel" maxOccurs="unbounded" minOccurs="0"/>  <element name="benefit" type="{http://www.w3.org/2001/XMLSchema}string" maxOccurs="unbounded" minOccurs="0"/>  The following is a test user example for a DSLogon non-surrogate user:  {"authorizationResponse":{"id":796220828,"idType":"SSN","edi":1045848716,"firstName":"ARTHUR","middleName":"E","lastName":"ROSE","gender":"MALE","birthDate":"1954-05-  26T00:00:00-07:00","deceased":false,"status":"SPONSOR","personnel":[{"category":"CIVILIAN\_RETIREE","entitlementCondition":"00","organization":62,"serviceBranchClassification":"F"},{"category":"GREY\_AREA\_RETIREE","entitlementCondition":"00","organization":42,"rank":"CMSGT","reservistIndicator":"V2","serviceBranchClassification":"F"}],"benefit":42,"headOfFamily":""}}  Julie: Please define "complete personal details" … |  |

More information on JWT from oAuth: <https://auth0.com/learn/json-web-tokens/>

*VAAFI Headers and JWT Claims Mapping Table*

|  |  |  |
| --- | --- | --- |
| **VAAFI Header Name** | **JWT Claim** | **Comments** |
| va\_eauth\_csid |  | Not required. *Authenticating CSP* |
| va\_eauth\_issueinstant |  | Not required. *Time of authentication* |
| iv-user |  | Not required. *This field to differentiate users that are from different CSPs like the va\_eauth\_csid, and can also differentiate between users that log in with either a PIV or CAC to the PKI CSP* |
| va\_eauth\_commonname |  | Not required. *Common Name* |
| va\_eauth\_emailAddress |  | Not required. *Email Address, already in JWT claim* |
| va\_eauth\_hash |  | Not required. *VAAFI's recommended value for a unique user identifier for applications to use.* |
| va\_eauth\_authenticationauthority |  | Not required. *Authentication Authority* |
| va\_eauth\_authenticationmethod |  | Not required. *Authentication Method* |
| va\_eauth\_uid |  | Not required. *Unique User Identifier* |
| PD-S-SESSION-ID |  | Not required. *multi-valued field that contains application cookies as well as the VAAFI cookie called PD-S-SESSION-ID* |
| va\_eauth\_birthdate\_v1 |  | Not required. *Birth Date, already in JWT claim* |
| va\_eauth\_secid |  | Not required. *Security Identifier from IAM Provisioning Service.* |
| va\_eauth\_sponsordodedipnid |  | Not required. This header is specific to the DSLogon CSP only |
| va\_eauth\_csponly |  | Not required. Used for only CSP Data |
| va\_eauth\_backenddown |  | Not required. Boolean provides Portal Strategy data due to an IAM service being down, false otherwise. |
| va\_eauth\_authorization |  | Not required.  Header values below are passed or extracted from JWT claims and correlationIds  authorizationResponse: {  status: 'VETERAN',  idType: 'SSN',  id: @user.ssn,  edi: @user.edipi,  firstName: @user.first\_name,  lastName: @user.last\_name,  birthDate: iso8601\_birth\_date  } |
| va\_eauth\_assurancelevel | assuranceLevel | User Assurance Level |
| va\_eauth\_email | email | Email address of user |
| va\_eauth\_firstName | firstName | User First Name |
| va\_eauth\_lastName | lastName | User Last Name |
| va\_eauth\_middleName | middleName | User Middle Name |
| va\_eauth\_birthdate | birthDate | User DOB |
| va\_eauth\_gender | gender | User Gender |
| va\_eauth\_prefix | prefix | User Name Prefix |
| va\_eauth\_suffix | suffix | User Name Suffix |
| va\_eauth\_pid |  | Value will be obtained from **correlationIds** claim |
| va\_eauth\_icn |  | Value will be obtained from **correlationIds** claim |
| va\_eauth\_birlsfilenumber |  | Value will be obtained from **correlationIds** claim  **birlsfilenumber used to identify a record in BIRLS** |
| va\_eauth\_pnid |  | Value will be obtained from **correlationIds** claim |
| va\_eauth\_pnidtype |  | Value will be obtained from **correlationIds** claim |
| va\_eauth\_dodedipnid |  | Value will be obtained from **correlationIds** claim **EDIPI for User** |
|  | alg | The "alg" (algorithm) Header Parameter identifies the cryptographic algorithm used to secure the JWS. |
|  | typ | Type- used by JWS applications to declare the media type [IANA.MediaTypes] of this complete JWS. |
|  | correlationIds | New claim for the correlation ids retrieved from MVI |
|  | iss | Issuer - claim identifies the principal that issued the JWT. |
|  | exp | Expiration time - claim identifies the expiration time on or after which the JWT MUST NOT be accepted for processing |
|  | iat | Issued At - The "iat" (issued at) claim identifies the time at which the JWT was issued. |
|  | jti | JWT ID - The "jti" (JWT ID) claim provides a unique identifier for the JWT. |

