526 Claim for Increase Flow Integration with PMR (4142/4142a)

(DRAFT)

Table of Contents

[Overview 3](#_Toc526256051)

[526 / 4142 Technical Design 3](#_Toc526256052)

[Vets UI 3](#_Toc526256053)

[Vets API 4](#_Toc526256054)

[Outstanding Items 4](#_Toc526256055)

[Design Considerations 4](#_Toc526256056)

[Providers per condition 4](#_Toc526256057)

[Form Builder 4](#_Toc526256058)

[Option 1 5](#_Toc526256059)

[Option 2 5](#_Toc526256060)

[Design Decision 5](#_Toc526256061)

[526/4142 Integration 5](#_Toc526256062)

[526/4142 Flow 5](#_Toc526256063)

[Flow Questions / Comments 7](#_Toc526256064)

[External Processing of PMR data 7](#_Toc526256065)

[Overview 7](#_Toc526256066)

[PMR Data Processing 7](#_Toc526256067)

[4142 Back End Processing 8](#_Toc526256068)

[JSON Implementation Details 10](#_Toc526256069)

[Method signature for 4142 sidekiq background job 14](#_Toc526256070)

[PDF Generation 16](#_Toc526256071)

[ICMHS Flow 17](#_Toc526256072)

[Design Decisions 17](#_Toc526256073)

[Design Points 17](#_Toc526256074)

[Requirements Refinements / Questions 18](#_Toc526256075)

[526 / 4142 Submission Process 18](#_Toc526256076)

[Steps during Submission Process 18](#_Toc526256077)

[Scenario 1 # 526ez and form 4142 submissions are successful 19](#_Toc526256078)

# Overview

The collection of Private Medical Records (PMRs) as supporting evidence in a claim for increase is initiated via forms 21-4142 and 21-4142a. We will refer to these as 4142 and 4142a or collectively as PMR data.

The Veterans Benefits Administration (VBA) uses VA Form 21-4142, Authorization to Disclose Information to the Department of Veterans Affairs (VA), to request private medical records and information regarding the source of records in support of claims for compensation and/or pension benefits. The response helps VA determine whether a source identified by the claimant has pertinent information or records available to consider.

# 526 / 4142 Technical Design

Below is a summary of the 526/4142 process. Ownership of these tasks needs to be identified since some of these changes require updates to files created and managed by Ad Hoc.

## Vets UI

* 4142 pages will be added to 526 form - BAH
* Schema for 4142 will be imported into the 526 schema - BAH
* Populate veteran information for the form 4142 during translation in 526 – Ad Hoc
  + Full Name, SSN, Date of Birth, Address, Email, Phone, VA Service Number and VA File number
  + Available in user profile
  + Example in 526
* Update the PMR choice page -BAH
  + <https://staging.vets.gov/disability-benefits/apply/form-526-disability-claim/supporting-evidence/0/private-medical-records-choice>
  + Need to add patient authorization text and checkbox
  + Need to remove download option
* Composite PMR – Flow needs to change to collect providers on one screen instead of per condition. Pending UI mockups
* Summary of Evidence Page – Display provider names - BAH
* Review and Submit page - BAH
  + Ability to edit all PMR data on “Review and Submit” page
  + <https://staging.vets.gov/disability-benefits/apply/form-526-disability-claim/review-and-submit>

## Vets API

* 526 submit processing includes 4142 in JSON payload – Ad Hoc
* 526 API saves data to saved\_claims table – Ad Hoc
* API invokes 526 async process– Ad Hoc
* Async process will split out 526 and 4142 data – Ad Hoc (BAH to define required elements) – Ad Hoc
* After completion of 526 EVSS submission the 4142 Async process will be invoked with 4142 JSON object – Ad Hoc
* 4142 async process creates PDF from 4142 JSON - BAH
* 4142 async submits 4142 via ICMHS API – BAH
* If 4142 async fails after maximum retires a message is added to the Dead Letter Queue and a notification is sent to Production Operations (*Notification part to be confirmed with Ad Hoc and DevOps*)

## Outstanding Items

* Failure processing for 526 async and 4142 async
* Correlating 526 submission with 4142 (Tracked Item)

# Design Considerations

## Providers per condition

The 526 UI collects supporting evidence for each rated condition in the evidence category of “VA Medical Records” , “Private Medical Records” and “Lay statements or other evidence”. The 4142a does not associate providers to a condition. For the implementation of the 4142 flow all providers will be included on a single 4142a and they will not be associated with a condition.

## Form Builder

The vets.gov platform includes a Form Builder capability to provide creation and processing of web forms. The forms are associated with the vets-api via a schema to support claims processing. In reviewing the options for processing of Private Medical Records there were two options for the processing of PMR related data.

### Option 1

The PMR data could be added to the 526 Schema for the web and API processing. This would minimize the amount of new code to be developed as it would leverage the existing 526 components and processing. This would also require coordination between the Ad Hoc and BAH teams as both teams would be potentially modifying the same components. Option 1 provides less flexibility to 4142 form UI as it would become tightly coupled within 526ez schema. So, in future if we need to trigger 4142 in a different flow this design won’t support it

### Option 2

Use the Form Builder to create a new web form to support the collection of PMR data. This approach requires implementation of all the processing components but minimizes dependencies with the 526 code base. This will also introduce some design considerations such as navigation components as the 4142 form will show up as a form in the navigation. Option 2 provides flexibility for the 4142 form UI as the form could be triggered in a different flow and integrated with it

