



UNITED STATES
DEPARTMENT OF VETERANS AFFAIRS

Department of Veterans Affairs

Product Development (PD)

Veterans Benefits Management System (VBMS)

VBMS eDocument Service Version 4.0
Service Description Document
VBMS-SPEC-00058



Version 3.0
May 2014

Space and Naval Warfare Systems Center Atlantic
P.O. Box 190022
North Charleston, SC 29419-9022



Revision History

Date	Version	Description	Author
05/20/2014	3.0	Updated to include updates for CDM version 4.1	VBMS Architecture
04/03/2014	2.0	Updated to include message header that allows service consumers to provide a user identifier.	VBMS Architecture
01/14/2014	1.0	Updates that capture eDocument Service Version 4.0 contract changes released in VBMS-Core R6.1. Forked from <i>eDocument Service Version 3.0 Service Description Document</i> (VBMS-SPEC-0013).	VBMS Architecture

Table of Contents

1. Veterans Benefits Management System Mission.....	1
2. Introduction	2
2.1 Scope	2
3. Service Information	2
3.1 Service Identification	3
3.2 Service Governance.....	3
3.3 Current Service Status.....	3
4. Dependency Model	4
5. eDocumentService Interface.....	5
5.1 Interface Description.....	5
5.2 Technical Specification	8
5.3 Information Model	9
5.3.1 Veteran.....	9
5.3.2 Document	9
5.3.3 DocumentContent.....	11
5.3.4 DocumentAssociationInput	11
5.3.5 FormDocument	11
5.3.6 LetterDocument	12
5.3.7 FormField	12
5.3.8 DocumentType.....	12
5.3.9 ConvertedDocumentResponse	13
5.3.10 DocumentWithContentResponse	13
5.3.11 DocumentAssociation	13
5.4 Behavior Model	14
5.4.1 Use Cases.....	14
5.4.2 Sequence Diagrams.....	14
5.4.3 Message Headers	15
5.4.4 Operation Contracts	16
5.5 Policies and Qualities of Service	36
5.5.1 Service Policies	36
5.5.2 SAML Assertion Guidelines.....	36
5.5.3 Performance.....	37
5.5.4 Business Metrics	38

5.5.5 Standards Compliance	38
6. Security	38
6.1 Message-Level Security.....	38
6.2 Document Content References	39
6.2.1 uploadFormDocument.....	39
6.2.2 uploadDocument	40
6.2.3 uploadDocumentWithAssociations.....	40
6.2.4 uploadLetterDocument.....	40
6.3 SAML Token Injection.....	41
Appendix A: Acronym List.....	44
Appendix B: Approval Signatures.....	46

List of Figures

Figure 1: eDocument Service Version 4.0 Dependencies.....	5
Figure 2: Use Cases	14
Figure 3: Request – Response Sequence Diagram.....	15
Figure 4: eDocumentService Credential Flow	42

List of Tables

Table 1: Service Administrative Information	3
Table 2: Current Service Status	3
Table 3: Technical Specification	8
Table 4: CDM Object for Defining a Veteran	9
Table 5: CDM Object: Document.....	9
Table 6: CDM Object: DocumentContent	11
Table 7: CDM Object: DocumentAssociationInput.....	11
Table 8: CDM Object: FormDocument.....	11
Table 9: CDM Object: LetterDocument.....	12
Table 10: CDM Object: FormField.....	12
Table 11: CDM Object: DocumentType	12
Table 12: CDM Object: ConvertedDocumentResponse	13
Table 13: CDM Object: DocumentWithContentResponse	13
Table 14: CDM Object: DocumentAssociation	13
Table 15: uploadDocumentWithAssociations	16
Table 16: uploadDocumentWithAssociations Request Element.....	18
Table 17: uploadDocumentWithAssociationsResponse Response Element	19
Table 18: vbmsServiceException	19
Table 19: uploadDocument	19
Table 20: uploadDocument Request Element	22

