# Overview

This doc is something of a brain-dump of various thoughts, questions and concerns about the EVSS -> vets.gov migration.

# Role of EVSS team

At the offsite, we had general agreement that the EVSS team should provide the API service layer for **most**/**all** benefits functionality. Vets.gov will remain a (thin) UI/UX layer on top of this. This is a departure from how we first started thinking about this.

#### Question

**Do we have broad agreement on this, and does everybody understand the implications (contracting, funding, etc.)?**

This is fundamentally incompatible with “EVSS might go away in the fall.” We need all stakeholders to understand and commit to “EVSS team supports vets.gov for a long time.”

#### History/Options

When we first started looking at moving eBenefits features to vets.gov, we were considering not using EVSS at all. As I dug into the complexity in EVSS, it started to make less sense that vets.gov could take on all of this logic and back-end functionality. EVSS contains a **huge** amount of business logic and orchestration logic to support the features currently on eBenefits.

###### Option 1: Throw away EVSS

As I dug into this, I reached the following conclusions:

* EVSS contains a huge amount of very hard-won business logic and orchestration logic – why throw all this away?
  1. If we re-wrote all this ourselves, we would likely end up with something equally complicated and with a whole different/new set of bugs.
* Scope would be ridiculously large to re-write all this functionality ourselves. Vets.gov team is nowhere near staffed to do this.
* Vets.gov should strictly remain a front-end UI/UX layer for VA services. This separation is important.

###### Option 2: Create/use APIs for some, but not all features

The thinking here is that some features (like 526 disability compensation) are extremely complicated so let’s leverage an API from EVSS for complex features like this, but consider going directly to the data sources (e.g. BGS) for simpler features like Payment History.

This was the direction we were thinking made the most sense, but as I dug into this, there really aren’t that many features that are actually “simple”. Things like VA Form Letters have been suggested as features that we could do ourselves, but even these are not at all trivial when you break down all the business requirements.

###### Option 3: Use EVSS API for all benefits functionality

This was the general agreement from offsite meeting.

Keeps separation clear – EVSS API becomes service layer for **most**/**all** benefits functionality on vets.gov. Vets.gov remains a thin UX/UI layer on top of this. This is similar to how we use the MHV API with vets.gov.

# UX/UI for eBenefits Features on Vets.gov

The disability compensation application currently on eBenefits is **significantly** more complex than any of the existing vets.gov forms. A typical application contains not just the 21-526 form, but many sub-forms (21-686c, 21-0781, etc.) depending on the benefits being claimed. EBenefits presents the disability compensation flow as “TurboTax”-style series of questions and largely abstracts away the concept of a specific form. The user answers a number of questions and fills out fields, and the back-end generates whatever necessary forms are required for the claim.

Other than general agreement that we will likely want do something similar, we have very little definition of what this feature will look like on vets.gov.

#### Concerns/Questions

* While it is possible to define and agree on an EVSS API for this functionality in isolation, it would be much better to have a more concrete idea of the UX that we want as we’re defining the API. **Ideally we can start on UX discovery for these features ASAP.**
* Disability compensation on eBenefits has a very complicated UX with a lot of conditional flows and UX for a lot of complex business rules. It’s not clear that much of this complexity can be eliminated – the UX for the vets.gov version of this feature will be significantly more complicated than any of the current vets.gov features. The scope for the UX/UI effort alone here is **very** large.

# Burials/Pensions Back-end

The back end for these 2 VONAPP forms may need to connect with VBMS and BGS and possibly require document upload services. EVSS doesn’t currently do these forms so it makes sense that we would need to build a backend for this, but…

#### Question

**A lot of what the EVSS team is doing is providing APIs for their systems that deal with VBMS, BGS, and document services. Should we coordinate this with them?**

# In-Flight Form Storage

This is something that we need for other forms (e.g. pension and burial), so it makes sense to define a general strategy for all of vets.gov. But this could be a much more complicated endeavor for something like the 526 disability compensation application that consists of many different pages, conditional sections, etc. This is another area where it would be best to have a good sense of the general UX for the 526 application as we’re designing the in-flight storage system.

#### Concern

**We need a good understanding of the UX requirements for the whole 526 flow before designing the general strategy for vets.gov in-flight form storage. Need to make sure we don’t design something that ends up being inadequate for 526.**

# Authorization and Trust Boundaries

We briefly talked about this at the offsite. SAML was mentioned, **but this needs a lot more thought and definition**. Assuming that we don’t want to use the “fake VAAFI” approach we’re using for Claim Status. (The current 526 API implementation still uses VAAFI)

Josh L. seems to have a good grasp on all this so I’d recommend leaning on him to help define a strategy here.

# API Structure

Lots of questions here that we will need to work with the EVSS team to define:

* Current 526 API is a good start, but discussion was to broaden this into a more general “disability” API that encapsulates the data for the various sub-forms, etc. This is likely much more complicated than that currently-defined “526 only” API. Are there cases where sub-forms need to be submitted in isolation? If so, how would this work?
* Current 526 API consists of major “Load”, “Validate” and “Submit” API calls. Are there cases where a user needs to “update” a claim? How would this work? Lots of open questions here that are increasingly hard to answer with no UX definition.

# AWS GovCloud Integration

General idea is for EVSS team to package up API components into microservices that can be hosted in AWS GovCloud instead of the AITC environment where it is now. One of the biggest advantages of this approach is that the EVSS team would be able to iterate and deploy things **much** faster if they don’t have to deal with the AITC deployment process.

#### Concern

**There are huge number of open questions here that I suspect will take a long time to answer.**

ATO questions, tunnel/VPN questions, etc… We should try hard to not have this block any progress on the rest of the migration. Ideally we can decouple this as much as possible such that any eventual transition is invisible (or a simple config change) to the vets.gov stack.

# Profile / Pre-Fill

Current implementation of EVSS 526 API has a **Load** function that will return a consolidated set of pre-fill data from a number of different data sources.

#### Question

**How does this mesh with our more general form pre-filling efforts? What’s our long-term strategy here? Do we even want to use this data from EVSS? Does this overlap with the “person” API that we discussed at the offsite?**