*Important notes for the claims and token:*

1. For the claim "**birthDate**", sample value would be "**1978-05-20**". On the EVSS side, we expect value for this claim to be a valid date format: “yyyy-MM-dd”.
2. JWT ID - The "**jti**" claim provides a unique identifier for the JWT. Unique and random identifier generated by the system (UUID). The identifier would be used to track down instrumentation data in the services.
3. Issued At - The "**iat**" (issued at) claim identifies the time at which the JWT was issued. Value to be passed in seconds using valid date with UTC time zone, UTC time zone.
4. Expiration time -  The “**exp**” claim identifies the expiration time on or after which the JWT MUST NOT be accepted for processing. Value to be passed in seconds using valid date with UTC time zone. Set with expiration time to be {Current time + 5 minutes}. We could discuss in the next meeting if there is a need to extend the window to 15 minutes.
5. JWT token needs to be signed in compliance with JWS standards (RFC-7515) using a HMAC SHA256 algorithm using a BASE64-encoded algorithm-specific signing key. EVSS would require this key to parse JWT token on our side.
6. In the Payload, the REQUIRED claims are at the least "iss", "iat", "jti", "exp". However, we expect all the claims to be populated in the token and passed to us. The intent is to avoid making any MVI calls for the missing information in the payload. EVSS services may end up using any of the claims information provided in the token.
7. For the legacy EVSS, no changes are suggested, continue to provide a VAAFI header. However, for the services that are planned to be hosted on GovCloud, they would need JWT token minus VAAFI header.
8. To allow a non-Veteran in the future, we will likely need additional claims to be passed in the JWT token. **Pending analysis, TBD**
9. Vets.gov must continue to send VAAFI headers for requests to legacy service applications. For any new service requests deployed on GovCloud, JWT token must be passed in Authorization header. We don’t recommend sending both the VAAFI and JWT headers in the requests as it may potentially impact application behavior.
10. *Patrick Vinograd [1:01 PM]*

The doc looks good and the table is helpful to me in terms of ensuring all attributes are covered. I do have one question about the intended duration of a given JWT token. I haven't used them extensively but frequently they are intended as a session token to be used across multiple requests.

I think many of vets.gov -> EVSS interactions are going to be single requests, but there may be some common patterns of correlated series of requests (get a list of claims, then drill down to get individual claims). But, given the short expiry of the JWT tokens proposed in the doc, we are thinking it is fine if vets.gov generates a new JWT token for each resource request. This line up with the document which states:

> Upon requesting a resource from EVSS, Vets.gov will \*generate\* a JWT token with JWS

I just wanted to see if there were any concerns from the EVSS side about us essentially remaining stateless and making a new token for each request, or if that's what you had in mind as well.

Abhijit Kulkarni [1:20 PM]

@**patrick** - There are no concerns generating a new token for every request. That is the proposed option which would ensure that EVSS services doesn't receive the same token for subsequent requests.

**SAMPLE JWT TOKEN 1**

{

"assuranceLevel": "2",

"birthDate": "1978-05-20",

"correlationIds": [

"77779102^NI^200M^USVHA^P","912444689^PI^200BRLS^USVBA^A","6666345^PI^200CORP^USVBA^A","1105051936^NI^200DOD^USDOD^A","912444689^SS"

],

"email": "jane.doe@va.gov",

"firstName": "JANE",

"gender": "FEMALE",

"lastName": "DOE",

"middleName": "M",

"prefix": "Ms",

"suffix": "S"

}

*DECODED TOKEN VALUE*

Header

{

typ: "JWT",

alg: "HS256"

}

Payload

{

"iss": "Vets.gov",

"iat": 1513364588,

"jti": "d3cf8355-7263-4c86-b413-1f476f54253b",

"exp": 1513368203,

"firstName": "JANE",

"middleName": "M",

"lastName": "DOE",

"prefix": "Ms",

"suffix": "S",

"birthDate": 264470400000,

"gender": "FEMALE",

"assuranceLevel": "2",

"email": "jane.doe@va.gov",

"correlationIds": [

"77779102^NI^200M^USVHA^P",

"912444689^PI^200BRLS^USVBA^A",

"6666345^PI^200CORP^USVBA^A",

"1105051936^NI^200DOD^USDOD^A",

"912444689^SS"

]

}

**SAMPLE JWT TOKEN 2**

{

"assuranceLevel": "2",

"birthDate": "1955-01-01",

"correlationIds": [

"77779102^NI^200M^USVHA^P","912444689^PI^200BRLS^USVBA^A","1105051936^NI^200DOD^USDOD^A","912444689^SS"

],

"email": "john.wayne@va.gov",

"firstName": "JOHN",

"gender": "MALE",

"lastName": "WAYNE",

"middleName": "M",

"prefix": "Mr",

"suffix": ""

}

*DECODED TOKEN VALUE*

Header

{

typ: "JWT",

alg: "HS256"

}

Payload

{

"iss": "Vets.gov",

"iat": 1513365960,

"jti": "ccb13e06-0926-4d77-805e-bd5fcfdddf53",

"exp": 1513366860,

"firstName": "JOHN",

"middleName": "M",

"lastName": "WAYNE",

"prefix": "Mr",

"suffix": "",

"birthDate": -473385600000,

"gender": "MALE",

"assuranceLevel": "2",

"email": "john.wayne@va.gov",

"correlationIds": [

"77779102^NI^200M^USVHA^P",

"912444689^PI^200BRLS^USVBA^A",

"1105051936^NI^200DOD^USDOD^A",

"912444689^SS"

]

}