Table 21: uploadDocumentResponse Response Element	22
Table 22: vbmsServiceException	22
Table 23: fetchDocumentByld	22
Table 24: fetchDocumentByld Request Element	23
Table 25: fetchDocumentResponse Response Element.....	23
Table 26: vbmsServiceException	23
Table 27: fetchDocument.....	24
Table 28: fetchDocument Request Element.....	25
Table 29: fetchDocumentResponse Response Element.....	25
Table 30: vbmsServiceException	25
Table 31: uploadFormDocument.....	25
Table 32: uploadFormDocument Request Element.....	29
Table 33: uploadFormDocumentResponse Response Element.....	29
Table 34: vbmsServiceException	29
Table 35: getFormFieldsForDocumentType	29
Table 36: getFormFieldsForDocumentType Request Element	30
Table 37: getFormFieldsForDocumentTypeResponse Response Element	30
Table 38: vbmsServiceException	30
Table 39: vbmsDataValidationException	30
Table 40: validationError	30
Table 41: uploadLetterDocument.....	31
Table 42: uploadLetterDocument Request Element.....	32
Table 43: uploadLetterDocumentResponse Response Element.....	32
Table 44: vbmsServiceException	32
Table 45: getDocumentTypes	33
Table 46: getDocumentTypes Request Element	33
Table 47: getDocumentTypesResponse Response Element	33
Table 48: listDocuments	33
Table 49: listDocuments Request Element	34
Table 50: listDocumentsResponse Response Element	34
Table 51: vbmsServiceException	34
Table 52: listDocumentsByDocumentTypeld	35
Table 53: listDocumentsByDocumentTypeld Request Element.....	35
Table 54: listDocumentsByDocumentTypeldResponse Response Element	35
Table 55: vbmsServiceException	35
Table 56: Standards Compliance	38
Table 57: uploadFormDocument Signature	39
Table 58: uploadFormDocument Encryption.....	39
Table 59: uploadDocument Signature	40
Table 60: uploadDocument Encryption.....	40
Table 61: uploadDocumentWithAssociations Signature	40
Table 62: uploadDocumentWithAssociations Encryption.....	40
Table 63: uploadLetterDocument Signature	40
Table 64: uploadLetterDocument Encryption.....	41

This page intentionally left blank.

1. Veterans Benefits Management System Mission

The mission of the Department of Veterans Affairs (VA), Office of Information and Technology (OIT), Product Development (PD) is to provide benefits and services to Veterans of the United States. In meeting these goals, OIT strives to provide high quality, effective, and efficient Information Technology (IT) services to those responsible for providing care to Veterans on-site and throughout all the points of the Veterans' health care in an effective, timely and compassionate manner. VA depends on Information Management/Information Technology (IM/IT) systems to meet mission goals. Under Interagency Agreement (IAA) VA 118-10-IA-0001, dated September 29, 2011 between VA and the Department of the Navy, Space and Naval Warfare Systems Center, Atlantic (SPAWARSYSCEN Atlantic), VA and OIT requests SPAWARSYSCEN Atlantic to assist VA in meeting its mission goals. This assistance is in the areas of project management, configuration management, requirements management, integration with VA and non-VA systems, interfaces to VA and non-VA systems, Virtual VA conversion, web portal development, services development, testing, information assurance, architectural engineering support and the integration of the various components/documents being delivered from various contractors and documentation to VA.

Veterans Benefits Administration (VBA) and OIT are seeking solutions for managing the backlog of paper claims, electronic documents, correspondence, and other content created and handled as part of their day-to-day business processes. As the volume of content grows exponentially, VA faces increased pressure to operate more efficiently, while reducing costs and addressing Veterans' needs for faster, simpler, and more efficient and effective services while adhering to industry best practices and compliance regulations.

To accomplish this, a comprehensive plan is in the process of being executed that will:

- Target deficiencies in workflows to maximize efficiencies and enable the Veteran to receive benefits in a timely and efficient manner, as well as provide Veterans with the ability to monitor the status of his/her benefits.
- Promote a solution that provides a unified content, process and compliant environment.
- Provide interoperability between databases, applications, operating systems, portals, security, servers, storage, systems management tools and web server environments.
- Deploy a Service Oriented Architecture (SOA) shared services platform to help reduce operations costs and simplify IT infrastructure.
- Provide operational efficiency across the organization by creating a common interface for building and deploying content and process applications.