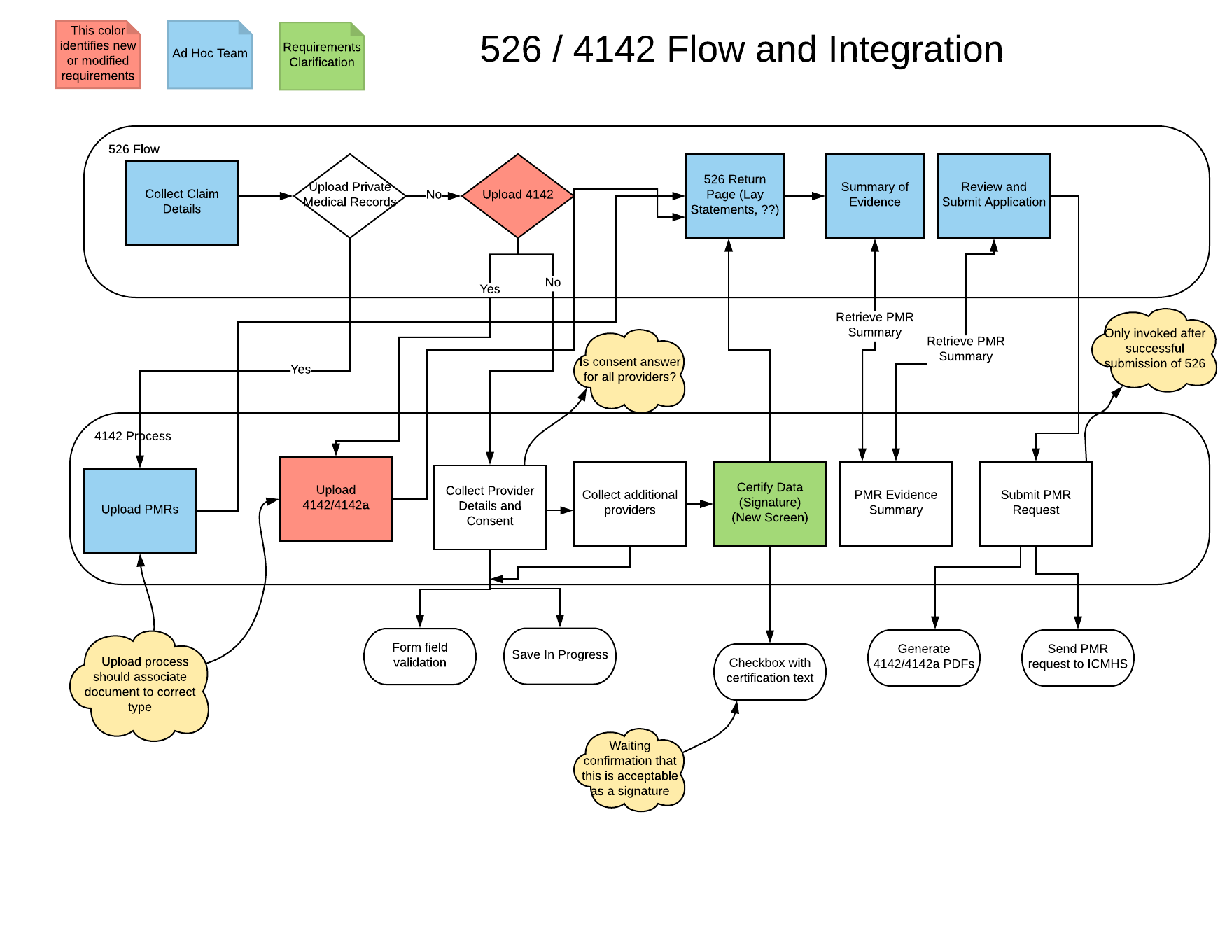
### Design Decision

Option 1 will be leveraged to support the PMR data processing. This option was chosen to eliminate the need to pass data between the two forms.

## 526/4142 Integration

### 526/4142 Flow

The diagram below depicts the proposed 526 flow and interactions with the PMR data processing components.



### Flow Questions / Comments

In 526 flow, interview question is “Do you want to upload your private medical records?” when the Veteran selects “No, my doctor has my medical records”, then in the next screen we start collecting provider information. I am not clear on where again the Yes option is for modified requirements to still upload private medical records?

In 526 flow, interview question is “Do you want to upload your private medical records?” if the Veteran changes the option and selects “Yes” after entering the provider data, then 526 flow must identify it and call an end point to DELETE existing in progress form data for 4142, if there isn’t a “No” option selected for any / all conditions.

On the “Summary of Evidence” page, we will need to display the private provider names on 526. So, we would need an API end point to support returning the list of private provider names.

On “Review and Submit Application” page, do we need any additional link to edit the 4142 form?

In the last flow of “Submit PMR Request” there is a sequential flow to “Generate 4142/4142a PDFs” first and then do submit for “Send PMR document and metadata to ICMHS”

