

# **Department of Veterans Affairs**

**VLER Direct Access Services (DAS) and VIERS Electronic Form  
Submission Service (EFSS)**

**Development Roadmap and Service Scenarios Document  
Document Version 1.0**



**December 2015**

## Revision History

<b>Date</b>	<b>Version</b>	<b>Description</b>	<b>Author</b>
11/12/2015	0.11	Incorporate suggestion from Panoramic	VRM TI
11/04/2015	0.10	Incorporate development team review changes	VRM TI
10/29/2015	0.9	Incorporate internal review changes	VRM TI
10/28/2015	0.8	Incorporate internal review changes	VRM TI
10/28/2015	0.8	Incorporate internal review changes	VRM TI
10/27/2015	0.7	Include additional scenarios to section <b>5 Service Scenarios</b>	VRM TI
10/26/2015	0.6	Incorporate internal review changes	VRM TI
10/08/2015	0.5	Incorporate suggestion from Panoramic	VRM TI
10/08/2015	0.4	Renamed document Incorporate suggestion from other Vendors and VSOs	VRM TI
10/01/2015	0.3	Changes from review with Panoramic	VRM TI
09/25/2015	0.2	Changes from reviews.	VRM TI
09/21/2015	0.1	Initial Draft	VRM TI
12/08/2015	1.0	Finalized draft versions	VRM TI

## Table of Contents

<b>1. Introduction</b>	<b>1</b>
1.1. Document Purpose	1
<b>2. Background</b>	<b>1</b>
2.1. D2D Components	1
2.2. Transaction Lexicon	3
2.3. Overview of VSO to DAS to VSO Submission Structure	4
2.4. Overview of DAS to EFSS Submission Structure	5
2.5. VSO to DAS to EFSS Submission Structure Submit Form Payload	6
2.5.1. VSO to DAS Submission Structure	6
2.5.2. DAS to EFSS Submission Structure	7
2.5.3. VSO to DAS Submit Form Payload Creation	8
2.6. VSO to DAS to EFSS Submission Structure Submit Attachment Payload	10
2.6.1. VSO to DAS Submission Structure	10
2.6.2. DAS to EFSS Submission Structure	11
2.6.3. VSO to DAS Submit Attachment Payload Creation	12
<b>3. DAS Service Operations Used By the VSO</b>	<b>14</b>
3.1. ProvideAndRegisterDocumentSet-bRequest	14
3.1.1. VSO.submitForm Option	14
3.1.2. VSO.submitAttachmentForm Option	15
3.1.3. VSO.checkStatus Option	15
3.1.4. VSO.confirmSubmission option	15
3.2. ProvideAndRegisterDocumentSet-bResponse	16
3.2.1. EFSSResponseType	16
3.2.2. VSO.submitForm-response	16
3.2.3. VSO.submitForm-response	16
<b>4. Service Protocol</b>	<b>17</b>
4.1. Form Transmission Protocol	18
4.2. Attachment Submission Protocol	20
4.2.1. Form PDF Attachment Transmission Protocol	22
4.2.2. PDF Attachment Transmission Protocol	23
4.2.3. Final Form Processing Protocol	23
<b>5. Service Scenarios</b>	<b>24</b>
5.1. Successful 21-526EZ submission	25
5.2. Unsuccessful 21-526EZ submission - Business Rule Failure	26
5.3. Unsuccessful 21-526EZ Submission – Failure to Connect to DAS at Beginning of Submission	27

<b>5.4. Unsuccessful 21-526EZ Submission – DAS Response Failure during Submission Processing .....</b>	<b>28</b>
<b>5.5. Unsuccessful 21-526EZ Submission - VDC Error .....</b>	<b>30</b>
<b>5.6. Known Issues with the Messaging Back to the VSO .....</b>	<b>31</b>

## Table of Figures

Figure 1 - D2D Logical Component Overview. ....	2
Figure 2 – ProvideAndRegisterDocumentSetRequest Conceptual Diagram.....	6
Figure 3 – ProvideAndRegisterDocumentSetRequest Example .....	6
Figure 4 – submitFormRequestMessage Conceptual Diagram .....	7
Figure 5 – submitFormRequestMessage Example .....	7
Figure 6 – ProvideAndRegisterDocumentSetRequest Submit Form Example .....	9
Figure 7 – ProvideAndRegisterDocumentSetRequest Conceptual Diagram.....	10
Figure 8 – ProvideAndRegisterDocumentSetRequest Example .....	10
Figure 9 – submitAttachmentRequestMessage Conceptual Diagram.....	11
Figure 10 – submitAttachmentRequestMessage Example .....	11
Figure 11 – ProvideAndRegisterDocumentSetRequest Submit Attachment Example .....	13
Figure 12 - 21-526 Submit Form Submission Sequence Diagram.....	18
Figure 13 - 21-526 Submit Attachment Submission Sequence Diagram .....	21
Figure 14 – Successful 21-526EZ Form/Attachment Submission.....	25
Figure 15 – Unsuccessful 21-526EZ Form/Attachment Submission.....	26
Figure 16 – Unsuccessful 21-526EZ Form/Attachment Submission.....	27
Figure 17 – Unsuccessful 21-526EZ Form/Attachment Submission.....	28

## Table of Tables

Table 1: VSO to DAS Submission Structure.....	4
Table 2: DAS to VSO Submission Structure.....	4
Table 3: DAS to EFSS Form and Attachment Submission Structure .....	5
Table 4: DAS to EFSS Status and Confirmation Submission Structure .....	5

# 1. Introduction

## 1.1. Document Purpose

The purpose of the Development Roadmap and Service Scenarios Document is to provide Digits 2 Digits (D2D) Veteran Service Organization (VSO) vendor claims management system (CMS) developers and system integrators with the conventions governing submittals of claims and documents to the D2D system. The Development Roadmap and Service Scenarios Document will provide:

- Background on D2D including
  - components
  - lexicon describing transactions
  - overview of how to structure calls to D2D and available operations
- Service Protocol
- DAS Service Operations (API's)
- Procedures to be followed under different response scenarios