Veterans Benefits Management System (VBMS) is a multi-year technology solutions project to transition VBA from a paper-intensive claims processing environment to a paperless-based environment. The ultimate goal of this program is to provide VBA with systems engineering support for the overall design of the VBMS solution (to include security components). This support also includes test and integration services and the development of VBMS, the development of SOA objects, the development of services for the user interface to communicate with the corporate database, the development of the VBMS database, security to meet the requirements of *VA Directive 6500 Information Security Handbook*, and the development of functionality to support end-to-end claims processing in an electronic environment. End-to-end claims processing provides functionality required for establishment, development, rating, award, and appeal of a claim. It is the intent of this effort to deliver VBMS incrementally using an agile development methodology based upon the priorities determined by OIT.

2. Introduction

The *VBMS eDocument Service Version 4.0 Service Description Document (SDD)* presents the information required to consume a service to clarify whether it is appropriate for the service consumer's needs. This document provides information and a behavior model, as well as descriptions of service policies, quality of service, and service governance.

2.1 Scope

VBMS eDocument Service Version 4.0 SDD defines the eDocument Service for effective consumption by applications and other services. This document intentionally does not contain information about specific service consumers. Also, this document does not contain information about the execution context as it may be implemented and deployed in different environments.

3. Service Information

This section defines the eDocument Service Version 4.0 for management, identification, and governance.

The eDocument Service Version 4.0 provides a way for external systems to add and retrieve VBMS Veteran's Compensation & Pension (C&P) claim documents. Uploaded documents could be:

- Scanned forms (e.g., VA Form 21-526)
- Correspondence
- Evidentiary documents
- Raw XML data for certain Disability Benefit Questionnaire (DBQ) documents
- Any other document relating to a specific Veteran's claim

The uploaded document (typically a PDF), any associated metadata, and Optical Character Recognized (OCRed) form fields are stored in the VBMS document repository. The eDocument Service Version 4.0 also provides a fetching capability to retrieve specific documents based on specific queries.

The eDocument Service Version 4.0 supports Message Transmission Optimization Mechanism (MTOM) streaming by moving document content outside of the encrypted SOAP (Simple Object Access Protocol) request/response messages. Also the eDocument Service Version 4.0 supports Version 4 of the VBMS Common Data Model (CDM), which streamlines various CDM objects (e.g., Document) and also organizes CDM objects into logical groups with unique version numbers for each CDM logical group. CDM Version 4 provides a common model across all VBMS projects. The eDocument Service Version 4.0 also enables validation of XSD schema and strict enforcement of MTOM policy.

For example, the target namespace for the *Document* object used by eDocument Service Version 4.0 is now:

<http://vbms.vba.va.gov/cdm/document/v4>

Where previously it was:

<http://vbms.vba.va.gov/cdm/document/v2>

Note: CDM logical groups have independent version numbers that do not correlate with version numbering for the eDocument Service Version 4.0. Also note that updated service operations are not backwards compatible and that existing eDocument Service clients must rebind to the new Web Service Description Language (WSDL) contracts to use the eDocument Service Version 4.0.

MTOM-streaming substantially reduces system resource consumption, especially when transferring large, multi-megabyte documents. MTOM implementation details are unique to each client's chosen Web Service framework, and MTOM specification details are out of scope of the *VBMS eDocument Service Version 4.0 SDD*.

3.1 Service Identification

Table 1 provides the eDocument Service Version 4.0 administrative information for management and governance purposes.

Table 1: Service Administrative Information

Item	Details
Service Name	eDocumentService
Service Version	4.0
Business Domain	To Be Determined (TBD)
Business Owner	TBD
Development Owner	OIT, VA Program Management Office (PMO)
Support Owner	OIT, VA PMO
Technical Owner	OIT, VA PMO
Service Authority Level	TBD

3.2 Service Governance

The overall service governance, including the change management process, for this and other VBMS hosted services, is not yet defined.

3.3 Current Service Status

The current service status shown in Table 2 shows service progress from initiation through development, deployment, and eventual retirement.

Table 2: Current Service Status

Service State	Description	Date Entered
Planned	Owner intends to create the service or a new version of the service (or a constituent service) in cases where it is not well defined. This document may still change materially.	02JAN14
Rejected	Owner has decided not to develop the service.	N/A

Service State	Description	Date Entered
Designed	The service design is complete. Service consumers may begin development based on provided documentation. This document should not change materially.	02JAN14
Developed	The service is developed. Service consumers can begin development integration with the service. This document is stable.	14JAN14
Deployed	The service has been deployed in a production environment. This document is complete.	N/A
Deprecated	The service has been replaced by a new release or a different service. All service consumers should transition to the new service and no new consumers will be accepted.	N/A
Retired	This service is no longer deployed.	N/A

