

Department of Veterans Affairs

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

Development Roadmap and Service Scenarios Document

Document Version 2.1

This Document is Consistent with D2D Release 5.0



November 2016

Revision History

Date	Version	Description	Author
11/29/2016	2.1	Added requirement to wait for receipt of the asynchronous submitFormResponseMessage before submitting attachments. Changes made to sections 1.3, 5.0, and 5.1.1 (4). Section 1.3: Maximum number of attachments changed from 50 to 100.	VRM TI
09/08/2016	2.0	Updated bases on lessons learned from the creation of the D2D Reference Implementation GuideLines v6.0 document.	VRM TI
03/1/2016	1.2	<ul style="list-style-type: none"> • Incorporate general suggestions from Panoramic • Documented the Additional Supporting Documents option in section 1.6.3 DAS to EFSS Submission Structure - Additional Supporting Documents Option and references to it. 	VRM TI
01/06/2016	1.1	Changes from review with Panoramic	VRM TI
12/08/2015	1.0	Finalized draft versions	VRM TI
11/12/2015	0.11	Incorporate suggestions 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 8 Service Scenarios	VRM TI
10/26/2015	0.6	Incorporate internal review changes	VRM TI
10/08/2015	0.5	Incorporate suggestions 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

Table of Contents

1. Introduction	1
1.1. Document Purpose.....	1
1.2. High Level D2D Overview.....	1
1.3. D2D Transaction Lexicon.....	3
1.4. Overview of VSO to DAS and Back to VSO Submission Structure	4
1.5. Overview of DAS to EFSS Submission Structure	5
1.6. VSO to DAS to EFSS Submission Structure Submit Form Payload.....	6
1.6.1. VSO to DAS Submission Structure	6
1.6.2. DAS to EFSS Submission Structure.....	7
1.6.3. DAS to EFSS Submission Structure - Additional Supporting Documents Option.....	8
1.6.4. VSO to DAS Submit Form Payload Creation	9
1.7. VSO to DAS to EFSS Submission Structure Submit Attachment Payload	10
1.7.1. VSO to DAS Submission Structure	10
1.7.2. DAS to EFSS Submission Structure.....	11
1.7.3. VSO to DAS Submit Attachment Payload Creation	12
2. XDRRequestService Operation Options Hosted by DAS Which Are Used by the VSO	13
2.1. ProvideAndRegisterDocumentSet-bRequest	13
2.1.1. VSO.submitForm Option	13
2.1.2. VSO.submitAttachmentForm Option.....	15
2.1.3. VSO.checkStatus Option.....	16
2.1.4. VSO.confirmSubmission option	16
2.2. ProvideAndRegisterDocumentSet-bResponse	17
2.2.1. EFSSResponseType	17
2.2.2. VSO.submitForm-response	17
2.2.3. VSO.submitAttachmentForm-response	17
3. XDRResponseService Operation Options Hosted by the VSO Which are Used by DAS	18
3.1. ProvideAndRegisterDocumentSet-bResponse using the RegistryResponse Message	18
3.1.1. VSO.submitForm-response Option	18
3.1.2. VSO.submitAttachmentForm-response Option.....	18
3.2. ProvideAndRegisterDocumentSet-bResponse using the Acknowledgement Message	18
4. DAS Request and Response Service Operations WSDLs	19
4.1. DAS XDRRequestService WSDL	19

4.2. DAS XDRResponse Service WSDL	19
5. Service Messaging Protocol using Sequence Diagrams	20
5.1. Form Transmission Protocol	22
5.1.1. Form Transmission Protocol	23
5.2. Attachment Submission Protocol	24
5.2.1. Form PDF Attachment Transmission Protocol.....	26
5.2.2. Supporting Document PDF Attachment Transmission Protocol	27
5.2.3. Final Form Processing Protocol.....	27
6. Service Messaging Protocol using Component/Collaboration Diagrams.....	28
6.1. Submit Form Service Messaging Protocol	29
6.2. Submit Attachment Service Messaging Protocol.....	30
7. Service Protocol Payload Examples	31
7.1. Form Transmission Protocol Example.....	31
7.2. Attachment Submission Protocol Example.....	32
7.2.1. Form PDF Attachment Transmission Protocol Example	32
7.2.2. PDF Attachment Transmission Protocol Example	33
7.2.3. Final Form Processing Protocol Example	34
8. Service Scenarios	35
8.1. Successful 21-526EZ submission	36
8.2. Unsuccessful 21-526EZ submission - Business Rule Failure	37
8.3. Unsuccessful 21-526EZ Submission – Failure to Connect to DAS at Beginning of Submission.....	38
8.4. Unsuccessful 21-526EZ Submission – DAS Response Failure during Submission Processing	39
8.5. Unsuccessful 21-526EZ Submission - VDC Error.....	41

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 – submitFormRequestMessage with the Additional Supporting Documents Option Example	8
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 Form Example	14
Figure 12 – ProvideAndRegisterDocumentSetRequest Submit Attachment Example	15
Figure 13 - 21-526 Submit Form Submission Sequence Diagram	22
Figure 14 - 21-526 Submit Attachment Submission Sequence Diagram	25
Figure 15 - Submit Form Component/Collaboration Diagram	29
Figure 16 - Submit Attachment Component/Collaboration Diagram	30
Figure 17 – Successful 21-526EZ Form/Attachment Submission	36
Figure 18 – Unsuccessful 21-526EZ Form/Attachment Submission	37
Figure 19 – Unsuccessful 21-526EZ Form/Attachment Submission	38
Figure 20 – Unsuccessful 21-526EZ Form/Attachment Submission	39

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:

- High Level D2D overview including
 - High level D2D component description
 - D2D Transaction Lexicon
 - Overview of how to structure payloads for Direct Access Services (DAS) and VIERS Electronic Form Submission (EFSS) Service.
- DAS Request operation options
- DAS Response operation options
- Das Request and Response operation WSDLs
- Service Messaging Protocol using Sequence Diagrams
- Service Messaging Protocol using Component/Collaboration Diagrams
- Service protocol payload examples
- Service scenario procedures to be followed under various circumstances