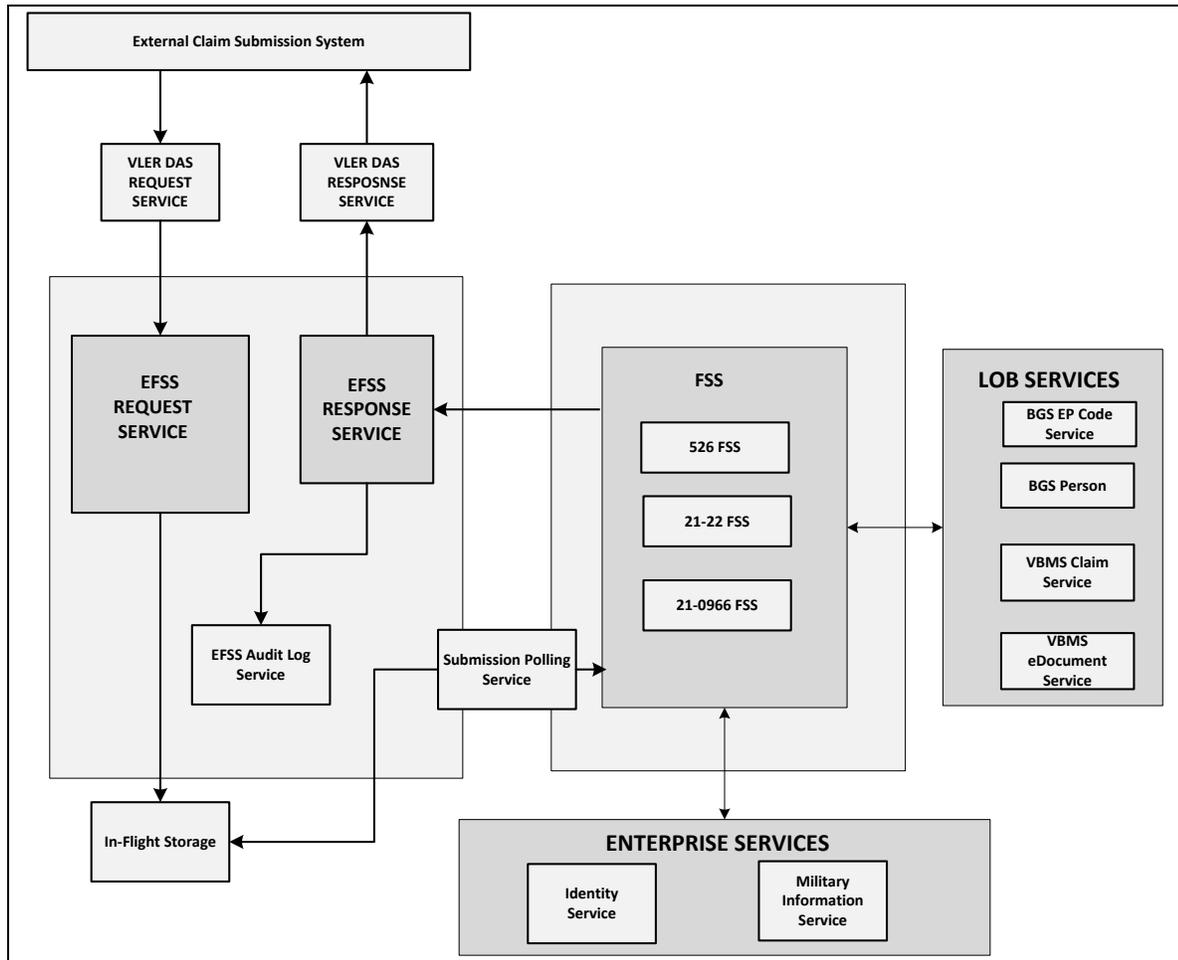
## 2. Background

### 2.1. D2D Components

The D2D service enables VSO CMS external to VA to claims as well as other forms via a SOAP submission to VA for electronic processing. This will eliminate paper records and speeding processing time for the Veteran. The CMS will submit SOAP packages to the VA's Digital Access Service (DAS) gateway. VA then uses a web orchestration of a number of services behind the gateway to process the submission. These services include:

- VIERS – Implementation of D2D functionality utilizing the Electronic Form Submission (EFSS) and Form Submission (FSS) Services framework.
  - EFSS – Implementation of the common functionality of the framework. The EFSS. It will receive forms and associated attachments as part of a submission and save them to the “hold area” in-flight storage. Upon successful completion of its responsibilities, it will initiate the appropriate FSS defined by form type (e.g., 21-526EZ)
  - FSS – Implementation of the specific functionality for a form type.
- In-Flight Storage – Service implementation for insertion of submitted forms and attachments into a “holding area”
- LOB Services Down line systems invoked by VIERS that supply specific business functionality needed by D2D
- Enterprise Services – Overarching services that are used across the VA processing infrastructure not just D2D

**Figure 1 - D2D Logical Component Overview.**



## 2.2. Transaction Lexicon

A single Transaction, sometimes referred to as a package, is analogous to a Submission, which will include one or more Transmissions. Transmissions include either a Form or an Attachment. The Manifest associates multiple Transmissions to a single Submission.

- Submission – A submission is a logical grouping of transmissions.
  - The Submission Identifier or Submission ID is unique for each submission. It is a unique, alphanumeric string generated by each VSO. The ID is not unique across VSOs, but is unique to each form transmission
  - The Submission ID associates Transmissions into a single package
- Transmission – A Transmission consists of a Form or Attachment
  - The Transmission Identifier or Transmission ID is unique for each transmission within a submission. It is a unique alphanumeric string generated by each VSO for each transmission
  - Each Transmission is submitted separately and in the case of a Form Transmission is accompanied with the Attachment Manifest
- Form – A form is a XML payload built per the XSD specification created by the VA for that form
- Attachment – An attachment is a PDF version of the specific form
  - Maximum attachment size is 25 MEG
  - The maximum number of attachments is currently set to 50 attachments per submission.
- Manifest – The Manifest is used to determine the completeness of attachment transmissions using the numberOfDocuments value versus attachments transmitted.

## 2.3. Overview of VSO to DAS to VSO Submission Structure

This section details the VSO to DAS to VSO web service submission structure. Detailed descriptions of payload packaging will be included in sections **2.5.3 VSO to DAS Submit Form Payload Creation** and **2.6.3 VSO to DAS Submit Attachment Payload Creation**

**Table 1: VSO to DAS Submission Structure**

Service	Operation	Description
DAS Request Service	ProvideAndRegisterDocument Set-bRequest	The VSO to DAS service request transmits the form or attachment using the ProvideAndRegisterDocument SetRequest element and responds using the Acknowledgement element.

**Table 2: DAS to VSO Submission Structure**

Service	Operation	Description
DAS Response Service	ProvideAndRegisterDocument Set-bResponse	The DAS to VSO response communicates the successful/unsuccessful processing of a single form or attachment or an entire submission using the RegistryResponse element and responds using the Acknowledgement element.

## 2.4. Overview of DAS to EFSS Submission Structure

This section details the DAS to EFSS web service submission structure. Although these web service interactions are not visible to the VSO, it was included for additional processing clarity.

**Table 3: DAS to EFSS Form and Attachment Submission Structure**

Service	Operation	Description
EFSS Request Service	submitForm	The DAS to EFSS service request transmits the form (a XML payload built per the D2D XSD specification) using the submitFormRequestMessage and responds using the submitFormResponseMessage element.
EFSS Request Service	submitAttachment	The DAS to EFSS service request transmits the attachment in a PDF format using the submitAttachmentRequestMessage and responds using the submitAttachmentResponseMessage element.

**Table 4: DAS to EFSS Status and Confirmation Submission Structure**

Service	Operation	Description
EFSS Request Service	confirmSubmission	The confirmSubmission uses the EFSSConfirmationType and responds using the EFSSResponseMessage element.  The operation allows an incomplete manifest to be cancelled and the “hold area” in-flight storage artifacts to be purged for a submission.
EFSS Request Service	checkStatus	The checkStatus operation uses the EFSSStatusType and responds using the EFSSAttachmentResponseMessage  The operation allows the consumer to check the status of a submitted form and all associated attachments within the “hold area” in-flight storage.

## 2.5. VSO to DAS to EFSS Submission Structure Submit Form Payload

The section contains conceptual diagrams and examples of the VSO to DAS to EFSS submitForm payload submission structure.

### 2.5.1. VSO to DAS Submission Structure

The ProvideAndRegisterDocumentSetRequest includes the Document element which contains the submitFormRequestMessage in a Base 64 format.

**Figure 2 – ProvideAndRegisterDocumentSetRequest Conceptual Diagram**



**Figure 3 – ProvideAndRegisterDocumentSetRequest Example**

```
<urn:ProvideAndRegisterDocumentSetRequest>
  <urn1:SubmitObjectsRequest id="a36d9442-e184-40e6-8b96-70c22f83dce7" >
    ...
    ...(Elements truncated for readability)
    ...
  </urn1:SubmitObjectsRequest>

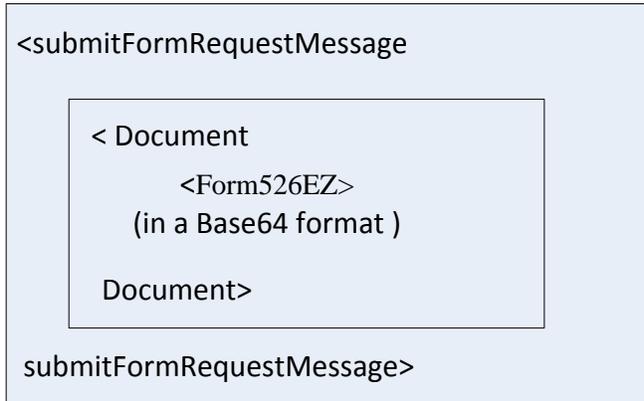
  <urn:Document id="0003" >PHYxOnN1Y...(Base64 payload truncated for readability)
    ...
    ...
    ==</urn:Document>
</urn:ProvideAndRegisterDocumentSetRequest>
```

## 2.5.2. DAS to EFSS Submission Structure

DAS will de-code the submitFormRequestMessage prior the interaction with EFSS.

The submitFormRequestMessage element includes the Document element which contains the Form526EZ (in the case of 526EZ transmission) element in a Base 64 format.