## External Processing of PMR data

### Overview

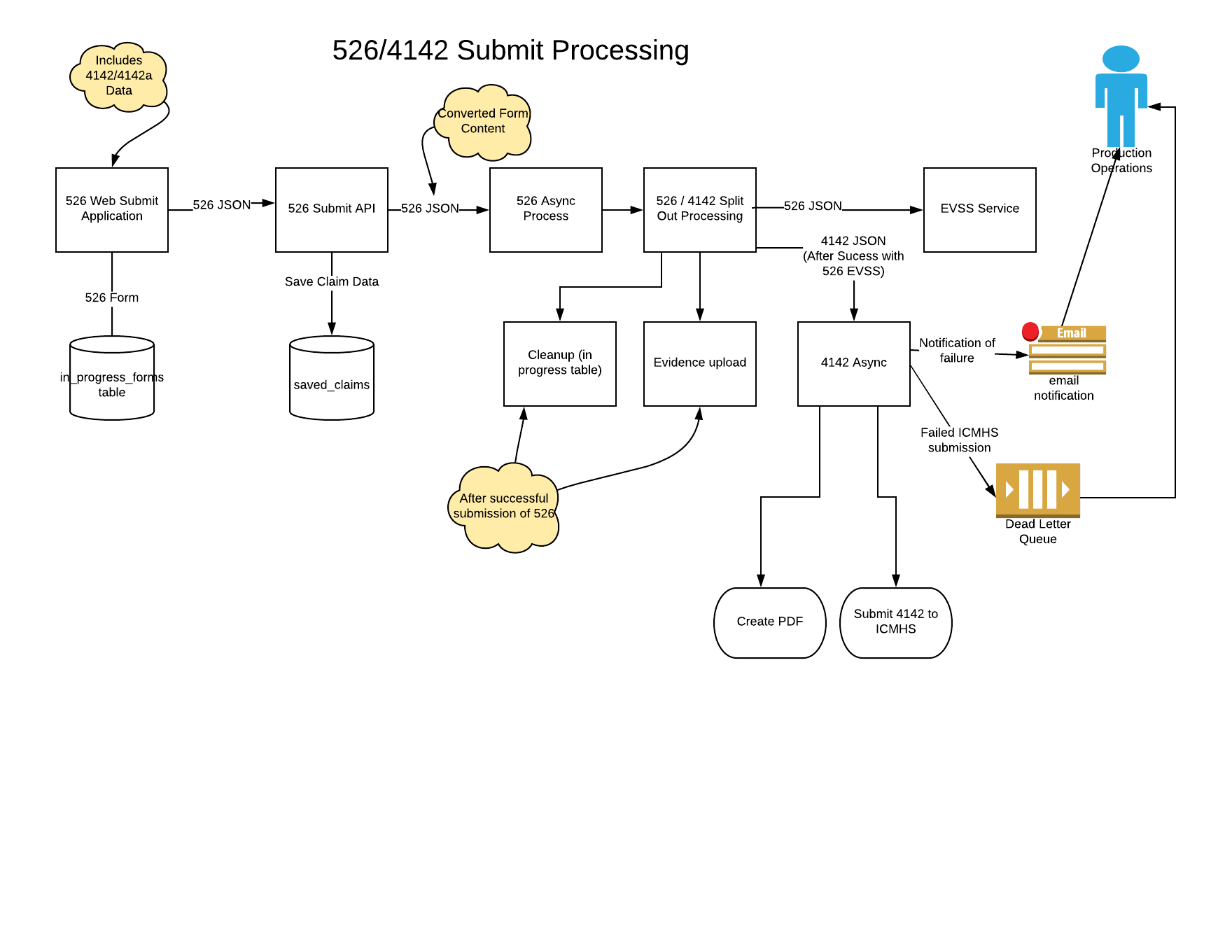
The 4142 and 4142a data (PMR data) require processing by the PMR vendor to retrieve the medical records and upload to VBMS E-Folder. An API has been implemented that can support this processing. The Intake, Conversion and Mail Handling Services (ICMHS) API is currently being leveraged by the Pension and Burials forms for similar processing. This interface is sometimes referred to as the Benefits Intake API and VCIP. For the purposes of this document we will use ICMHS when describing interactions with the API. The process flow for ICMHS is shown below. The 526 flow leverages a different API based on EVSS to submit data and upload forms to the VBMS E-Folder. This presents a challenge in associating the PMR data with the 526 Claim data. Processing of the PMR data can take up to 20 days whereas the 526 data is processed near real time. The VBMS system provides the ability to correlate this data through a Tracked Item. This capability needs to be explored as a solution for the 526/4142 flow.

### PMR Data Processing

The PMR processing will leverage the vets.gov platform capabilities via vets-website, vets-json-schema, and vets-api for processing of the data collected via the web interface. This requires interaction with the 526 flow as described above. Vets will provide claim data and supporting evidence through the 526 flow. The platform provides a save in progress capability so that Vets can add data incrementally. Once all the data is completed the Vet will certify the data by selecting a checkbox associated with certifying text. It is assumed this certification dialogue is sufficient for applying a digital signature to the 4142. At the completion of the flow the Vet will click the “Submit” claim button. This will trigger the submission of the 526ez claim data. This will be processed asynchronously. After the 526ez processing component has received confirmation that the 526 was submitted it will trigger the asynchronous processing of the 4142 data.

## 4142 Back End Processing

The diagram below depicts the process flow for 4142 data and the integration with the 526 flow.



The 526 Async process will receive JSON content that contains both the 526 and 4142/4142a data. The process will extract the 526 data and submit to EVSS for processing. Upon successful processing (acknowledgement from EVSS), the 526 async process will invoke the 4142 async process with the extracted 4142 data and veteran profile data. The 526 async process will need to translate the extract JSON to a format for the 4142 process. The format is documented below.

Value statement

As a development team member, I need to ensure that everyone understands the JSON structure so that the 4142 form can be integrated into the 526EZ.

### JSON Implementation Details

For 4142 and 4142A forms, PDF is generated using a JSON form content. As a part of 526 integration, there was an agreement that Ad Hoc will own completing the code changes to extract 4142 form content from 526 form, translate it into the JSON format, save it and pass it as expected by the background job of 4142.

Expected sample JSON file for 4142 PDF generation is available on: https://github.com/department-of-veterans-affairs/vets-api/blob/master/spec/fixtures/pdf\_fill/21-4142/kitchen\_sink.json and included below.