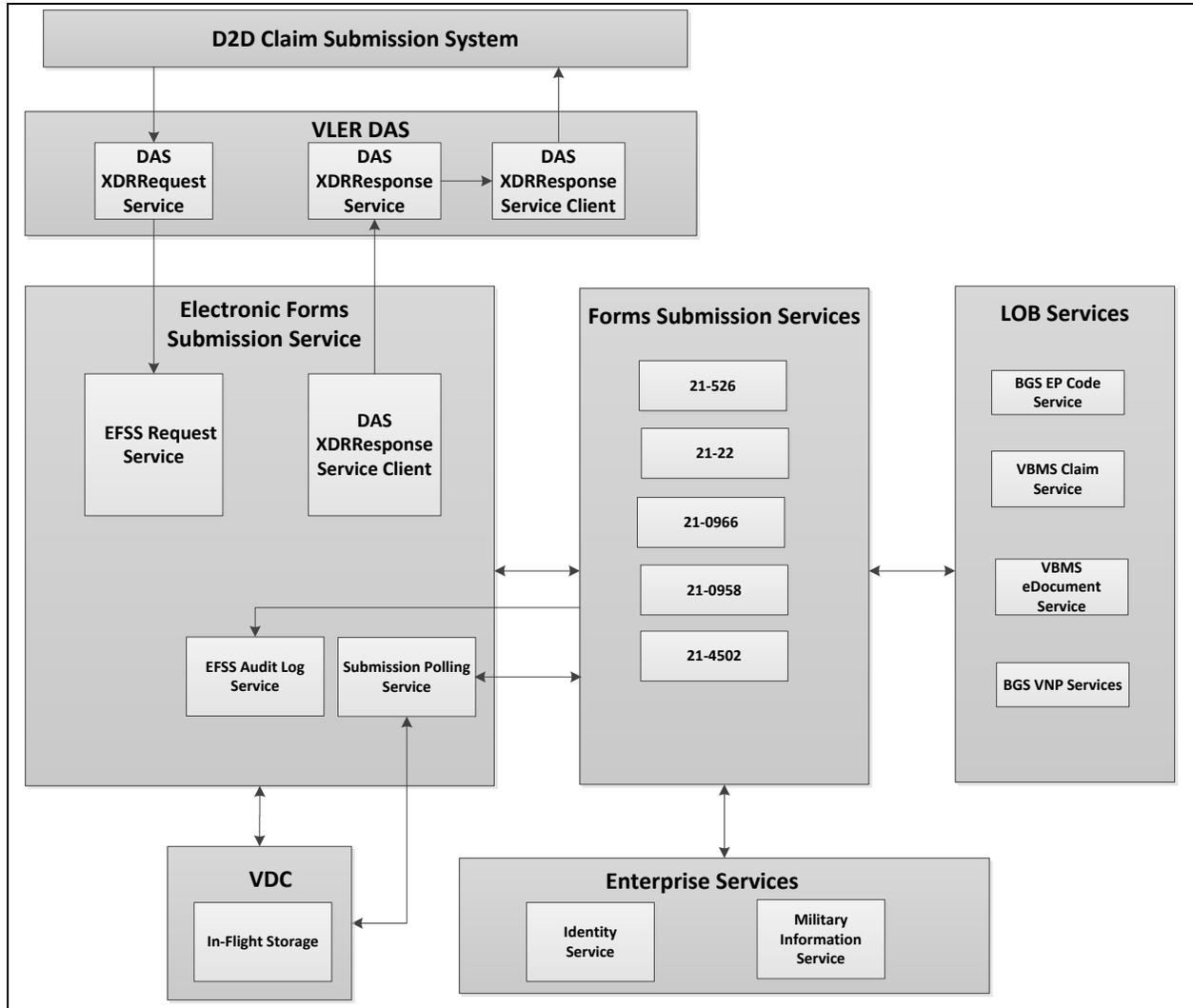
1.2. High Level D2D Overview

DAS and EFSS services enable an external VSO CMS to submit claims and forms via a SOAP submission to the VA for electronic processing. This will eliminate paper records and speed up processing time for the Veteran. The CMS will submit SOAP packages to the DAS gateway resulting in the 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 frameworks.
 - EFSS – Implementation of the common functionality of the framework. The EFSS 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”
- Line of Business (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.



1.3. D2D 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 ID within a single Submission will have a unique ID value
 - 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 a specific form or a non-form attachment
 - Maximum attachment size is 25 MEG
 - The maximum number of attachments is currently set to 100 attachments per submission
- Manifest – The Manifest is used to determine the completeness of attachment transmissions using the numberOfDocuments value versus the actual number attachments transmitted.
- Submission Sequence – When submitting forms with attachments, the VSO must wait for the receipt of the VIERS EFSS asynchronous submitFormResponseMessage, which is contained in the RegistryResponse DAS message, forwarded through DAS before submitting attachments to assure that the Manifest will be open to receive the attachments.

1.4. Overview of VSO to DAS and Back to VSO Submission Structure

This section details the VSO to DAS back to VSO web service submission structure. Detailed descriptions of payload packaging will be included in sections **1.6.4 VSO to DAS Submit Form Payload Creation** and **1.7.3 VSO to DAS Submit Attachment Payload Creation**

Table 1: VSO to DAS Submission Structure

Service	Operation	Description
XDRRequestService	ProvideAndRegisterDocumentSet-bRequest	The VSO to DAS service request transmits the form or attachment using the ProvideAndRegisterDocumentSetRequest message and responds using the Acknowledgement message.

Table 2: DAS to VSO Submission Structure

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

1.5. 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 specifically because the VSO must populate the `submitFormRequestMessage`, the `submitAttachmentRequestMessage`, the `EFSSConfirmationType` and the `EFSSStatusType` messages which are imbedded in the DAS submission payload.

Table 3: DAS to EFSS Form and Attachment Submission Structure

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

Table 4: DAS to EFSS Status and Confirmation Submission Structure

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

1.6. 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.

1.6.1. VSO to DAS Submission Structure

The ProvideAndRegisterDocumentSetRequest message 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>
```

1.6.2. DAS to EFSS Submission Structure

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

Figure 4 – submitFormRequestMessage Conceptual Diagram

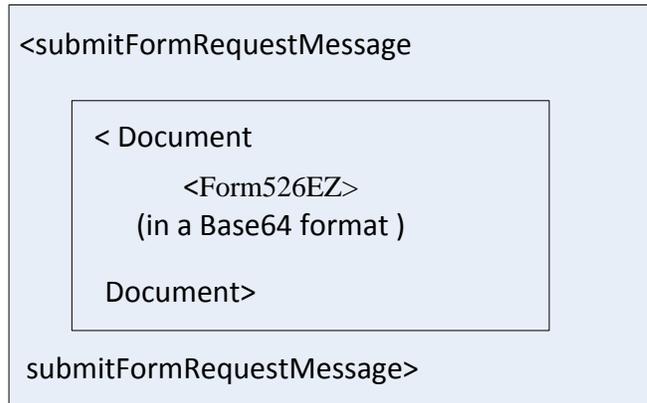


Figure 5 – submitFormRequestMessage Example

```
<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>
```

1.6.3. DAS to EFSS Submission Structure - Additional Supporting Documents Option

The Form526EZ submission or any form that contains supporting documentation attachments can be utilized to submit additional attachments to an existing claim in pending status. The process is the same as the initial form submission except a number of name/value parameters must be set in the submitFormRequestMessage.

The EFSSManifestItem element contains an unbound Name/Value construct named EFSSClaimInfo.

To initiate the Additional Supporting Documents processing the Name/Value string constructs contained in the EFSSClaimInfo must be populated as follows:

- Name = FORM_SUBTYPE Value = LATE
- Name = CLAIM_ID Value = *Claim ID of the claim where additional attachments are needed*
- Name = CLAIM_DATE Value = *Submission date of the claim where additional attachments are needed*

Figure 6 – submitFormRequestMessage with the Additional Supporting Documents Option 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>2</v1:numberOfDocuments>
    ...
    <v1:Document id="0003" >PD94bWwg...(Base64 payload truncated for readability)</v1:Document>
    ...
  </v1:formInfo>
    (Element truncated for readability)
    ...
    <v1:EFSSManifestType>
      <v1:ManifestItem>
        ...
        <v1: name> FORM_SUBTYPE</v1: name>
        <v1: value> LATE </v1: value>
        <v1: name> CLAIM_ID </v1: name>
        <v1: value> 123456789 </v1: value>
        <v1: name> CLAIM_DATE </v1: name>
        <v1: value> 12/31/2015 </v1: value>
        ...
        (Element truncated for readability)
      </v1:EFSSManifestType>
    </v1:formInfo>
  </v1:submitFormRequestMessage>
```

1.6.4. 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** message. 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** message.

1.7. 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.