**Figure 4 – submitFormRequestMessage Conceptual Diagram**



**Figure 5 – submitFormRequestMessage Example**

```
<v1:submitFormRequestMessage xmlns:v1="http://viers.va.gov/efss/v1">
  ...
  <v1:sender>
    ... (Element truncated for readability)
  </v1:sender>
  <v1:formInfo>
    <v1:formType>21-526EZ</v1:formType>
    <v1:numberOfDocuments>0</v1:numberOfDocuments>
    ...
    <v1:Document id="0003" >PD94bWwg... (Base64 payload truncated for readability)
    ...
    ==</v1:Document>
    ...
  </v1:formInfo>
  ... (Element truncated for readability)
  <v1:EFSSManifestType>
    ... (Element truncated for readability)
  </v1:EFSSManifestType>
</v1:formInfo>
</v1:submitFormRequestMessage>
```

### 2.5.3. VSO to DAS Submit Form Payload Creation

This section will detail the step for the creation of the 21-526EZ form payload that will be compatible with the DAS, EFSS and 21-526EZ WSDL and XSD specifications.

1. The VSO will capture the 526EZ Form information and create the 21-526EZ Form XML. The **21-526EZ X.X zip** where X.X is the current D2D release on the VACI web site contains in addition to the 526EZ WSDL and XSD files, a sample 21-526EZ XML document. Use this as a guide for the automated VSO application XML payload creation process.
2. Validate the created 21-526EZ Form XML against the XSD included in the zip file. Business and syntactical rule validations for the created 21-526EZ Form XML file are contained in the **D2DDataDictionary VRM TI Updates Inc1ItX.X.xlsx** file where X.X is the current D2D release contained in the VACI web site
3. When the 21-526EZ Form XML payload has been created convert it to a Base 64 Encoded format
4. The **EFSS X.X zip** file where X.X is the current D2D release contained in the VACI web site contains a file named EFSSTypes.xsd. This file contains the XSD template for the **submitFormRequestMessage** element. Create and populate an XML file using this template. The Base64 Encoded format file created in step 3 is inserted in the **document** element that is contained in the **formInfo** element
5. Convert the file created using the **submitFormRequestMessage** template to a Base 64 Encoded format
6. The VSO submits a 21-526EZ payload with claim initiation information using the **ProvideAndRegisterDocumentSet-bRequest** operation hosted by the **DAS XDRRequestService** service. The service operation uses the **ProvideAndRegisterDocumentSetRequest**. The **DAS\_Gateway\_ICD.docx** on the VACI web site contains an imbedded file named **XDRRequestService2.xsd** that contains the XSD definition. The Base64 Encoded format file created in step 5 is inserted in the **document** element that is contained in the **ProvideAndRegisterDocumentSetRequest** element

## Figure 6 – ProvideAndRegisterDocumentSetRequest Submit Form Example

This is a **ProvideAndRegisterDocumentSetRequest** submit form construct with the Base 64 Encoded element truncated for readability.

```
<urn:ProvideAndRegisterDocumentSetRequest>
  <urn1:SubmitObjectsRequest id="a36d9442-e184-40e6-8b96-70c22f83dce7" >
    <urn2:RequestSlotList>
      <urn3:Slot name="operationName" >
        <urn3:ValueList>
          <!--Zero or more repetitions:-->
          <urn3:Value>VSO.submitForm</urn3:Value>
        </urn3:ValueList>
      </urn3:Slot>
      <urn3:Slot name="originatingOrganizationName" >
        <urn3:ValueList>
          <urn3:Value>vso_sms_dev</urn3:Value>
        </urn3:ValueList>
      </urn3:Slot>
      <urn3:Slot name="originatingApplicationName" >
        <urn3:ValueList>
          <urn3:Value>Submit Form</urn3:Value>
        </urn3:ValueList>
      </urn3:Slot>
    </urn2:RequestSlotList>
  </urn1:SubmitObjectsRequest>

  <urn:Document id="0003" >PHYx ... Z lc3RNZXXNzYWdlPg==</urn:Document>

</urn:ProvideAndRegisterDocumentSetRequest>
```

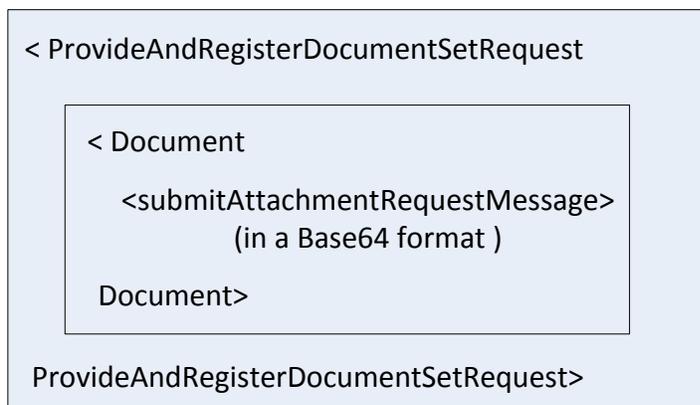
## 2.6. VSO to DAS to EFSS Submission Structure Submit Attachment Payload

The section contains conceptual diagrams and concrete examples of the VSO to DAS to EFSS submitAttachment payload submission structure.

### 2.6.1. VSO to DAS Submission Structure

The ProvideAndRegisterDocumentSetRequest element includes the Document element which contains the submitAttachmentRequestMessage in a Base 64 format.

**Figure 7 – ProvideAndRegisterDocumentSetRequest Conceptual Diagram**



**Figure 8 – ProvideAndRegisterDocumentSetRequest Example**

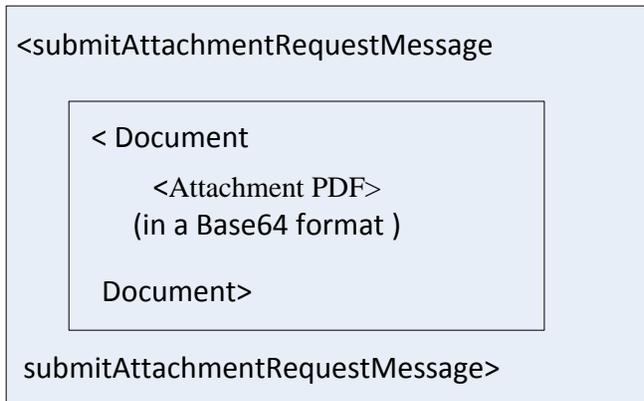
```
<urn:ProvideAndRegisterDocumentSetRequest>  
  <urn1:SubmitObjectsRequest id="a36d9442-e184-40e6-8b96-70c22f83dce7" >  
    ...  
    ...(Elements truncated for readability)  
    ...  
  </urn1:SubmitObjectsRequest>  
  
  <urn:Document id="0003" >PHYxOnN1Y...(Base64 payload truncated for readability)  
    ...  
    ...  
    ==</urn:Document>  
</urn:ProvideAndRegisterDocumentSetRequest>
```

## 2.6.2. DAS to EFSS Submission Structure

DAS will de-code the submitAttachmentRequestMessage prior the interaction with EFSS.

The submitFormRequestMessage element contains the Document element which contains the Form526EZ (in the case of 526EZ transmission) element which is in a Base 64 format.

**Figure 9 – submitAttachmentRequestMessage Conceptual Diagram**



**Figure 10 – submitAttachmentRequestMessage Example**

```
<v1:submitAttachmentRequestMessage xmlns:v1="http://viers.va.gov/efss/v1">
  ...
  <v1:sender>
    ...(Element truncated for readability)
  </v1:sender>
  <v1:attachmentInfo>
    ...
    <v1:Document id="0003" >PD94bWwg...(Base64 payload truncated for readability)
    ...
    ==</v1:Document>
    ...(Element truncated for readability)
  </v1:attachmentInfo>
</v1:submitAttachmentRequestMessage>
```

### 2.6.3. VSO to DAS Submit Attachment Payload Creation

This section will detail the step for the creation of the PDF attachment form payload that will be compatible with the DAS and EFSS WSDL and XSD specifications.