4. Dependency Model

The Dependency Model shown in Figure 1 identifies other services, systems, and databases the eDocument Service Version 4.0 either depends upon or interacts with, including:

- **Benefits Gateway Service (BGS)** – The eDocument Service Version 4.0 uses BGS to store “flash” information to alert users about the availability of new uploaded documents into VBMS. BGS is also used to search for information about the Veteran associated with uploaded or retrieved documents, and to verify the existence of claims or contentions associated with uploaded documents.
- **VBMS Database** – The eDocument Service Version 4.0 uses the VBMS database for storing auditing, performance logging, and document association information.
- **FileNet** – The eDocument Service Version 4.0 interacts with FileNet, an IBM Commercial Off-the-Shelf Software (COTS) product, as the document management system and repository for VBMS documents and associated document metadata.

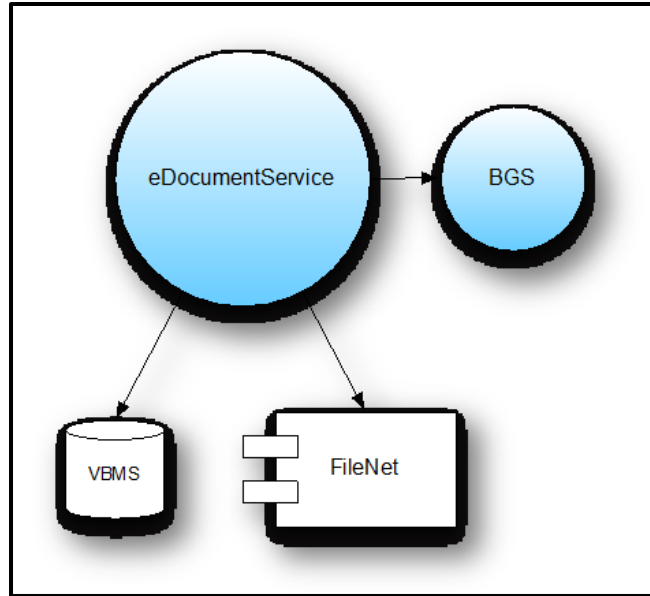


Figure 1: eDocument Service Version 4.0 Dependencies

5. eDocumentService Interface

The primary differences between previous versions and Version 4.0 of the eDocument Service are the support for CDM Version 4.0, XSD schema validation, and support for the MTOM streaming specification with strict enforcement of the MTOM policy. This support enables a reference to document content outside of the *Document* element within the sibling *DocumentContent* element. Web Service Security (WSS) encryption is not applied to the *DocumentContent* element as MTOM-streaming is incompatible with WSS encryption applied to the *DocumentContent* reference. Section 6.1 provides additional information on WSS changes for eDocument Service Version 4.0.

The eDocument Service Version 4.0 performs additional validations on the *uploadFormDocument* operation, ensuring certain Document Control Sheet (DCS) fields are provided for the traceability of other not yet scanned VA Forms for a specific File Number.

5.1 Interface Description

The eDocument Service interface supports the management of eDocuments used in VBMS. Available operations are as follows:

- **uploadDocumentWithAssociations** – Uploads an MTOM-streamed document into the eDocument Service Version 4.0. This operation accepts a document and an optional list of *DocumentAssociationInput* objects for associations with other (external) business entities. The currently supported document association types are 'CLAIM' and 'CONTENTION' (referencing the Claim ID Number or Contention ID Number supplied by BGS). The *uploadDocumentWithAssociations* operation returns both a *Document* object containing the identifier of the stored document and a list of associated *Document* objects containing unique identifiers for each uploaded and converted document.
 - If the *Document* or the *DocumentContent* argument is missing, the operation returns a *VbmsServiceException* SOAP fault.