1.7.1. 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

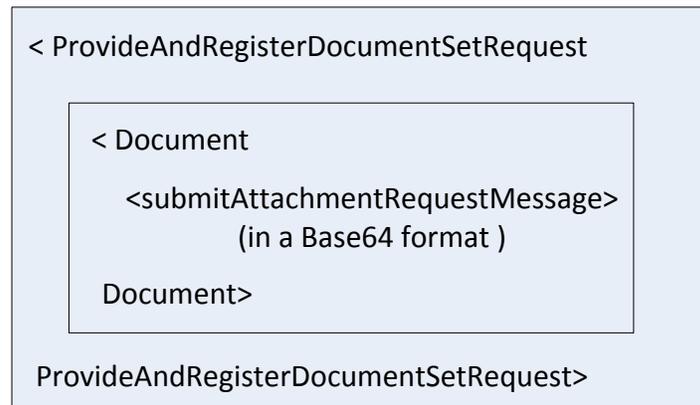


Figure 8 – ProvideAndRegisterDocumentSetRequest Example

```
<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>
```

1.7.2. DAS to EFSS Submission Structure

The submitAttachmentRequestMessage element contains the Document element which contains a PDF version of the Form526EZ (in the case of 526EZ transmission), another form or a supporting document attachment all of which are in a Base 64 format.

Figure 9 – submitAttachmentRequestMessage Conceptual Diagram

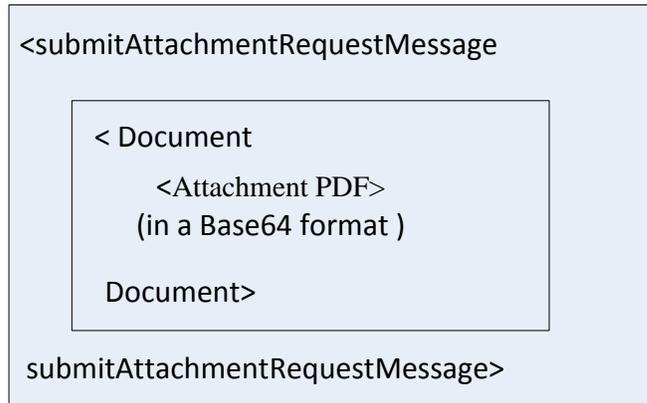


Figure 10 – submitAttachmentRequestMessage Example

```
<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>
```

1.7.3. 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** message. 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** message.

2. XDRRequestService Operation Options Hosted by DAS Which Are Used by the VSO

This focus of this section is on the **DAS XDRRequestService** options, hosted in DAS that are invoked by the VSO.

2.1. 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

2.1.1. 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** message. Within the message is the **formType** element that identifies the appropriate FSS framework form specific component that will be utilized.

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

Figure 11 – ProvideAndRegisterDocumentSetRequest Submit Form Example

```
<urn:ProvideAndRegisterDocumentSetRequest>
  <urn1:SubmitObjectsRequest id="a36d9442-e184-40e6-8b96-70c22f83dce7" >
    <urn2:RequestSlotList>
      <urn3:Slot name="operationName" >
        <urn3:ValueList>
          <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 lc3RNZXNzYWdlPg==</urn:Document>

</urn:ProvideAndRegisterDocumentSetRequest>
```

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

2.1.2. 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>
```

Figure 12 – ProvideAndRegisterDocumentSetRequest Submit Attachment Example

```
<urn:ProvideAndRegisterDocumentSetRequest>
  <urn1:SubmitObjectsRequest id="a36d9442-e184-40e6-8b96-70c22f83dce7" >
    <urn2:RequestSlotList>
      <!--Zero or more repetitions:-->
      <urn3:Slot name="operationName" >
        <urn3:ValueList>
          <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>
```

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

2.1.3. 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>
```

2.1.4. 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>
```

2.2. ProvideAndRegisterDocumentSet-bResponse

2.2.1. 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>
```

2.2.2. 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>
```

2.2.3. VSO.submitAttachmentForm-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>
```

3. XDRResponseService Operation Options Hosted by the VSO Which are Used by DAS

This focus of this section is on the **DAS XDRResponseService** operation options, hosted by the VSO, that are invoked by DAS.

3.1. ProvideAndRegisterDocumentSet-bResponse using the RegistryResponse Message

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

- VSO.submitForm-response
- VSO.submitAttachmentForm-response

3.1.1. VSO.submitForm-response Option

The **RegistryRequest** message contains a number of name/value pair elements, the element that defines a submit form response 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>
```

3.1.2. VSO.submitAttachmentForm-response Option

The **RegistryRequest** message contains a number of name/value pair elements, the element that defines a submit attachment form response 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>
```

3.2. ProvideAndRegisterDocumentSet-bResponse using the Acknowledgement Message

The **ProvideAndRegisterDocumentSet-bResponseResponse** operation uses the **Acknowledgement** message for communication from DAS to the VSO. The **Acknowledgement** message is shown in the following diagram.

```
<Acknowledgement>
  <message>SUCCESS </message>
</Acknowledgement>
```

4. DAS Request and Response Service Operations WSDLs

Section 2 **XDRRequestService Operation Options Hosted by DAS Which Are Used by the VSO** and section 3 **XDRResponseService Operation Options Hosted by the VSO Which are Used by DAS**.

The DAS Request and Response Services each contain two WSDL files with the same name except one is suffixed with a 0. The 0 suffixed WSDL contains the Message and Port Type elements which describe the service operations and operation elements. These two files are included in this section.

4.1. DAS XDRRequestService WSDL



XDRRequestService0
.wsdl

4.2. DAS XDRResponse Service WSDL



XDRResponseService
0.wsdl

5. Service Messaging Protocol using Sequence Diagrams

The service messaging 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

D2D Transaction LexiconSection **1.3 D2D 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.

The communication link between DAS and VIERS EFSS is multi-threaded; since the form and each attachment are sent as individual transmissions it is possible for the attachment(s) to be received by the EFSS before the form which will result in an error response because the Manifest has not been opened for the form. To avoid this problem, the VSO sender must wait for the receipt of the asynchronous submitFormResponseMessage, which is contained in the RegistryResponse DAS message, before submitting attachments.

5.1. Form Transmission Protocol

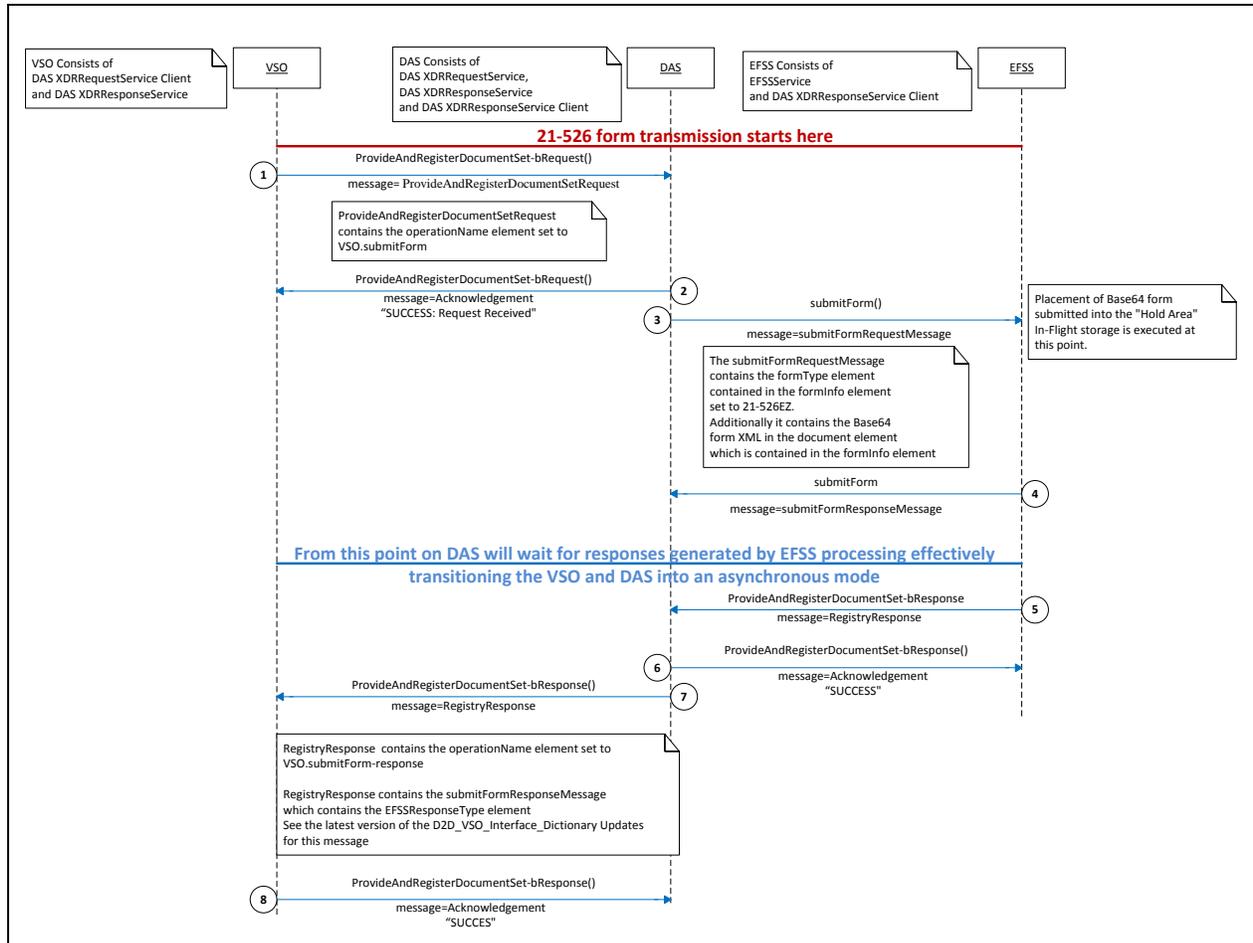
The following 21-526EZ sequence diagrams details the flow of submitForm processing.