1. The attachment PDF file must be converted to a Base 64 Encoded format
2. The **EFSS X.X zip** file, where X.X is the current D2D release contained in the VACI web site, contains a file named **EFSSTypes.xsd**. This file contains the XSD template for the **submitAttachmentRequestMessage** element. Create and populate an XML file using this template. The Base64 Encoded format file created in step 1 is inserted in the **document** element that is contained in the **attachmentInfo** element
3. Convert the file created using the **submitAttachmentRequestMessage** template to a Base 64 Encoded format
4. The VSO submits a 21-526EZ payload with claim initiation information using the **ProvideAndRegisterDocumentSet-bRequest** operation hosted by the **DAS XDRRequestService** service. The service operation uses the **ProvideAndRegisterDocumentSetRequest**. The **DAS\_Gateway\_ICD.docx** on the VACI web site contains an imbedded file named **XDRRequestService2.xsd** that contains the XSD definition. The Base64 Encoded format file created in step 3 is inserted in the **document** element that is contained in the **ProvideAndRegisterDocumentSetRequest** element

## Figure 11 – ProvideAndRegisterDocumentSetRequest Submit Attachment Example

This is a **ProvideAndRegisterDocumentSetRequest** submit attachment construct with the Base 64 Encoded element truncated for readability.

```
<soapenv:Envelope xmlns:v1="http://viers.va.gov/efss/v1"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:urn="urn:ihe:iti:xds-b:2007"
  xmlns:urn1="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
  xmlns:urn2="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
  xmlns:urn3="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:ProvideAndRegisterDocumentSetRequest>
      <urn1:SubmitObjectsRequest id="a36d9442-e184-40e6-8b96-70c22f83dce7" >
        <urn2:RequestSlotList>
          <!--Zero or more repetitions:-->
          <urn3:Slot name="operationName" >
            <urn3:ValueList>
              <!--Zero or more repetitions:-->
              <urn3:Value>VSO.submitAttachment</urn3:Value>
            </urn3:ValueList>
          </urn3:Slot>
          <urn3:Slot name="originatingOrganizationName" >
            <urn3:ValueList>
              <!--Zero or more repetitions:-->
              <urn3:Value>vso_sms_dev</urn3:Value>
            </urn3:ValueList>
          </urn3:Slot>
          <urn3:Slot name="originatingApplicationName" >
            <urn3:ValueList>
              <!--Zero or more repetitions:-->
              <urn3:Value>Auto-CEST</urn3:Value>
            </urn3:ValueList>
          </urn3:Slot>
        </urn2:RequestSlotList>
      </urn1:SubmitObjectsRequest>

      <urn:Document id="0003" >PHYx ... Z lc3RNZXNzYWdlPg==</urn:Document>
    </urn:ProvideAndRegisterDocumentSetRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

## 3. DAS Service Operations Used By the VSO

This focus of this section is on the **DAS XDR Request Service** operations because DAS is the VSO exit and entry points for D2D. There are a number of sections in this document that references service operations in EFSS, but this was done to increase understanding.

### 3.1. ProvideAndRegisterDocumentSet-bRequest

The **ProvideAndRegisterDocumentSet-bRequest** operation uses the **ProvideAndRegisterDocumentSetRequest** message for communication from the VSO to DAS. There are four supported EFSS operation options that can be used in the **valueList** parameter contained in the **ProvideAndRegisterDocumentSetRequest** message. The values are as follows:

- VSO.submitForm
- VSO.submitAttachment
- VSO.checkStatus
- VSO.confirmSubmission

#### 3.1.1. VSO.submitForm Option

The **ProvideAndRegisterDocumentSetRequest** message contains a number of name/value pair elements, the element that controls the routing from DAS to EFSS component is shown in the following diagrams.

```
<urn3:Slot name="operationName" >
  <urn3:ValueList>
    <urn3:Value>VSO.submitForm</urn3:Value>
  </urn3:ValueList>
</urn3:Slot>
```

Imbedded in the **ProvideAndRegisterDocumentSetRequest** message is the Base64 encoded **submitFormRequestMessage**. Within the message is the **formType** element that identifies the appropriate EFSS framework form specific component that will be utilized.

```
<v1:formInfo>
  <v1:formType>21-526EZ</v1:formType>
  ...
  ...
</v1:formInfo>
```

### 3.1.2. VSO. submitAttachmentForm Option

The **ProvideAndRegisterDocumentSetRequest** message name/value pairs for the submitAttachmentForm follows:

```
<urn3:Slot name="operationName" >  
  <urn3:ValueList>  
    <urn3:Value> VSO.submitAttachmentForm</urn3:Value>  
  </urn3:ValueList>  
</urn3:Slot>
```

### 3.1.3. VSO. checkStatus Option

The **ProvideAndRegisterDocumentSetRequest** message name/value pair for the checkStatus follows:

```
<urn3:Slot name="operationName" >  
  <urn3:ValueList>  
    <urn3:Value> VSO.checkStatus</urn3:Value>  
  </urn3:ValueList>  
</urn3:Slot>
```

### 3.1.4. VSO. confirmSubmission option

The **ProvideAndRegisterDocumentSetRequest** message name/value pair for the confirmSubmission follows:

```
<urn3:Slot name="operationName" >  
  <urn3:ValueList>  
    <urn3:Value> VSO.confirmSubmission</urn3:Value>  
  </urn3:ValueList>  
</urn3:Slot>
```

## 3.2. ProvideAndRegisterDocumentSet-bResponse

### 3.2.1. EFSSResponseType

The **ProvideAndRegisterDocumentSet-bResponse** operation uses the **RegistryResponse** message for response communications from the EFSS to DAS and DAS to the VSO. The **RegistryResponse** message has a number of elements that are populated with pertinent response information. These elements are contained within the **EFSSResponseType** element which is contained in the **submitFormResponseMessage** element. The response information element follows:

```
<NS1:submitFormResponseMessage xmlns:NS1="http://viers.va.gov/efss/v1">
  ...
  ...
  <NS1:EFSSResponseType>
    <NS1:status>Success</NS1:status>
    <NS1:code>Success</NS1:code>
    <NS1:value> Form has been submitted successfully >
  </NS1:EFSSResponseType>
</NS1:submitFormResponseMessage>
```

### 3.2.2. VSO.submitForm-response

The **RegistryResponse** message contains a number of name/value pair elements. The element that defines an acknowledgement of the processing performed a result of a VSO.submitForm is shown in the following diagram.

```
<Slot xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0" name="operationName">
<ValueList>
<Value>VSO.submitForm-response</Value>
</ValueList>
</Slot>
```

### 3.2.3. VSO.submitForm-response

The **RegistryResponse** message contains a number of name/value pair elements. The element that defines an acknowledgement of the processing performed a result of a VSO.submitAttachmentForm is shown in the following diagram.