4142 Background Job Initial Version is available on: https://github.com/department-of-veterans-affairs/vets-api/blob/master/app/workers/central\_mail/submit\_form4142\_job.rb

Details of the JSON fields are as below:

* Fields populated from pre-fill information as below:  
  - "veteranFullName", "vaFileNumber", "veteranSocialSecurityNumber", "veteranServiceNumber" , "veteranDateOfBirth", "veteranAddress { "city","country","postalCode", "street","street2","state"}, "email", "veteranPhone"
* Fields available from 526 integrated form as captured from 4142 UI as below:
  + "providerFacility" [{"providerFacilityName","treatmentDateRange"[{"from","to"}],"providerFacilityAddress" {"street","street2","city","state","country","postalCode"}}, "privacyAgreementAccepted", "limitedConsent"

**4142 definition to be integrated into 526 Schema is shown below:**

*const form4142 = {type: 'object',properties: {limitedConsent: {type: 'string'},providerFacility: {type: 'array',required: ['providerFacilityName','treatmentDateRange','providerFacilityAddress'],items: {type: 'object',properties: {providerFacilityName: {type: 'string'},treatmentDateRange: {type: 'array',properties: {from: {$ref: '#/definitions/date'},to: {$ref: '#/definitions/date'}}},providerFacilityAddress: {$ref: '#/definitions/centralMailAddress'}}}},*

*privacyAgreementAccepted: {$ref: '#/definitions/privacyAgreementAccepted'}}};*

**Sample 4142 JSON Payload:**

|  |
| --- |
| { |
|  | "privacyAgreementAccepted": true, |
|  | "vaFileNumber": "12345678", |
|  | "veteranSocialSecurityNumber": "123456789", |
|  | "veteranServiceNumber": "987654321", |
|  | "veteranFullName": { |
|  | "first": "Testy", |
|  | "middle": "Tester", |
|  | "last": "Testerson" |
|  | }, |
|  | "veteranDateOfBirth": "1981-11-05", |
|  | "veteranAddress": { |
|  | "city": "Baltimore", |
|  | "country": "USA", |
|  | "postalCode": "21231-1234", |
|  | "street": "street", |
|  | "street2": "1B", |
|  | "state": "MD" |
|  | }, |
|  | "email": "testy.testerson@gmail.com", |
|  | "veteranPhone": "2123456789", |
|  | "limitedConsent": "limit1 limit2 limit3 limit4 limit1 limit2 limit3 limit4 limit1 limit2 limit3 limit4", |
|  | "providerFacility": [ |
|  | { |
|  | "providerFacilityName": "provider 1", |
|  | "treatmentDateRange": [ |
|  | { |
|  | "from": "1980-1-1", |
|  | "to": "1985-1-1" |
|  | }, |
|  | { |
|  | "from": "1986-1-1", |
|  | "to": "1987-1-1" |
|  | } |
|  | ], |
|  | "providerFacilityAddress": { |
|  | "street": "123 Main Street", |
|  | "street2": "1B", |
|  | "city": "Baltimore", |
|  | "state": "MD", |
|  | "country": "USA", |
|  | "postalCode": "21200-1111" |
|  | } |
|  | }, |
|  | { |
|  | "providerFacilityName": "provider 2", |
|  | "treatmentDateRange": [ |
|  | { |
|  | "from": "1980-2-1", |
|  | "to": "1985-2-1" |
|  | }, |
|  | { |
|  | "from": "1986-2-1", |
|  | "to": "1987-2-1" |
|  | } |
|  | ], |
|  | "providerFacilityAddress": { |
|  | "street": "456 Main Street", |
|  | "street2": "1B", |
|  | "city": "Baltimore", |
|  | "state": "MD", |
|  | "country": "USA", |
|  | "postalCode": "21200-1111" |
|  | } |
|  | }, |
|  | { |
|  | "providerFacilityName": "provider 3", |
|  | "treatmentDateRange": [ |
|  | { |
|  | "from": "1980-3-1", |
|  | "to": "1985-3-1" |
|  | }, |
|  | { |
|  | "from": "1986-3-1", |
|  | "to": "1987-3-1" |
|  | } |
|  | ], |
|  | "providerFacilityAddress": { |
|  | "street": "789 Main Street", |
|  | "street2": "1B", |
|  | "city": "Baltimore", |
|  | "state": "MD", |
|  | "country": "USA", |
|  | "postalCode": "21200-1111" |
|  | } |
|  | }, |
|  | { |
|  | "providerFacilityName": "provider 4", |
|  | "treatmentDateRange": [ |
|  | { |
|  | "from": "1980-4-1", |
|  | "to": "1985-4-1" |
|  | }, |
|  | { |
|  | "from": "1986-4-1", |
|  | "to": "1987-4-1" |
|  | } |
|  | ], |
|  | "providerFacilityAddress": { |
|  | "street": "101 Main Street", |
|  | "street2": "1B", |
|  | "city": "Baltimore", |
|  | "state": "MD", |
|  | "country": "USA", |
|  | "postalCode": "21200-1111" |
|  | } |
|  | }, |
|  | { |
|  | "providerFacilityName": "provider 5", |
|  | "treatmentDateRange": [ |
|  | { |
|  | "from": "1980-5-1", |
|  | "to": "1985-5-1" |
|  | }, |
|  | { |
|  | "from": "1986-5-1", |
|  | "to": "1987-5-1" |
|  | } |
|  | ], |
|  | "providerFacilityAddress": { |
|  | "street": "102 Main Street", |
|  | "street2": "1B", |
|  | "city": "Baltimore", |
|  | "state": "MD", |
|  | "country": "USA", |
|  | "postalCode": "21200-1111" |
|  | } |
|  | } |
|  | ] |
|  | } |