- 21-526 form PDF attachment transmission starts here
- DAS will wait for responses generated by EFSS processing

The elements will be detailed in the following section:

- **5.1.1 Form Transmission Protocol**

Figure 13 - 21-526 Submit Form Submission Sequence Diagram



5.1.1. Form Transmission Protocol

1. **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 respond to the VSO by returning an **Acknowledgement** response message populated with "SUCCESS: Request Received"
3. DAS will continue the operation using the EFSS **submitForm** operation hosted by the **EFSService** service. The service operation uses the **submitFormRequestMessage** which is replicated in the DAS **ProvideAndRegisterDocumentSetRequest** message.
4. EFSS will respond back to DAS using the using **submitFormResponseMessage** message. The VSO must wait for this response forwarded through DAS (see item 7 below) before submitting attachments to assure that the Manifest will be open to receive the attachments.
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. EFSS will acknowledge to DAS by returning an **Acknowledgement** response element populated with "SUCCESS: Request Received"
7. DAS will forward the **RegistryResponse** 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: See the latest version of the **D2D_VSO_Interface_Dictionary Updates** for this message.
8. The VSO will respond to DAS by returning an **Acknowledgement** response message 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 **RegistryResponse** message elements can be found in the **D2D_VSO_Interface_Dictionary Updates Inc3It5.0.xlsx** spreadsheet where 5.0 is the current D2D release contained in the VACI web site.

A detailed explanation of the **submitFormRequestMessage** element can also be found in this **D2D_VSO_Interface_Dictionary** spreadsheet.

5.2. Attachment Submission Protocol

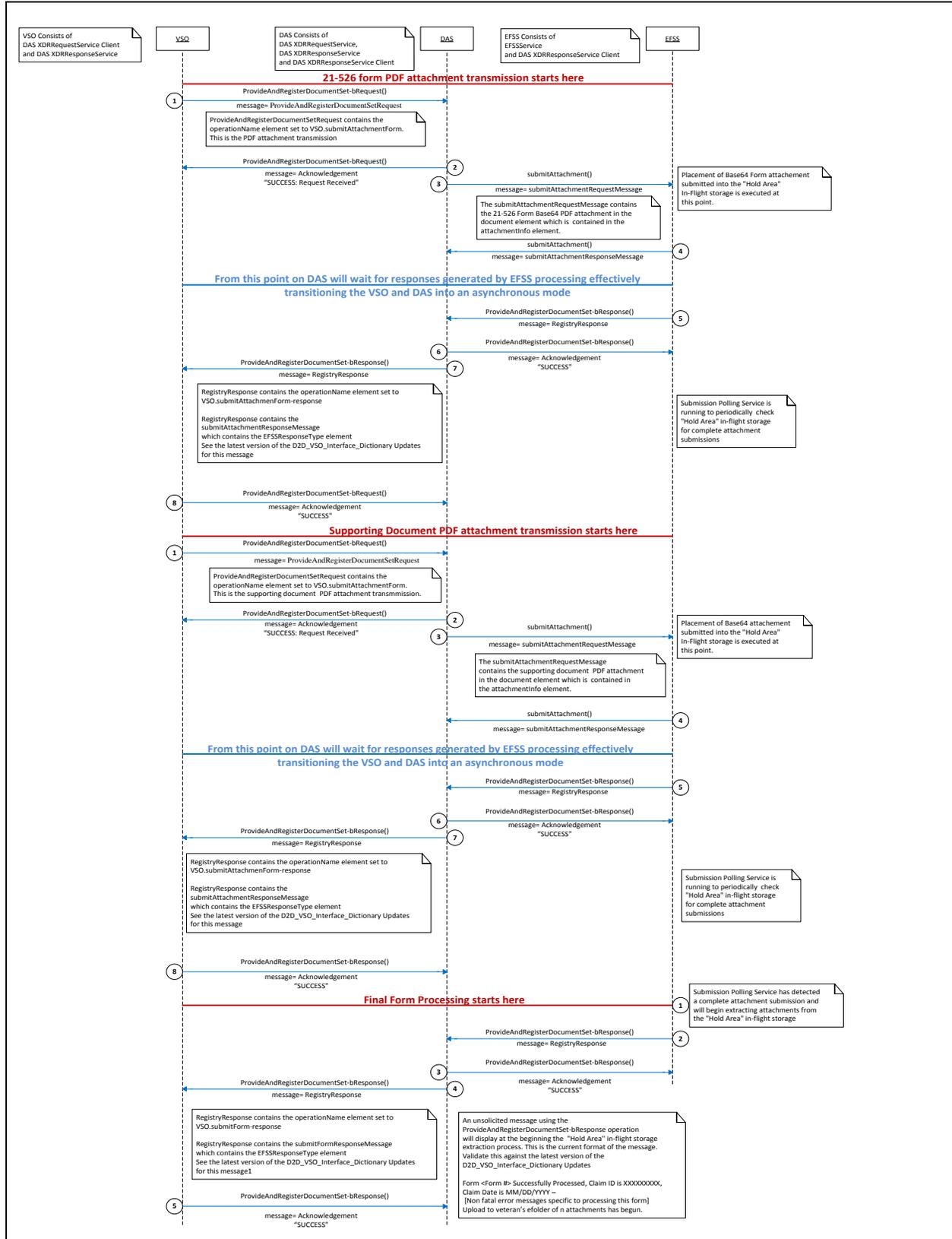
The following 21-526EZ sequence diagrams details the flow of submitAttachment processing.