```
<Slot xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0" name="operationName">
<ValueList>
<Value>VSO.submitAttachmentForm-response</Value>
</ValueList>
</Slot>
```

## 4. Service Protocol

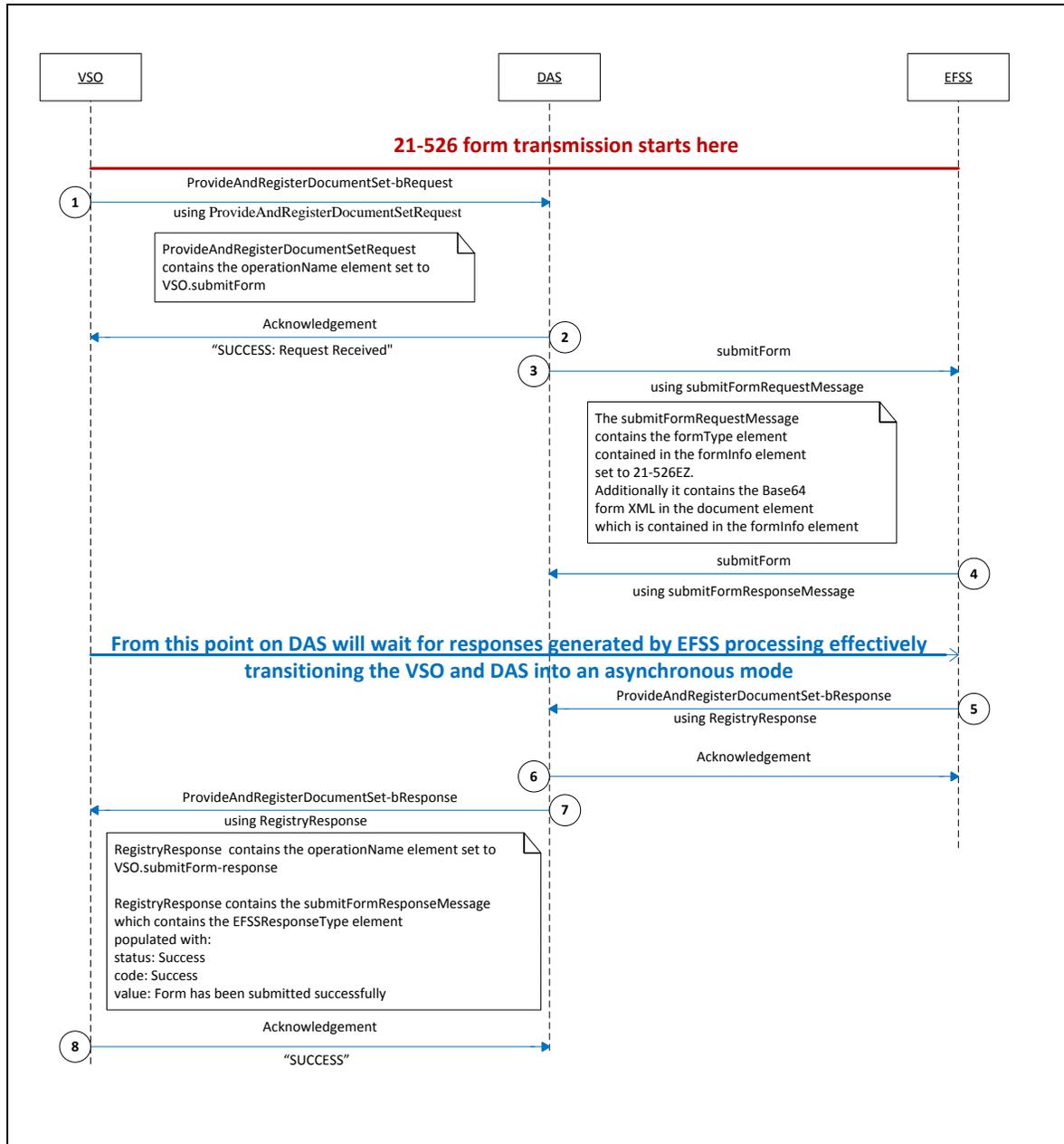
The service protocol is defined using the 21-526EZ form as an example. The 21-526EZ form processing from a form transmittal perspective is basically the same for all forms. The only difference is the makeup and creation of the payload for each specific form.

Section **2.2 Transaction Lexicon** defined the relationship between Submissions, Transmissions, Forms and Attachments. The submitForm invocation is a transmission that is part of a submission. This submission will become complete when all the submitAttachment transmission(s) occur.

The submitForm and submitAttachment transmissions are logically part of a single submission but they are physically separate transmissions.

## 4.1. Form Transmission Protocol

Figure 12 - 21-526 Submit Form Submission Sequence Diagram



1. A VSO submits a 21-526EZ payload with claim initiation information using the **ProvideAndRegisterDocumentSet-bRequest** operation hosted by the **DAS XDRRequestService** service. The service operation uses the **ProvideAndRegisterDocumentSetRequest** message which contains among other data elements; the Base64 encoded 21-526 XML payload
2. DAS will acknowledge to the VSO by returning an **Acknowledgement** response element populated with "SUCCESS: Request Received"

3. DAS will continue the operation using the EFSS **submitForm** operation hosted by the **EFSSService** request service. The service operation uses the **submitFormRequestMessage** which is replicated in the DAS **ProvideAndRegisterDocumentSetbResponseRequest** message.
4. EFSS will respond back to DAS using the using **submitFormResponseMessage**.
5. EFSS will also respond back to DAS when it's processing EFSS will also respond back to DAS when it's processing (successful/unsuccessful) has completed using the **ProvideAndRegisterDocumentSet-bResponse** operation hosted by the **DAS XDRRequestService** service the using the **RegistryResponse** message
6. EFSS will acknowledge to DAS by returning an **Acknowledgement** response element populated with "SUCCESS: Request Received"
7. DAS will forward the **ProvideAndRegisterDocumentSetbResponseRequest** message sent by EFSS to the VSO. It will use the **ProvideAndRegisterDocumentSet-bResponse** operation hosted by the VSO's **DAS XDRResponse Service** implementation. The message contains among other data elements, the Form Submission completion data:
  - status: Success
  - code: Success
  - value: Form has been submitted successfully
8. The VSO will acknowledge back to DAS by returning an **Acknowledgement** response element populated with "SUCCESS"

**NOTE:** Some of the preceding bullet items define interaction behavior between DAS and EFSS. These operations are not visible from a VSO perspective. It was added for clarity as it is referenced in the above 21-256EZ submitForm Transmission sequence diagram,

A detail explanation of the **ProvideAndRegisterDocumentSetRequest** and **ProvideAndRegisterDocumentSetbResponseRequest** message elements can be found in the **D2D\_VSO\_Interface\_Dictionary Updates Inc1ItX.X2.3.xlsx** spreadsheet where X.X is the current D2D release contained in the VACI web site.

A detailed explanation of the **submitFormRequestMessage** element can be found in the **D2DDataDictionary VRM TI Updates Inc1ItX.X.xlsx** spreadsheet where X.X is the current D2D release contained in the VACI web site.

## 4.2. Attachment Submission Protocol

The following 21-526EZ sequence diagrams details the flow of submitAttachment processing.