### Method signature for 4142 sidekiq background job

*# Performs an asynchronous job for submitting a Form 4142 to central mail service*

*#*

*# @param evss\_claim\_id [String] EVSS Claim id received from 526 submission to generate unique PDF*

*file path*

*# @param saved\_claim\_id [Integer] Saved Claim id*

*# @param form\_content [Hash] The form content for 4142 and 4142A that is to be submitted*

def perform(evss\_claim\_id, saved\_claim\_id, submission\_id, form\_content)

Refer to :  <https://github.com/department-of-veterans-affairs/vets-api/blob/master/app/workers/central_mail/submit_form4142_job.rb>

For error handling, an alert will be sent in the event the 4142 retries are exhausted or non-retryable. The message will also be added to the Dead Letter Queue.

There were 2 design options deliberated to track the asynchronous transaction and capture the ancillary forms submission status

***Option 1# Write an Asynchronous transaction record for 4142 submission to Central Mail (ICMHS)***

* From within Sidekiq job, this option would create a new asynchronous transaction record for 4142 form submission
* In 526 job, it creates a separate asynchronous transaction as needed for the front end to poll the status and report to the user. 526 also has to hit multiple EVSS the sub-systems, more likely would run into timeout and/or some non-retryable validation error and hence stores into transaction separately
* Sidekiq job retrying is not dependent on the job being tracked by a class that extends class “AsyncTransaction”. Asynchronous transactions do not help with monitoring failures as production support resource would need to get on a production server and run a query for any root cause analysis
* 4142 asynchronous job is kicked off by the 526 job, so there's no way to pass the job id up to the front end for it to poll. 4142 ancillary form submission is basically similar to evidence medical documents upload and the user doesn’t have to track these submissions as a separate process from 526
* If 4142 job fails, it is easier to access via Sentry or AWS logs. Resending an exhausted job(s) will take manual intervention in any case
* Moving 526 to a background job was never a first choice for the team. Primary reasons were the consistent timeouts for the endpoints in particular from EVSS. The subsequent tasks are in jobs as they depend on EVSS submission success. However, as they don't need to report their status, outside of logging failures, they're not backed by asynchronous transaction records

***Option 2# Write a new Tracker and Reporting class that writes into a table to track the status for ancillary forms (Proposed by***

***AdHoc)***

* New tracker and reporting classes to be written and made available to the ancillary forms for reporting their status. AdHoc has already started work on it and can be tracked via following items
* <https://github.com/department-of-veterans-affairs/vets-api/pull/2326>
* <https://github.com/department-of-veterans-affairs/vets.gov-team/issues/13659>
* This implementation will allow the system to pull a daily report on the status of 526 submissions and check jobs / logs to verify the status of ancillary items along with the main form submission
* Reporting for the 4142 job would be similar to evidence document uploads. Evidence uploads is a Sidekiq batch job, so it uses a callback for success, however for 4142 job it would report its success inline

After considering the above options, we accept Option 2 for 4142, as it more aligns with 526 submissions for evidence upload documents and assuming that it would support future ancillary forms. By selecting Option 2, we expect design decisions and changes to be completed for 4142.

* Discussion with AdHoc to review the proposed option 2 and understand the extensibility for other ancillary forms (Sprint 4)
* Remove 4142 Asynchronous Transaction class, any related unit tests and updates to remove references (DONE in bah-sprint3 branch)
* Add Inline call to the new reporting class to update 4142 status (Sprint 5)
  + <https://github.com/department-of-veterans-affairs/vets.gov-team/issues/13837>

### PDF Generation

The 4142 Async job will take the form data that has been collected and create a single PDF consisting of the 4142 and 4142a data. The PDF creation will leverage the platform capability including the application of a digital signature to the document. It is assumed that the 4142 processing would not be invoked unless the Vet certified the data via the certification checkbox. The 4142 job will create the necessary meta-data and leverage the ICMHS API to submit the PMR PDF for processing by the PMR vendor.

### ICMHS Flow



# Design Decisions

Below are some design decisions that need to be validated and approved. The approach for these items are described above but summarized here for consideration.

## Design Points

* Signature
* Tracked Items
* Providers collected per condition
* Limited Consent Text Box – Web Interface doesn’t provide ability to enter free text as is allowed on PDF form. Need text to be provided if Vet is limiting consent. Also need to confirm we can certify the data since this was not entered by the Vet
* Certification of data

## Requirements Refinements / Questions

* Upload of completed 4142 and 4142a forms – During a requirements session the option of uploading completed 4142 and 4142a was discussed. This requirement needs to be evaluated, approved, and then implemented which will require new questions in the flow.
* Text on the following page indicates a 4142 must be completed for each doctor (<https://staging.vets.gov/disability-benefits/apply/form-526-disability-claim/supporting-evidence/0/private-medical-records-choice>) Text - *Since your doctor has your private medical records, you’ll need to fill out an Authorization to Disclose Information to the VA (VA Form 21-4142) so we can request your records. You’ll need to fill out a form for each doctor.*
* Edit option on Review and Submit Page - <https://staging.vets.gov/disability-benefits/apply/form-526-disability-claim/review-and-submit> – Veteran can edit their option to provide PMR data on the summary screen. In staging site this doesn’t take them back to screen for adding PMR data.

# 526 / 4142 Submission Process

The section documents several conditions, system behavior, track and reporting status during 526 and 4142 submission process.