- 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:

- **5.2.1 Form PDF Attachment Transmission Protocol**
- **5.2.2 Supporting Document PDF Attachment Transmission Protocol**
- **5.2.3 Final Form Processing Protocol**

Figure 14 - 21-526 Submit Attachment Submission Sequence Diagram



5.2.1. 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 respond to the VSO by returning an **Acknowledgement** response message populated with "SUCCESS: Request Received"
3. DAS will continue the operation using the EFSS **submitAttachment** operation hosted by the **EFSSService** service. The service operation uses the **submitAttachmentRequestMessage** message.
4. EFSS will respond back to DAS using the using the **submitAttachmentResponseMessage** message.
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 message populated with "SUCCESS"
7. DAS will forward the **RegistryResponse 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:
See the latest version of the D2D_VSO_Interface_Dictionary Updates for this message
8. The VSO will respond to DAS by returning an **Acknowledgement** response message 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.

5.2.2. Supporting Document PDF Attachment Transmission Protocol

The processing for the PDF Attachment is exactly the same as described in section **5.2.1 Form PDF Attachment Transmission Protocol** so this process will not be detailed here.

5.2.3. 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 **RegistryResponse** 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:
See the latest version of the D2D_VSO_Interface_Dictionary Updates for this message
 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

6. Service Messaging Protocol using Component/Collaboration Diagrams

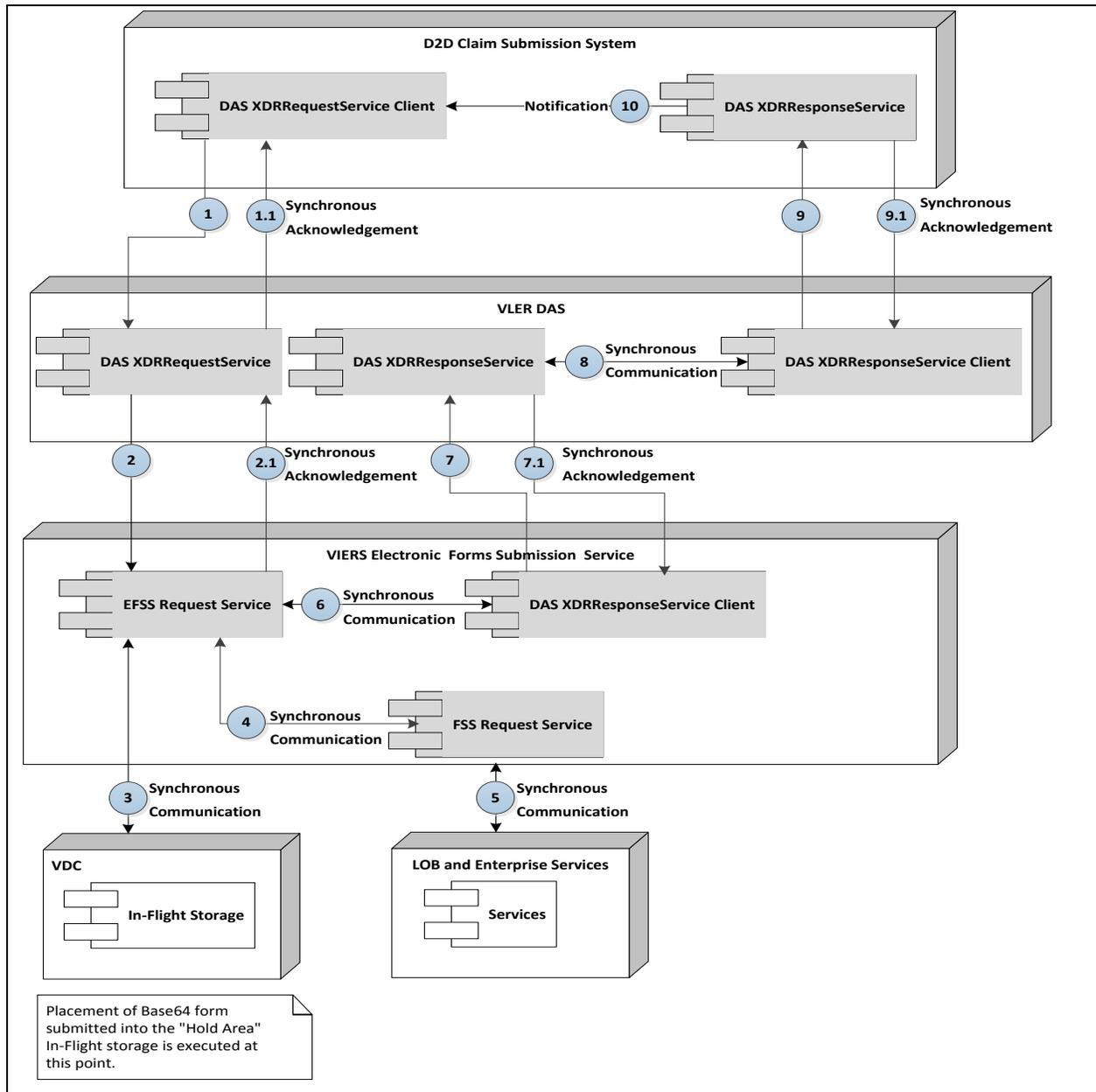
Section 5 **Service Messaging** Protocol defined the service protocol from a Sequence Diagram perspective. This section will define the Service Messaging Protocol using a combination of a component and a collaboration diagram.

This section will document the submitForm and submitFormAttachment operations using the combination Component/Collaboration diagram

6.1. Submit Form Service Messaging Protocol

The sequence of component interactions is defined by the numbered circles on the diagram. Numbers with a .1 reference (e.g. 1.1) signify a synchronous communication between layers on the component stack. This was done to highlight the intra-component stack touch points.

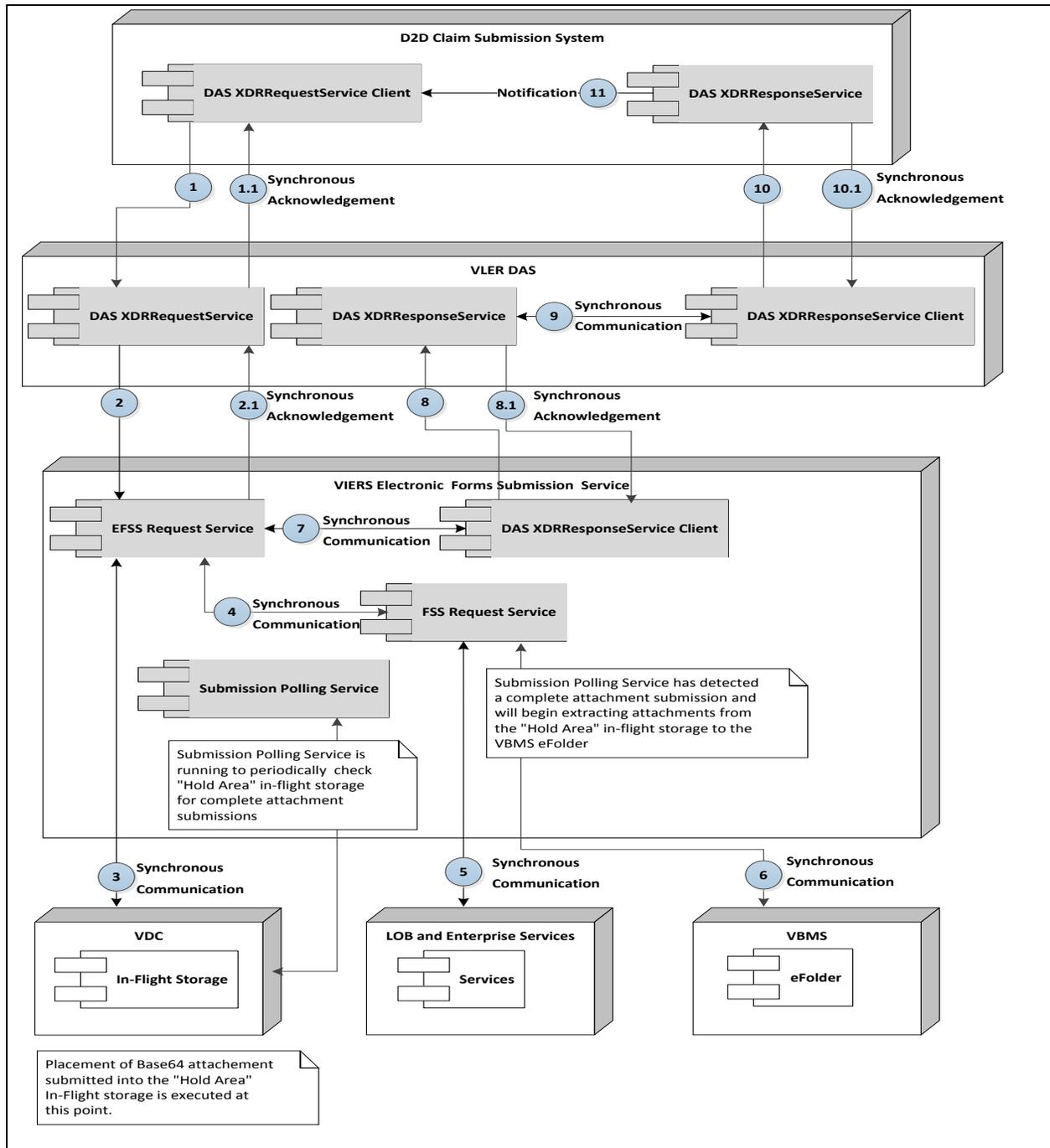
Figure 15 - Submit Form Component/Collaboration Diagram