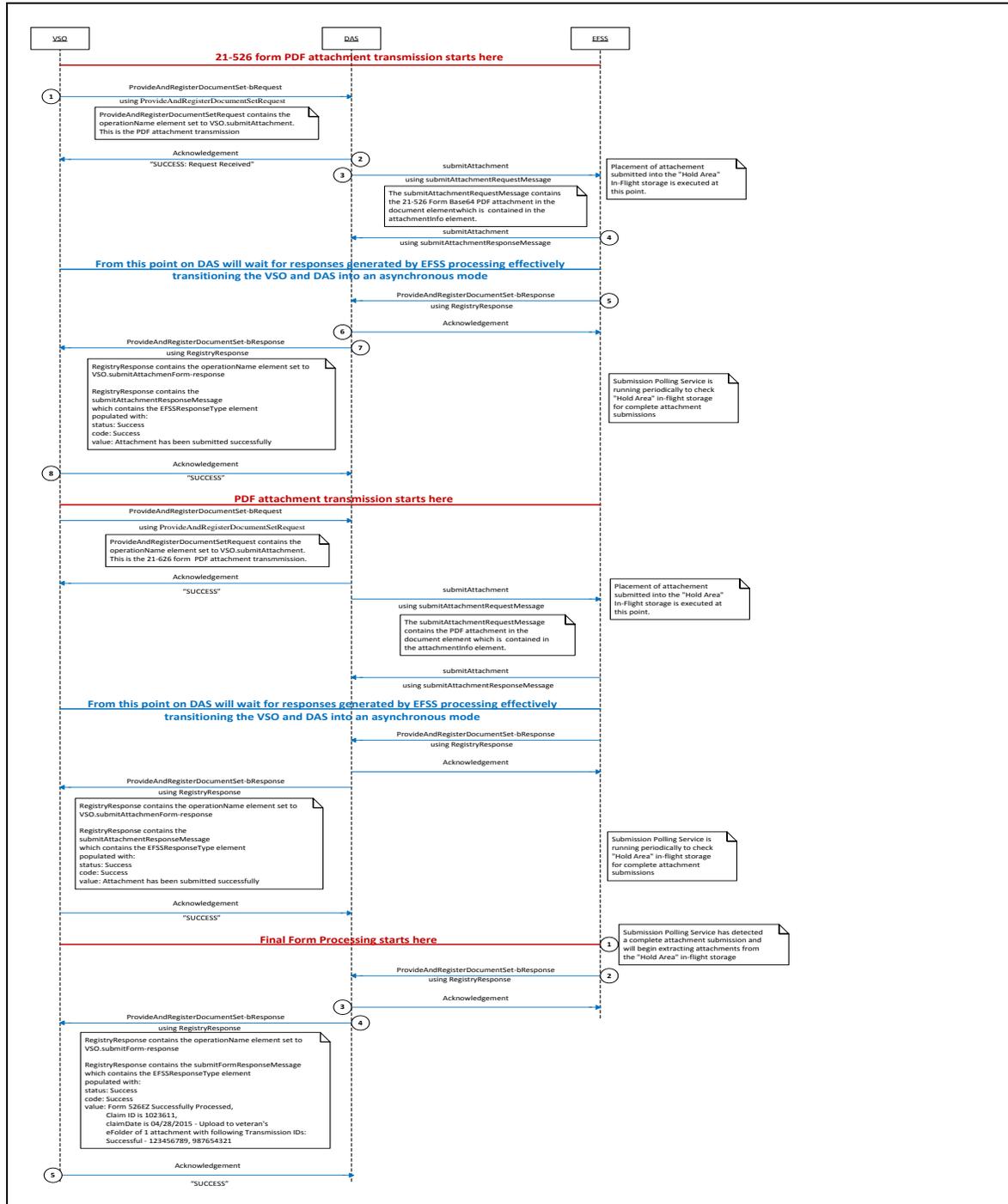
- 21-526 form PDF attachment transmission starts here
- PDF attachment transmission starts here
- Final Form Processing starts here

The elements will be detailed in the following sections:

- 0

- Form PDF Attachment Transmission Protocol
- 4.2.2 PDF Attachment Transmission Protocol
- 4.2.3 Final Form Processing Protocol

Figure 13 - 21-526 Submit Attachment Submission Sequence Diagram



#### 4.2.1. Form PDF Attachment Transmission Protocol

1. A VSO submits 21-526 Form Base64 PDF attachment payload using the **ProvideAndRegisterDocumentSet-bRequest** operation hosted by the **DAS XDRRequestService** service. The service operation uses the **ProvideAndRegisterDocumentSetRequest** message which contains the Base64 encoded 21-526 Form PDF attachment payload
2. DAS will acknowledge to the VSO by returning an **Acknowledgement** response element populated with "SUCCESS: Request Received"
3. DAS will continue the operation using the EFSS **submitAttachment** operation hosted by the **EFSS Request Service**. The service operation uses the **submitAttachmentRequestMessage**
4. EFSS will respond back to DAS using the using **submitAttachmentResponseMessage**.
5. EFSS will also respond back to DAS when it's processing (successful/unsuccessful) has completed using the **ProvideAndRegisterDocumentSet-bResponse** operation hosted by the **DAS XDRRequestService** service the using the **RegistryResponse** message
6. DAS will acknowledge to EFSS by returning an **Acknowledgement** response element populated with "SUCCESS"
7. DAS will forward the **ProvideAndRegisterDocumentSetbResponseRequest** message sent by EFSS to the VSO. It will use the **ProvideAndRegisterDocumentSet-bResponse** an operation hosted by the VSO's **DAS XDRResponse Service** implementation. The message contains among other data elements, the Form Submission completion data:
  - status: Success
  - code: Success
  - value: Attachment has been submitted successfully
8. The VSO will acknowledge to DAS by returning an **Acknowledgement** response element populated with "SUCCESS"

The **Submission Polling Service** will continue running periodically to check "hold area" in-flight storage for complete attachment submissions

**NOTE:** Some of the preceding bullet items define interaction behavior between DAS and EFSS. These operations are not visible from a VSO perspective. It was added for clarity as it is referenced in the above submitAttachment Transmission sequence diagram.

## 4.2.2. PDF Attachment Transmission Protocol

The processing for the PDF Attachment is exactly the same as described in section 4.2.2 **PDF Attachment Transmission Protocol** so this process will not be detailed here.

## 4.2.3. Final Form Processing Protocol

1. Submission Polling Service has detected a complete attachment submission and will begin extracting attachments from the "Hold Area" in-flight storage
  2. EFSS will respond to DAS with the **ProvideAndRegisterDocumentSet-bResponse** operation using the using the using **RegistryResponse**
  3. DAS will acknowledge to EFSS by returning an **Acknowledgement** response element populated with "SUCCESS"
  4. DAS will forward the **ProvideAndRegisterDocumentSetbResponseRequest** message sent by EFSS to the VSO. It will use the **ProvideAndRegisterDocumentSet-bResponse** operation hosted by the VSO's **DAS XDRResponse Service** implementation. The message contains among other data elements, the Form Submission completion data:
    - status: Success
    - code: Success
    - value: Form 526EZ Successfully Processed,  
Claim ID is 1023611,  
claimDate is 04/28/2015 - Upload to veteran's  
eFolder of 1 attachment with following Transmission IDs:  
Successful - 123456789, 987654321