## Steps during Submission Process

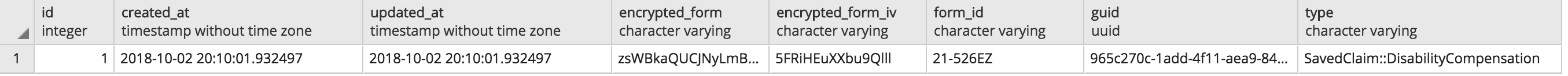
1. Veteran submits 526 application with the included private medical records (PMR) form 4142 data
2. 526 controller [*V0::DisabilityCompensationFormController*] submit method parses the entire form as JSON object
3. If form4142 is present in 526 form, then the class [*EVSS::DisabilityCompensationForm::Form4142*] translates it to the expected JSON format for 4142 background sidekiq job to process it
4. Form 526 data translation is accomplished by the class [*EVSS::DisabilityCompensationForm::DataTranslation*]
5. 526 controller then asynchronously invokes submit form 526 background job [*EVSS::DisabilityCompensationForm::SubmitForm526*] passing multiple parameters, including the translated form 4142
6. 526 background job then submits form526 to an upstream submission service (currently EVSS) & returns Claim ID in response
7. 526 background job then asynchronously invokes 4142 background job [*CentralMail::SubmitForm4142Job*]
8. 4142 background job generates, stamps 4142/4142A PDF document. Job then uploads the document to Central Mail using Benefits Intake API (ICMHS)
9. 526 background job then asynchronously invokes 526 clean up background job (*EVSS::DisabilityCompensationForm::SubmitForm526Cleanup*] for cleaning up 21-526EZ form data from InProgress table and REDIS cache
10. If evidence uploads are available, 526 background job then asynchronously invokes background batch job [*EVSS::DisabilityCompensationForm::SubmitUploads*] to upload evidence attachments into VBMS via EVSS document service

During 526ez submit, below are the expected possibilities:

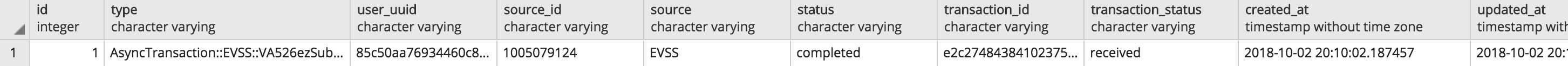
1. Both 526 and 4142 submissions are successful
2. 526 submission successful, but 4142 fails
3. 526 submission fails, and hence the process doesn’t attempt to submit 4142

## Scenario 1: Both 526 and 4142 submissions are successful

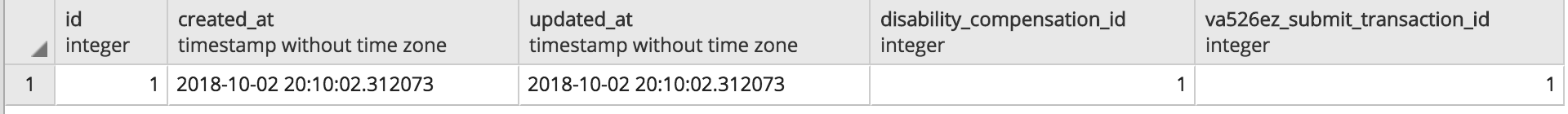
* In this scenario, it can be concluded that Steps 6 and 8 were successfully completed
* Post submit for the form with id **21-526EZ** (Claims for Increase), PostgreSQL tables would contain records as shown below (example from developer local testing)
  + saved\_claims



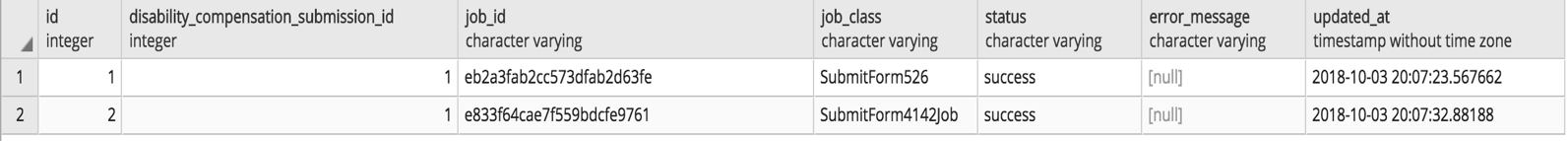
* + async\_transactions



* + disability\_compensation\_submissions



* + disability\_compensation\_job\_statuses



* 4142 submission job status is tracked in the newly created table “disability\_compensation\_job\_statuses”.
  + PR from Ad Hoc for the new table creation and related code changes. BAH also completed some code changes for 4142 integration work within the same Pull Request
    - <https://github.com/department-of-veterans-affairs/vets-api/pull/2326>
* Rails application log with the entries as shown below for successful submissions of 526 and 4142

2018-10-04 16:04:11.461125 D [13:70316550366300] Rails -- [upsert]

Selector: {"job\_id"=>"4bb8e6d557d4fe0d4696a67a"}

Setter: {"disability\_compensation\_submission\_id"=>1, "job\_class"=>"SubmitForm526", "job\_id"=>"4bb8e6d557d4fe0d4696a67a", "status"=>"success", "updated\_at"=>2018-10-04 16:04:11 UTC}

2018-10-04 16:04:11.522771 I [13:70316550366300] Rails -- Form526 Submission -- { "saved\_claim\_id" => 1, "submission\_id" => 1, "job\_id" => "4bb8e6d557d4fe0d4696a67a", "event" => "success" }

2018-10-04 16:04:11.522836 I [13:70316550366300] Rails -- Form526 Submission -- { "saved\_claim\_id" => 1, "submission\_id" => 1, "job\_id" => "4bb8e6d557d4fe0d4696a67a", "event" => "try" }