6.2. Submit Attachment Service Messaging Protocol

The sequence of component interactions is defined by the numbered circles on the diagram. Numbers with a .1 reference (e.g. 1.1) signify a synchronous communication between components between layers on the component stack. This was done to highlight the intra-component stack touch points.

Figure 16 - Submit Attachment Component/Collaboration Diagram



7. Service Protocol Payload Examples

Section 5 **Service Messaging** Protocol defined the submitForm and submitFormAttachment scenarios utilizing sequence diagram as well verbiage that detailed the steps. This section will include payload examples mapped back to the numbered items in the sequence diagrams.

7.1. Form Transmission Protocol Example

This section maps the submitForm numbered items in the sequence diagram contained in section 5.1 **Form Transmission Protocol**.

Step 1: ProvideAndRegisterDocumentSet-bRequest operation using the ProvideAndRegisterDocumentSetRequest message

This link contains the **submitFormRequest.xml** file corresponding to this step:



submitFormRequest.
xml

Step 2: ProvideAndRegisterDocumentSet-bRequest operation using the Acknowledgement message

This link contains the **SUCCESS: Request Received** message corresponding to this step:



submitFormRequestR
esponse.xml

Step 7: ProvideAndRegisterDocumentSet-bResponse operation using the RegistryResponse message

This link contains the **submitFormResponse.xml** file corresponding to this step:



submitFormResponse
.xml

Step 8: ProvideAndRegisterDocumentSet-bResponseResponse using Acknowledgement "SUCCESS: Request Received"

This link contains the **submitFormResponseResponse.xml** file corresponding to this step:



submitFormResponse
Response.xml

7.2. Attachment Submission Protocol Example

7.2.1. Form PDF Attachment Transmission Protocol Example

This section maps the submitFormAttachment numbered items in the sequence diagram contained in section **5.2 Attachment Submission Protocol**.

Step 1: ProvideAndRegisterDocumentSet-bRequest operation using the ProvideAndRegisterDocumentSetRequest message

This link contains the **submitAttachmentFormRequest.xml** file corresponding to this step:



submitAttachmentFo
mRequest.xml

Step 2: ProvideAndRegisterDocumentSet-bRequestResponse operation using the Acknowledgement message

This link contains the **SUCCESS: Request Received** message corresponding to this step:



submitAttachmentFo
mRequestResponse.

Step 7: ProvideAndRegisterDocumentSet-bResponse operation using the RegistryResponse message

This link contains the **submitAttachmentFormResponse.xml** file corresponding to this step:



submitAttachmentFo
mResponse.xml

Step 8: ProvideAndRegisterDocumentSet-bResponseResponse operation using the Acknowledgement message

This link contains the **SUCCESS: Request Received** message corresponding to this step:



submitAttachmentFo
mResponseResponsi

7.2.2. PDF Attachment Transmission Protocol Example

The mapping of the PDF Attachment is exactly the same as described in section **7.2.1 Form PDF Attachment Transmission Protocol Example** so this process will not be detailed here.

7.2.3. Final Form Processing Protocol Example

This section maps the Final Form Processing items in the sequence diagram contained in section 5.2.3 Final Form Processing Protocol.

Step 3: ProvideAndRegisterDocumentSet-bResponse operation using the RegistryResponse message

This link contains the **submitFormResponseMessage.xml** file corresponding to this step:



FinalFormResponse.
xml

Step 4: ProvideAndRegisterDocumentSet-bResponse operation using the Acknowledgement message“

This link contains the **SUCCESS: Request Received** message corresponding to this step:



FinalFormResponseR
esponse.xml

8. 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 5 **Service Messaging** 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 operation sent from the DAS to the VSO.

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

Section 5 **Service Messaging** Protocol supplies a more detail operation interaction view.

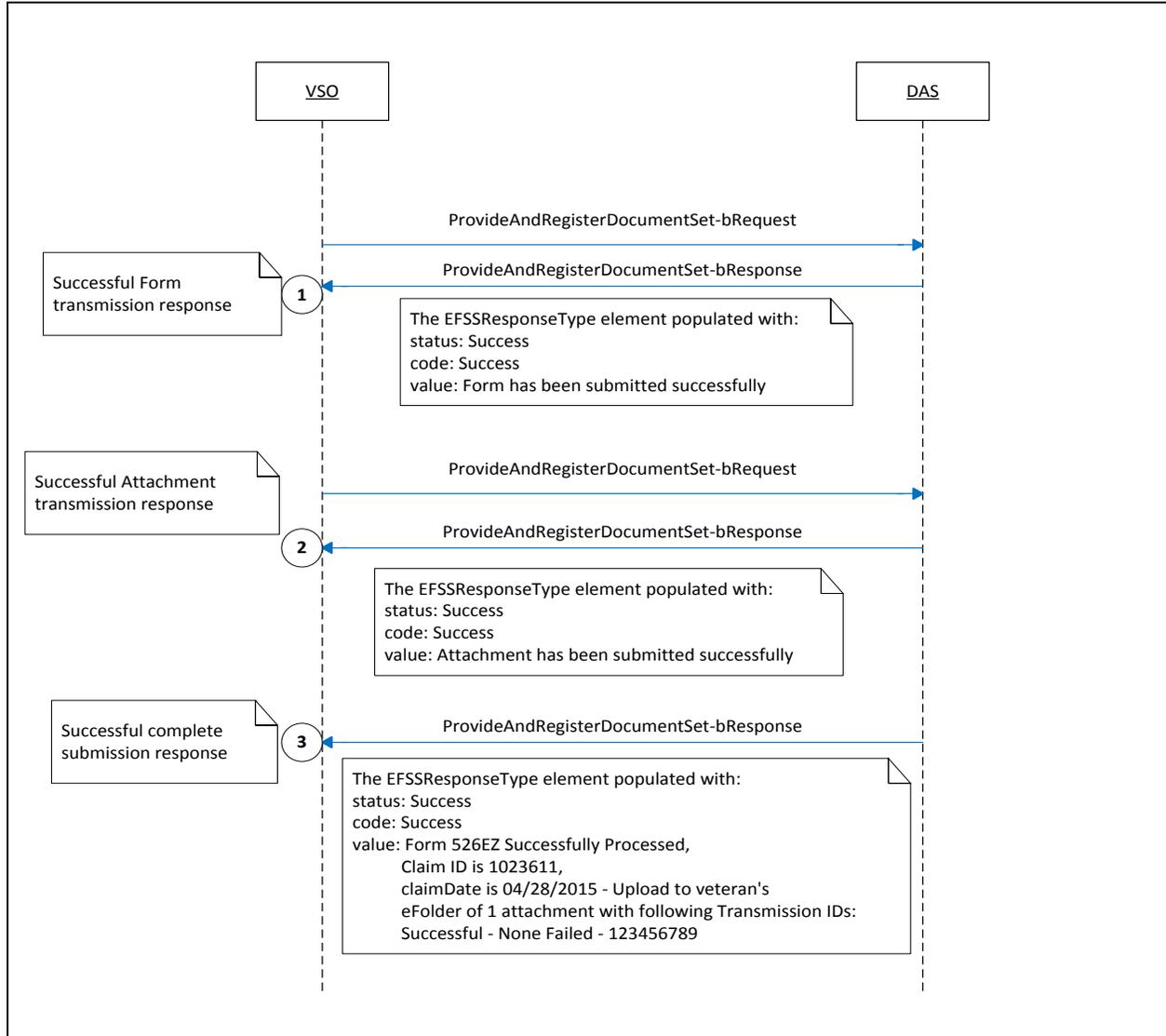
The scenario completion state will be categorized by:

- Successful
- Recoverable Failure
- Non-recoverable failure

8.1. Successful 21-526EZ submission

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

Figure 17 – Successful 21-526EZ Form/Attachment Submission



Response Results:

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

Completion State -:

Successful

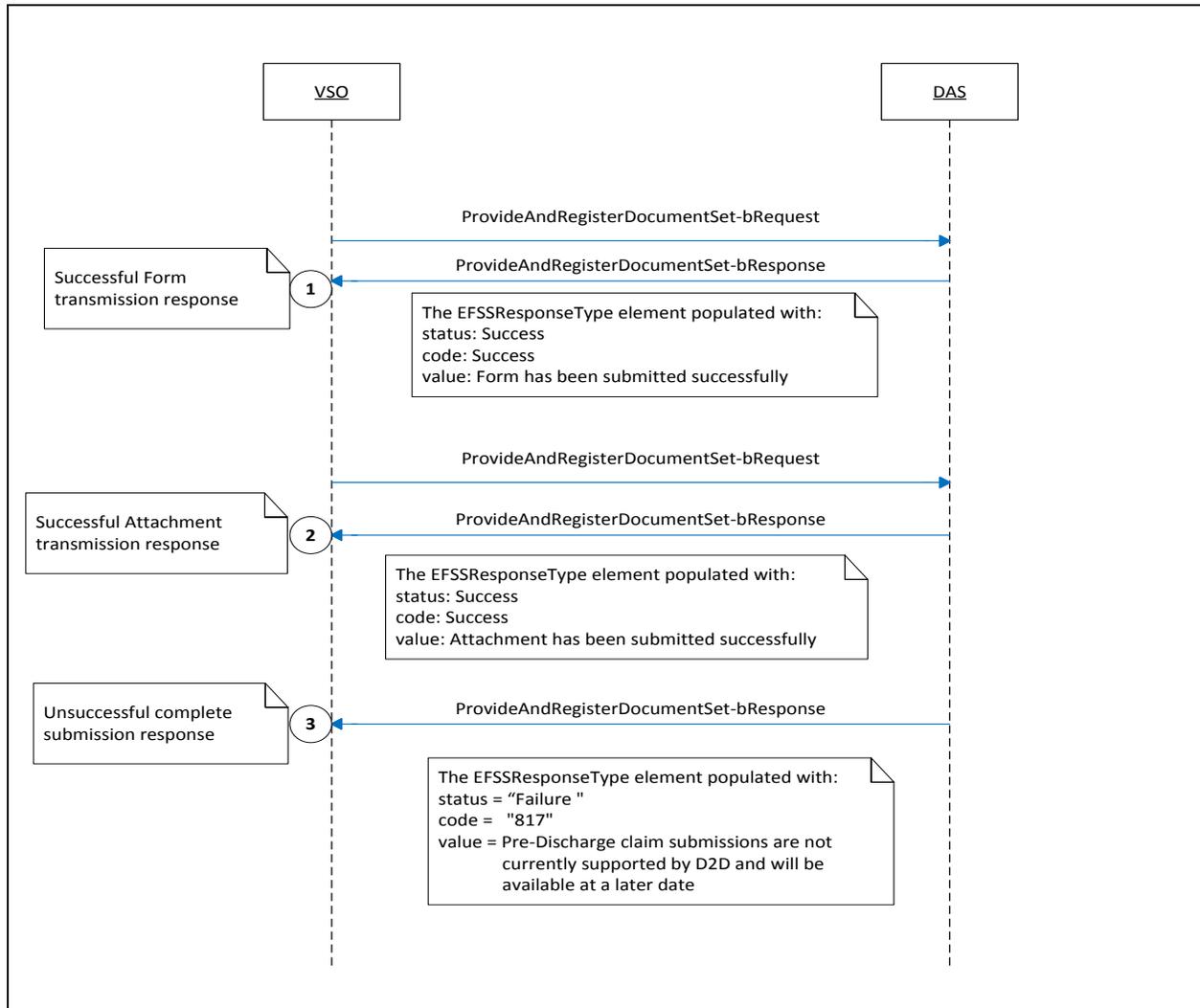
Remediation Strategy:

Because this is a successful scenario remediation is not required

8.2. 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 18 – Unsuccessful 21-526EZ Form/Attachment Submission



Response Results:

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

Completion State

Unrecoverable:

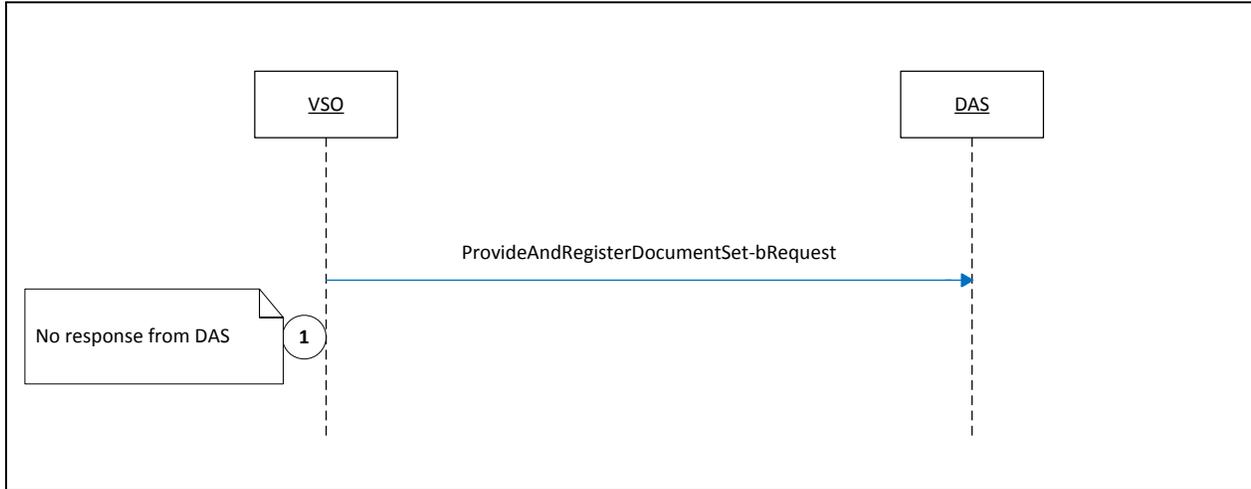
Remediation Strategy:

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

8.3. 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 19 – Unsuccessful 21-526EZ Form/Attachment Submission



Response Results:

1. No Response from DAS

Completion State:

Potentially recoverable with the **Additional Supporting Document Submission** option or an entire form and attachment resubmission

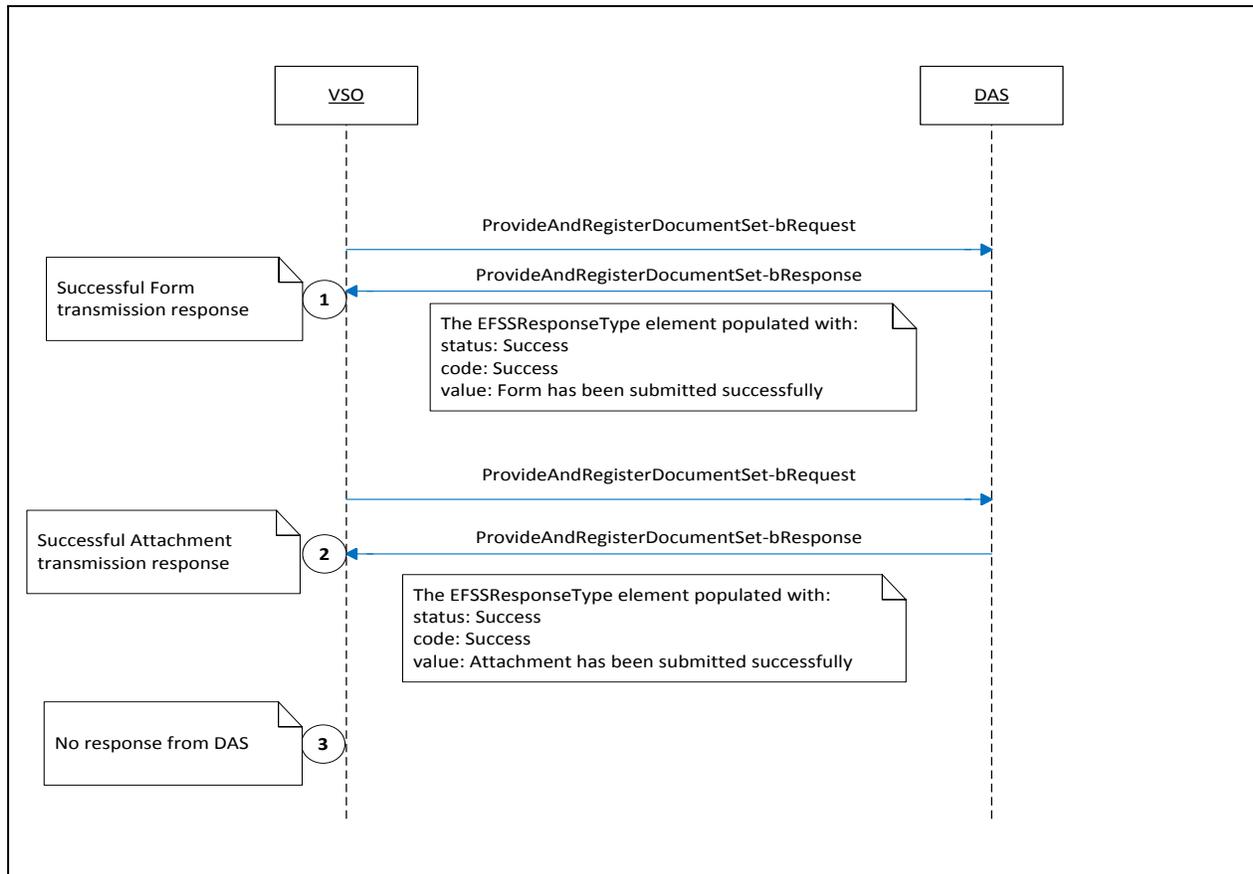
Remediation Strategy:

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 Additional Supporting Documents 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)

8.4. 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 20 – 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 **Additional Supporting Document Submission** option or an entire form and attachment resubmission

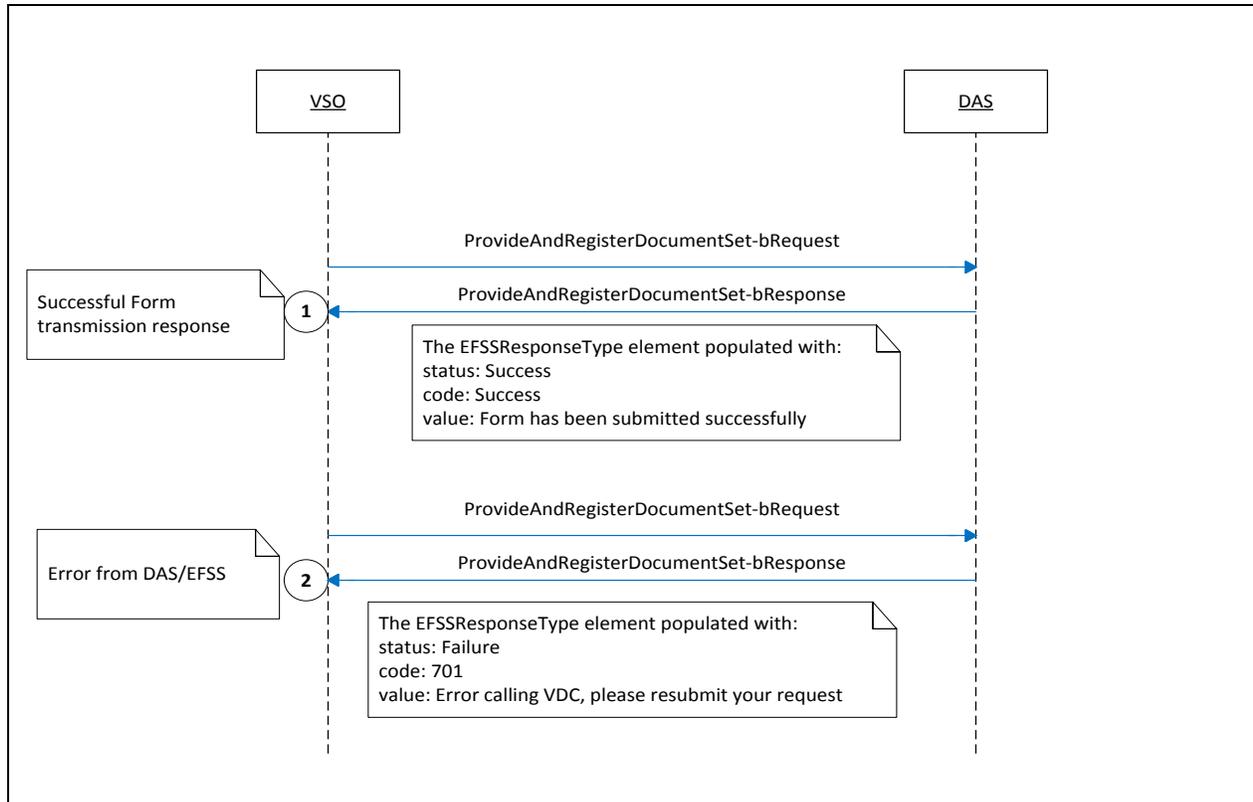
Remediation Strategy:

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 Additional Supporting Documents 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)

8.5. 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 **Additional Supporting Document Submission** 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 Additional Supporting Documents 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)