- Required fields for the Document parameter:
 - If filenumber or filename is empty, the operation returns a VbmsServiceException SOAP fault.
- Required fields for each *DocumentAssociationInput* object (if specified):
 - If associationType or associationValue is empty, the operation returns a VbmsServiceException SOAP fault.
- **uploadDocument (DEPRECATED)** – Uploads an MTOM-streamed document into the eDocument Service. This operation expects a *Document* and a *Veteran* object.
 - This uploadDocument legacy operation does not support the *DocumentAssociations* element and requires a *Veteran* object to be passed to provide additional document related information. This operation also requires an externalId field with each document upload request. The externalId field is particular to only the service consumer and cannot be used to convey meaning in VBMS.
 - If either the *Veteran* or *Document* objects are missing, the operation returns a VbmsServiceException SOAP fault.
 - Required fields:
 - The *Document*, *docType*, and *DocumentContent* elements and attributes are mandatory.
 - If the document's filename, filenumber, or externalId is empty, the operation returns a VbmsServiceException SOAP fault.
- **fetchDocumentById** – Fetches a document using a document identifier. Document content is returned in an MTOM stream referenced by the *DocumentContent* element.
 - If the document identifier is empty, the operation returns a VbmsServiceException SOAP fault.
- **fetchDocument** – Fetches a document based on the veteran fileNumber and document externalId attributes. The document content is returned in an MTOM stream referenced by the *DocumentContent* element.
 - The document fetch operation does not support the *DocumentAssociationInput* element type. The document externalId field is required in this fetch operation.
 - Required fields:
 - If the *Veteran* element or filenumber attribute is empty, the operation will return a VbmsServiceException SOAP fault.
 - If the Document or externalId attribute is empty, the operation will return a VbmsServiceException SOAP fault.
- **uploadFormDocument** – Uploads an MTOM-streamed document into the eDocument Service Version 4.0 with associated form data. This operation expects a *FormDocument* object, which includes a list of *FormField* type elements, each containing a field name and field value). This version of the operation requires the dcsId and dcsScanningComplete fields in the *FormDocument* object to be provided on each invocation.
 - If either the *FormDocument* or *DocumentContent* argument is missing, the operation returns a VbmsServiceException SOAP fault.

- Required and invalid fields:
 - `DocumentType`, `dcsId`, and `dcsScanningComplete` fields are mandatory
 - If `filenumber` or `filename` is empty, the operation returns a `VbmsServiceException` SOAP fault
 - If any required form field is omitted, the operation returns a `VbmsDataValidationException` SOAP fault that includes a list of any missing required fields
 - If the submitted data exceeds the `maxLength` for any form field, the operation returns a `VbmsDataValidationException` SOAP fault that includes a list of invalid fields
- **getFormFieldsForDocumentType** – An operation that returns a list of *FormField* objects available for the document type passed as a parameter
 - Required fields:
 - If the provided `DocumentType` value is not valid, the method returns a `VbmsServiceException` SOAP fault.
- **uploadLetterDocument (UNSUPPORTED)** – Uploads an MTOM-streamed letter document into the eDocument Service. Accepts a *claimId* and a *letterId* element in addition to the *Document* element's content.
 - This service operation is not currently used and is not being actively tested. Please do not use unless explicit permission is granted from your VBMS point of contact (POC).
 - If the *Document* or *DocumentContent* object is missing, the operation returns a `VbmsServiceException` SOAP fault.
 - Required fields:
 - `DocumentType` is mandatory
 - If the `filename` or `filenumber` is empty, the operation returns a `VbmsServiceException` SOAP fault
- **getDocumentTypes** – Returns information about VBMS document types.
- **listDocuments** – Returns a list of the *Document* objects associated with a Veteran (filtered by the specified criteria). This operation returns *Document* objects from the `http://vbms.vba.va.gov/cdm/document/v4` namespace. The previous version of this operation returned *Document* objects from the `http://vbms.vba.va.gov/cdm` namespace.
 - Required fields:
 - If the `Veteran` or `filenumber` is empty, the operation returns a `VbmsServiceException` SOAP fault.
 - Whenever there are entries in the list of *DocumentAssociationInput* objects, if `associationType` or `associationValue` fields are empty, the method returns a `VbmsServiceException` SOAP fault.
- **listDocumentsByDocumentTypeId** – This operation returns a list of *Document* objects based on the `filenumber` and `documentTypeId` fields.
 - Required fields:

- If the fileNumber field is null or empty, the operation returns a VbmsServiceException SOAP fault.
- If the documentTypeId field is null or empty, the operation returns a VbmsServiceException SOAP fault.

5.2 Technical Specification

The technical specification in Table 3 allows service consumer developers to discover the service for runtime consumption.