2018-10-04 16:04:15.569678 D [13:86655540] Rails -- [upsert]

Selector: {"job\_id"=>"122c7b160b859915fb3749dc"}

Setter: {"disability\_compensation\_submission\_id"=>1, "job\_class"=>"SubmitForm4142Job", "job\_id"=>"122c7b160b859915fb3749dc", "status"=>"success", "updated\_at"=>2018-10-04 16:04:15 UTC}

2018-10-04 16:04:15.575856 I [13:86655540] Rails -- Form4142 Submission -- { "saved\_claim\_id" => 1, "submission\_id" => 1, "job\_id" => "122c7b160b859915fb3749dc", "event" => "success" }

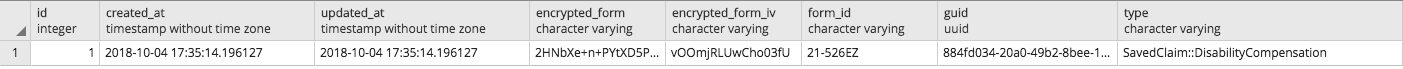
2018-10-04 16:04:15.575915 I [13:86655540] Rails -- Form4142 Submission -- { "saved\_claim\_id" => 1, "submission\_id" => 1, "job\_id" => "122c7b160b859915fb3749dc", "event" => "try" }

## Scenario 2: 526 submission successful, but 4142 fails

* For this scenario, it can be concluded that Steps 6 was successful, however there were failures during Step 8
* For 4142 PDF upload, ICMHS API call will immediately return one of the following codes to vets.gov

|  |  |
| --- | --- |
| Error Code | Description |
| 200 | Success – ICMHS received successfully, but not processed and at VBMS |
| 400 | Duplicate – document already uploaded with that uuid |
| 401 | Unauthorized – missing or incorrect token |
| 409 | Conflict - mismatched attachments and numbers |
| 412 | Precondition Failed – for meta data failures |
| 415 | Unsupported File type |
| 422 | Unprocessable Request– no file, or issue with PDF, missing attachments |
| 500 | Server Error |

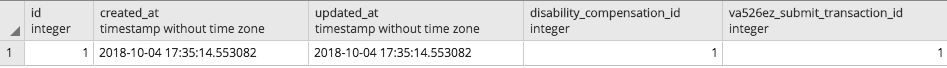
* 4142 background job will retry submissions for all the conditions listed below
  + ICMHS returns error code in the range of 201-600
  + ICMHS connection times out (Error of type “*Common::Exceptions::GatewayTimeout”*)
  + Any other standard error (Error of type “StandardError”) raised during processing
* Post submit for the form with id **21-526EZ** (Claims for Increase), PostgreSQL tables would contain records as shown below (example from developer local testing)
  + saved\_claims



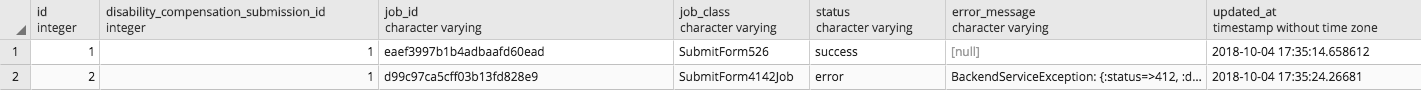
* + async\_transactions

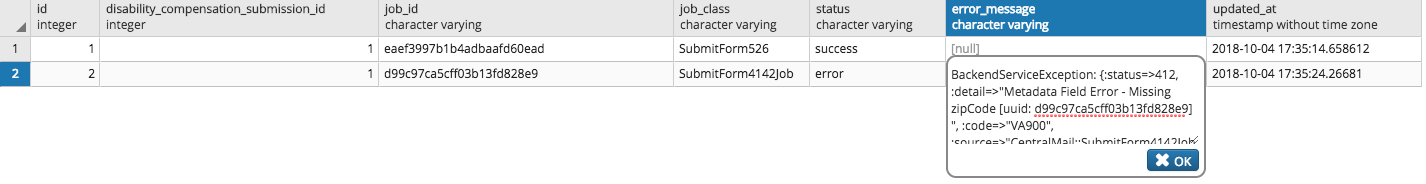


* + disability\_compensation\_submissions



* + disability\_compensation\_job\_statuses





* Rails application log with the entries as shown below for the successful 526 submission, but failed 4142

2018-10-04 17:35:14.718440 D [13:43130340] Rails -- [upsert]

Selector: {"job\_id"=>"eaef3997b1b4adbaafd60ead"}

Setter: {"disability\_compensation\_submission\_id"=>1, "job\_class"=>"SubmitForm526", "job\_id"=>"eaef3997b1b4adbaafd60ead", "status"=>"success", "updated\_at"=>2018-10-04 17:35:14 UTC}

2018-10-04 17:35:14.767671 I [13:43130340] Rails -- Form526 Submission -- { "saved\_claim\_id" => 1, "submission\_id" => 1, "job\_id" => "eaef3997b1b4adbaafd60ead", "event" => "success" }

2018-10-04 17:35:14.767760 I [13:43130340] Rails -- Form526 Submission -- { "saved\_claim\_id" => 1, "submission\_id" => 1, "job\_id" => "eaef3997b1b4adbaafd60ead", "event" => "try" }

2018-10-04 17:35:24.266094 E [13:43127280 submit\_form4142\_job.rb:144] Rails -- { "Error Message" => "BackendServiceException: {:status=>412, :detail=>\"Metadata Field Error - Missing zipCode [uuid: d99c97ca5cff03b13fd828e9] \", :code=>\"VA900\", :source=>\"CentralMail::SubmitForm4142Job\"}" }

2018-10-04 17:35:24.266266 E [13:43127280 sentry\_logging.rb:42] Rails -- BackendServiceException: {:status=>412, :detail=>"Metadata Field Error - Missing zipCode [uuid: d99c97ca5cff03b13fd828e9] ", :code=>"VA900", :source=>"CentralMail::SubmitForm4142Job"}.