  5. The VSO will acknowledge to DAS by returning an **Acknowledgement** response element populated with "SUCCESS"
- **NOTE:** Some of the preceding bullet items define interactions behavior between DAS and EFSS. These operations are not visible from a VSO perspective. It was added for clarity as it is referenced in the above submitAttachment Transmission sequence diagram

## 5. Service Scenarios

This section will provide various VSO to DAS interaction scenarios ranging from the happy path to recoverable to unrecoverable error situations. This document will consist of a subset of the 21-526-EZ sequence diagrams contained in section **4 Service Protocol**.

The transmission configuration of all of the following scenario examples will consist of the transmission of a 21-526EZ form and a single PDF which is the PDF version of the 21-526EZ form.

For purposed of simplification and clarity, the sequence diagrams used in this section will contain these operations only:

- **ProvideAndRegisterDocumentSet-bRequest** request operations sent from the VSO to DAS
- **ProvideAndRegisterDocumentSet-bResponse** response operations sent from the DAS to the VSO.

With an emphasis on the response operations sent from the DAS to the VSO.

Section **4 Service Protocol** supplies a more detail operation interaction view.

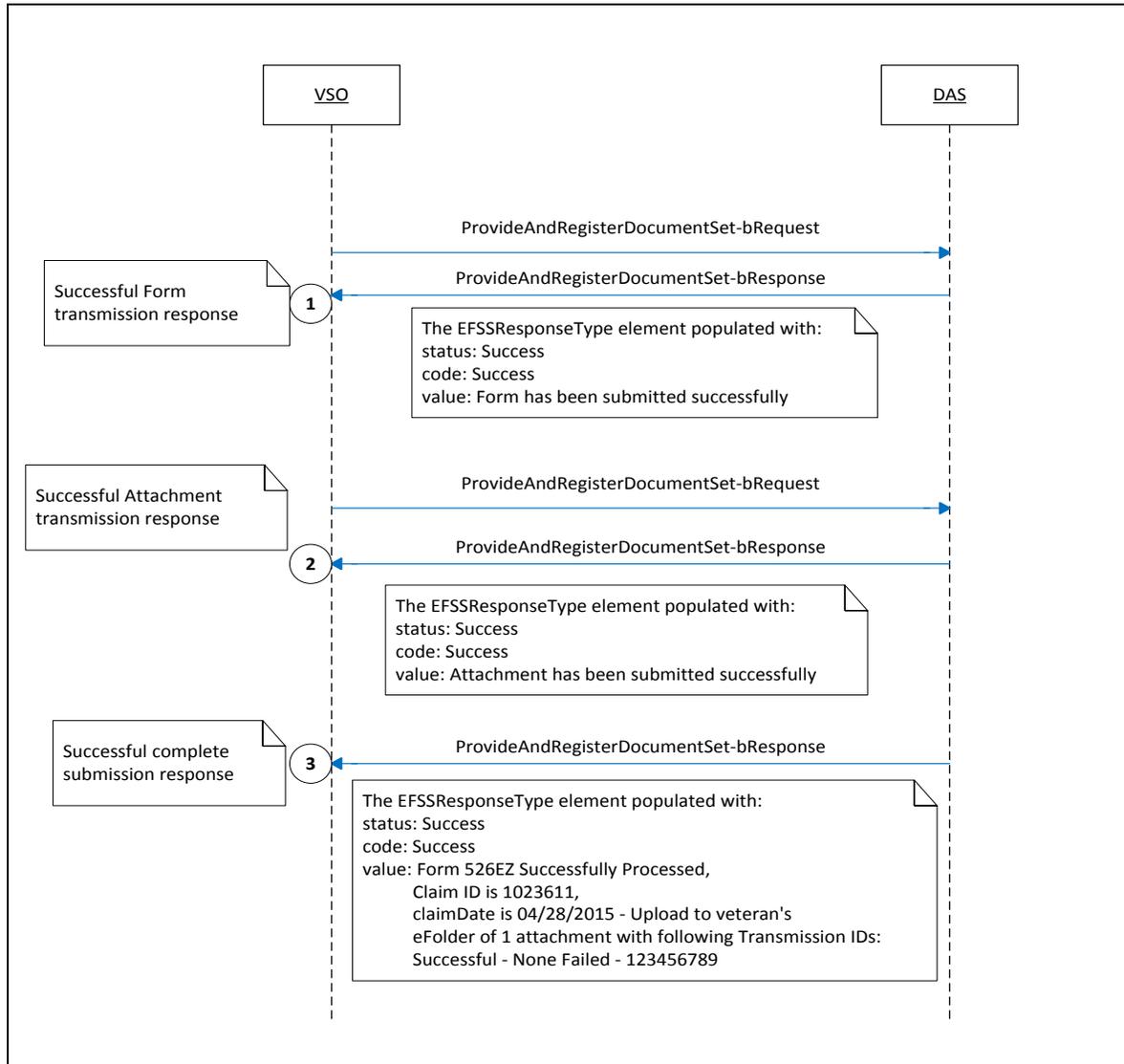
The scenario completion state will be categorized by:

- Successful
- Recoverable Failure
- Non-recoverable failure

## 5.1. Successful 21-526EZ submission

This scenario is an example of successful 526EZ submitForm and submitAttachments submission.

**Figure 14 – Successful 21-526EZ Form/Attachment Submission**



### Response Results:

1. Successful Form transmission
2. Successful Attachment transmission
3. Successful Complete Submission

### Completion State -:

Successful

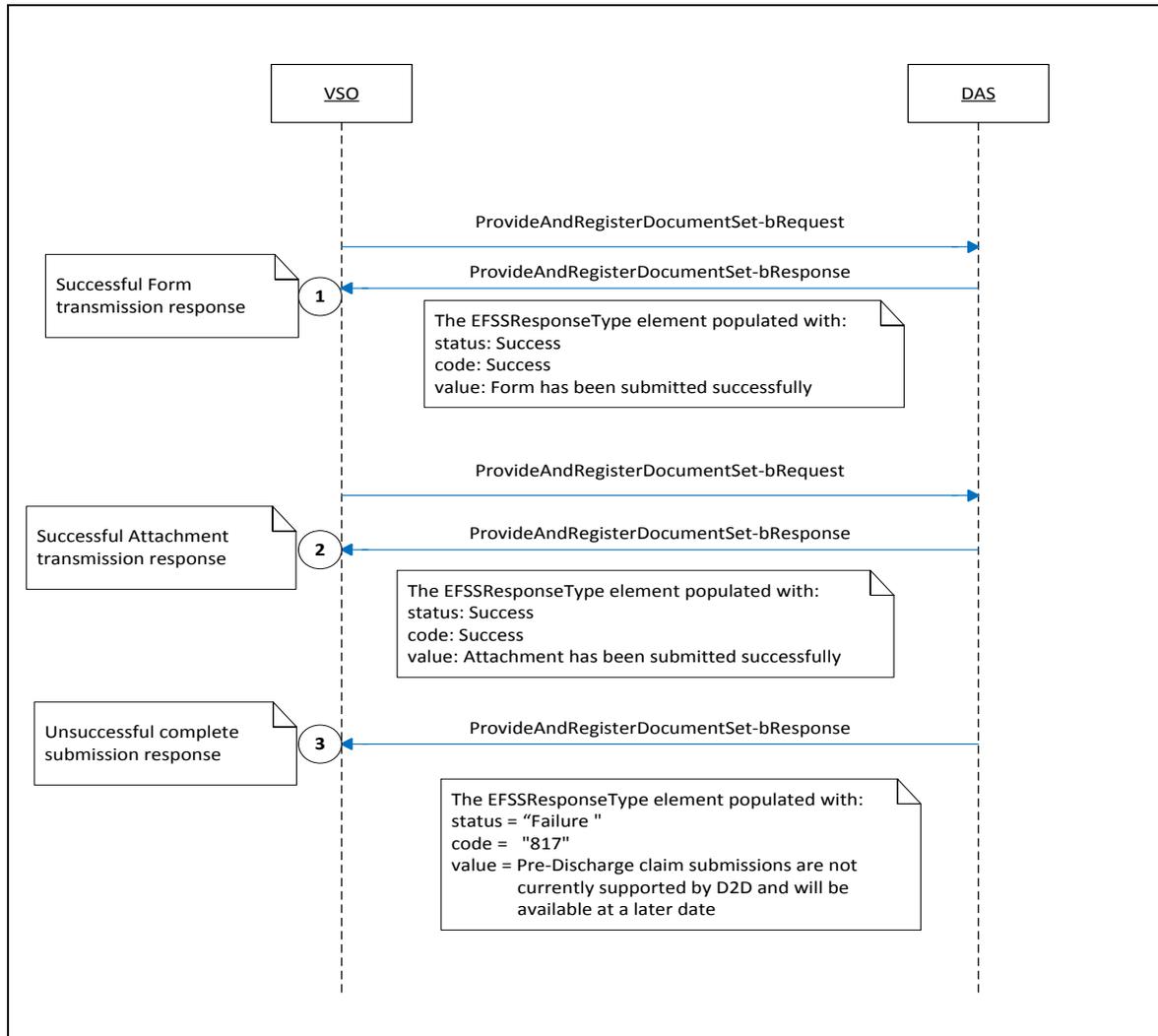
### Remediation Strategy:

Because this is a successful scenario remediation is not required

## 5.2. Unsuccessful 21-526EZ submission - Business Rule Failure

This scenario is an example of 526EZ submitForm and submitAttachments submission that fails the Pre-Discharge claim submission business rule.

**Figure 15 – Unsuccessful 21-526EZ Form/Attachment Submission**



### Response Results:

1. Successful Form transmission
2. Successful Attachment transmission
3. Unsuccessful Complete Submission

### Completion State

Unrecoverable:

### Remediation Strategy:

Because this is an unrecoverable violation of a business rule, remediation is not possible

### 5.3. Unsuccessful 21-526EZ Submission – Failure to Connect to DAS at Beginning of Submission

This scenario is an example of 526EZ submitForm and submitAttachments submission that fails to connect to DAS at beginning of submission.

**Figure 16 – Unsuccessful 21-526EZ Form/Attachment Submission**



#### **Response Results:**

1. No Response from DAS

#### **Completion State:**

Potentially recoverable with the late following mail option or an entire form and attachment resubmission

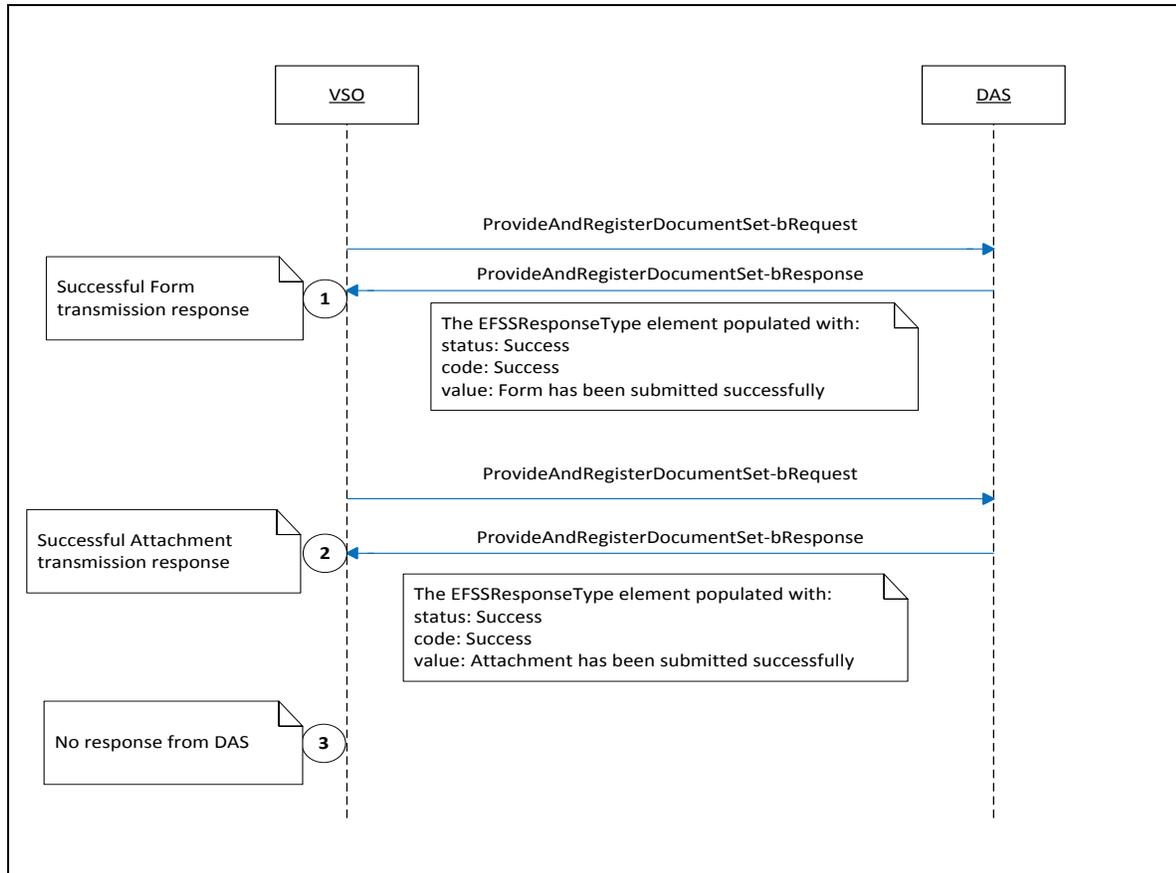
#### **Remediation Strategy:**

1. Validate that the correct DAS Request Service end point is being used
2. If end point is correct:
  - a. Submit the **ProvideAndRegisterDocumentSet-bRequest** operation to DAS using the **VSO.checkStatus** option
  - b. If the **VSO.checkStatus** option submission indicates that the transaction can be re-initiated, resubmit the attachment using the late following mail option
  - c. If the **VSO.checkStatus** option submission indicates that the transaction cannot be re-initiated, purge the submission by submitting the **ProvideAndRegisterDocumentSet-bRequest** operation using the **VSO.confirmSubmission** option then resubmit the entire form/attachment
3. If the resubmission steps above fails, utilize an alternative submission source (e.g., Centralized Mail)

## 5.4. Unsuccessful 21-526EZ Submission – DAS Response Failure during Submission Processing

This scenario is an example of 526EZ submitForm and submitAttachments submission that fails to receive a response from DAS in a time frame that is beyond the anticipated DAS response period.

**Figure 17 – Unsuccessful 21-526EZ Form/Attachment Submission**



### Response Results:

1. Successful Form transmission
2. Successful Attachment transmission
3. No Response from DAS

### Completion State:

Potentially recoverable with the late following mail option or an entire form and attachment resubmission

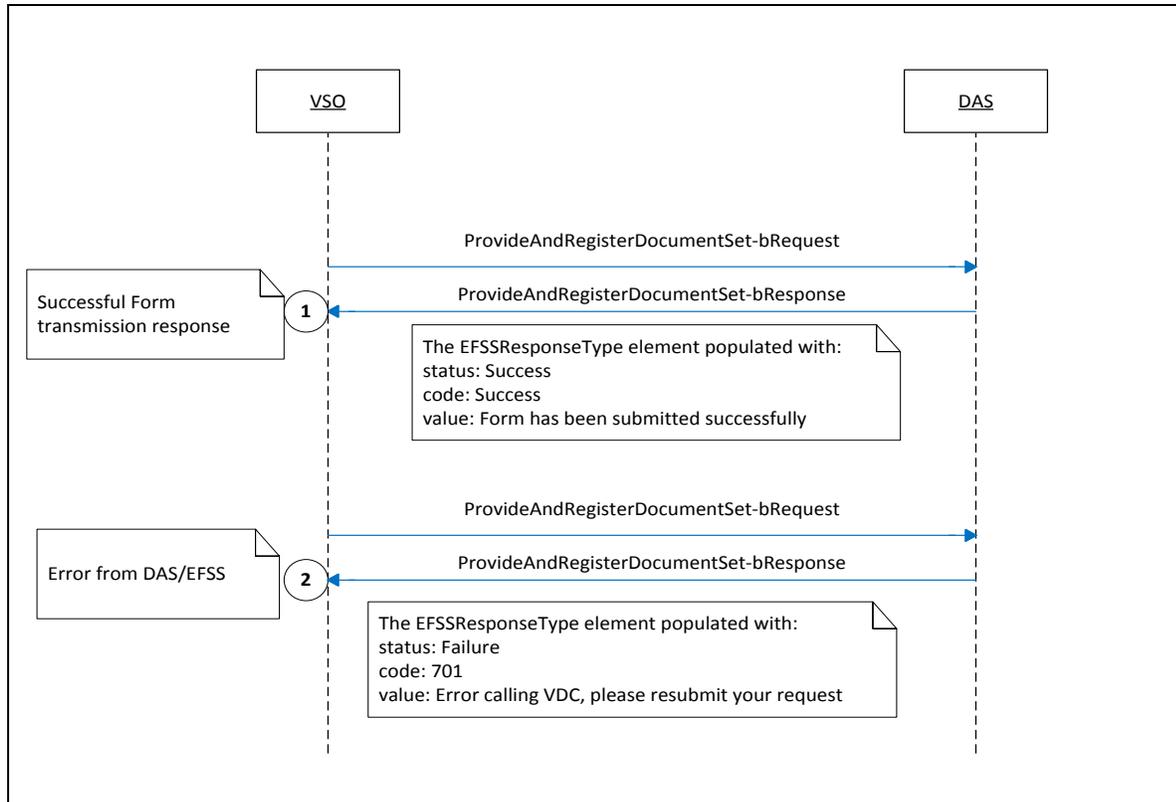
### Remediation Strategy:

1. The form was processed and the corresponding form attachment was placed in the in-flight storage “hold area”. This is validated by the form and attachment success response messages

2. Submit the **ProvideAndRegisterDocumentSet-bRequest** operation to DAS using the **VSO.checkStatus** option
3. If the **VSO.checkStatus** option submission indicates that the transaction can be re-initiated, resubmit the attachment using the late following mail option
4. If the **VSO.checkStatus** option submission indicates that the transaction cannot be re-initiated, purge the submission by submitting the **ProvideAndRegisterDocumentSet-bRequest** operation using the **VSO.confirmSubmission** option then resubmit the entire form/attachment
5. If the resubmission steps above fails, utilize an alternative submission source (e.g., Centralized Mail)

## 5.5. Unsuccessful 21-526EZ Submission - VDC Error

This scenario is an example of 526EZ submitForm and submitAttachments submission that encounters a VDC (in flight storage “hold area”) error.



### Response Results:

1. Successful Form transmission
2. Error from DAS/EFSS

### Completion State:

Potentially recoverable with the late following mail option or an entire form and attachment resubmission

### Remediation Strategy:

1. The form was processed. This is validated by the form success response messages
2. Submit the **ProvideAndRegisterDocumentSet-bRequest** operation to DAS using the **VSO.checkStatus** option
3. If the **VSO.checkStatus** option submission indicates that the transaction can be re-initiated, resubmit the attachment using the late following mail option
4. If the **VSO.checkStatus** option submission indicates that the transaction cannot be re-initiated, purge the submission by submitting the **ProvideAndRegisterDocumentSet-bRequest** operation using the **VSO.confirmSubmission** option then resubmit the entire form/attachment
5. If the resubmission steps above fails, utilize an alternative submission source (e.g., Centralized Mail)

## 5.6. Known Issues with the Messaging Back to the VSO

- The lack of a final validation response from EFSS to DAS signifying the successful completion of the 21-22 and the 21-0966

**Workaround:**

Query VBMS or SEP to determine the state of the 21-22 or 21-0966 of the submissions

- Lack of a response of a response from DAS that EFSS is non-responsive

**Workaround:**

Refer to section **5.4 Unsuccessful 21-526EZ Submission – DAS Response Failure during Submission Processing**

- Lack of a response of a response from EFSS to DAS that that a payload is greater than 25 Meg

**Workaround:**

This issue resolution is being worked internally within the VA. In the interim submit large attachments through centralized mail

- An intermittent error where a claim ID is generated, but the transmitted artifacts fail to be persisted in VBMS.

**Workaround:**

- An entire form and attachment resubmission is required