2018-10-04 17:35:24.276371 D [13:43127280] Rails -- [upsert]

Selector: {"job\_id"=>"d99c97ca5cff03b13fd828e9"}

Setter: {"disability\_compensation\_submission\_id"=>1, "error\_message"=>"BackendServiceException: {:status=>412, :detail=>\"Metadata Field Error - Missing zipCode [uuid: d99c97ca5cff03b13fd828e9] \", :code=>\"VA900\", :source=>\"CentralMail::SubmitForm4142Job\"}", "job\_class"=>"SubmitForm4142Job", "job\_id"=>"d99c97ca5cff03b13fd828e9", "status"=>"error", "updated\_at"=>2018-10-04 17:35:24 UTC}

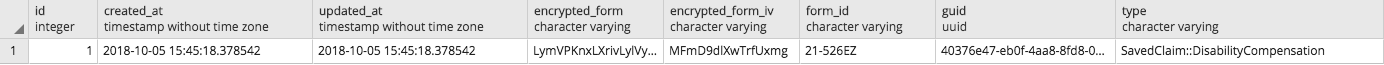
2018-10-04 17:35:24.282209 E [13:43127280 job\_status.rb:53] Rails -- Form4142 Submission -- { "saved\_claim\_id" => 1, "submission\_id" => 1, "job\_id" => "d99c97ca5cff03b13fd828e9", "event" => "error", "error\_message" => "BackendServiceException: {:status=>412, :detail=>\"Metadata Field Error - Missing zipCode [uuid: d99c97ca5cff03b13fd828e9] \", :code=>\"VA900\", :source=>\"CentralMail::SubmitForm4142Job\"}" }

2018-10-04 17:35:24.282476 I [13:43127280] Rails -- Form4142 Submission -- { "saved\_claim\_id" => 1, "submission\_id" => 1, "job\_id" => "d99c97ca5cff03b13fd828e9", "event" => "try" }

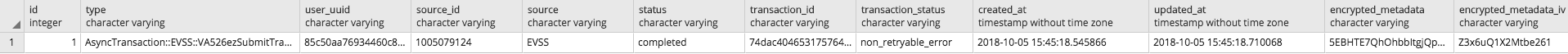
* As seen from the Rails logs, submit of 4142 failed with error code 412 is returned from ICMHS. Error details are logged and recorded in the “disability\_compensation\_job\_statuses” table
* For 4142 background job, a maximum retries of 10 is configured which will likely result in a runtime of ~8 hours
* **For 526 or 4142 submissions, there will LESS likely be a scenario where 526 form and 4142 PDF will be submitted with some missing data that went un-caught during schema validation. However, for this less likely scenario the ONLY way submissions can later be re-tried if the following ACTIONS are POSSIBLE (Ad Hoc and DevOps to confirm the feasibility of the below actions)**
  + **Job is available to be retried. For job to be retries, it has to be in the Dead or Retries Queue**
  + **Arguments passed to the job (form, headers, user info etc.) can be fixed if required for certain cases before manual / configured retry happens in sidekiq.**

## Scenario 3: 526 submission fails, 4142 submit isn’t attempted

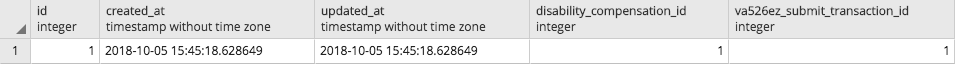
* For this scenario, it can be concluded that Step 6 was failed and hence Step 8 didn’t execute
* 526 submission to EVSS can fail for various reasons of application error, backend EVSS system error and timeouts
* Post submit for the form with id **21-526EZ** (Claims for Increase), PostgreSQL tables would contain records as shown below (example from developer local testing)
  + saved\_claims



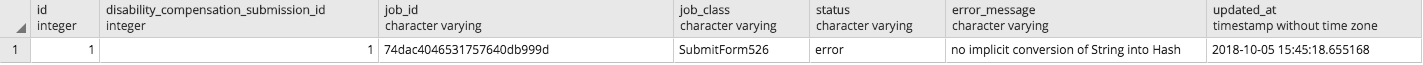
* + async\_transactions



* + disability\_compensation\_submissions



* + disability\_compensation\_job\_statuses



* For 526 background job, a maximum retry of 13 is configured which will likely result in a runtime of ~25 hours (This is as per the comment in EVSS::DisabilityCompensationForm::SubmitForm526 )
* 526 Background job follows the below conditions for retries
  + EVSS timeout errors are **re-tried** for submissions
  + Any other service exceptions or standard errors are logged into sentry, but **NOT** **retried**
* For timeout errors, if 526 is submitted successfully within the window of 13 retries, then 4142 will be attempted and follow the path mentioned in Scenario 2
* For any other service or standard errors, the errors and details will be logged into sentry and no retries will be attempted. In this case, manual intervention is needed. **UNCLEAR about this process yet, need a discussion with AdHoc and DevOps**. The only identified option for now is getting on a box, then via rails console use the sidekiq api to fix the data. UNCLEAR to me yet if this process even works or has been tried before.
* 4142 submission will be complete **ONLY** when 526 submission job is available for retry and is completed without errors.