Table 3: Technical Specification

Item	Details
Service Invocation Type	SOAP over HTTP
Service Interface Type	WSDL via Web Service 2.0
Service Name	eDocumentService-v4
Interface	https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4?wsdl
Schemas	https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4?wsdl
End Points	<p>Document Uploading: https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4/uploadDocumentWithAssociations https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4/uploadDocument https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4/uploadFormDocument https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4/uploadLetterDocument</p> <p>Document Fetching: https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4/fetchDocument https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4/fetchDocumentById</p> <p>Document Listing: https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4/listDocuments https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4/listDocumentsByDocumentTypeId</p> <p>Get Form Fields for Document Type: https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4/getFormFieldsForDocumentType</p> <p>Get Document Types: https://<service_domain>/vbmsp2-cms/streaming/eDocumentService-v4/getDocumentTypes</p>

5.3 Information Model

This section defines data objects provided by eDocument Service Version 4.0 as part of message payloads and service contracts. These objects do not define a database schema or class model, but do define communicated objects between the service and service consumers.

5.3.1 Veteran

Veteran is a CDM object for defining a Veteran; it extends a CDM *Person* object. Only *firstName*, *lastName*, and *fileNumber* are used.

Table 4: CDM Object for Defining a Veteran

Name	Element/ Attribute	Data Type	Comments
firstName	Element	string	Used only in uploadDocument operation
lastName	Element	string	Used only in uploadDocument operation
fileNumber	Attribute	string	Used in both uploadDocument and fetchDocument operations

5.3.2 Document

Document is a CDM object representing a VBMS electronic document.

Table 5: CDM Object: Document

Name	Element/ Attribute	Data Type	Comments
ID	Attribute	string	
fileNumber	Attribute	string	Required Not used in uploadDocument Used in uploadFormDocument and uploadLetterDocument Typical format is eight or nine digits in length
externalId	Attribute	string	Required for uploadDocument Used for fetching documents Documents can ONLY be fetched by externalId
filename	Attribute	string	Required File names MUST be unique for each document uploaded to a particular file number Refer to the <i>VBMS PDF</i>

Name	Element/ Attribute	Data Type	Comments
			<i>Specification Document</i> (VBMS-SPEC-00021) for details on the expected file naming convention
docType	Attribute	string	Required
actionable	Attribute	boolean	
veteranPersonId	Attribute	long	
path	Attribute	string	
category	Attribute	string	
metadata	Attribute	string	
newMail	Attribute	boolean	Required. The newMail Boolean indicator must be set to "true" if the claim associated with the uploaded document is still in a pending (i.e., non-completed) status. If all associated claims have been completed, this indicator should be set to "false"
contentType	Attribute	string	
subject	Attribute	string	
endProduct	Attribute	string	
source	Attribute	string	
shippingNum	Attribute	string	
veteranFirstName	Attribute	string	
veteranMiddleName	Attribute	string	
veteranLastName	Attribute	string	
veteranSuffix	Attribute	string	
dcslId	Attribute	string	
receivedDt	Element	date	Required
annotation	Element	<i>List</i> <Annotation>	List of references to annotation element types
documentAssociations	Element	<i>List</i> <DocumentAssociation>	List of references to associated documents

5.3.3 DocumentContent

DocumentContent is a CDM object that represents streamable document content to be stored as the contents of the related *Document* object.

Table 6: CDM Object: DocumentContent

Name	Element/ Attribute	Data Type	Comments
data	Element	base64Binary	eDocument Service Version 4.0 enables strict enforcement of the MTOM policy that requires document content is an XOP attachment. This removes the option of having inline base-64 encoded representation for document content.

5.3.4 DocumentAssociationInput

DocumentAssociationInput is a CDM object that represents the consumer-provided data used to relate entities to a Document. This is the same as DocumentAssociations, in that neither contains a Document ID reference (in most cases the Document ID is not known), nor does it create and modify the date.

Table 7: CDM Object: DocumentAssociationInput

Name	Element/ Attribute	Data Type	Comments
associationType	Attribute	Enumeration	Enumerations defining the relationship between a document and another VBMS business object. 'CLAIM' and 'CONTENTION' are the only input values accepted during document upload. On document listing, 'CLAIM', 'CONTENTION', and 'DOCUMENT' may be returned.
associationValue	Element	string	The associated business object's value (e.g., a valid Claim Id or a valid Contention Id). The currently supported document association types are 'CLAIM' and 'CONTENTION'

5.3.5 FormDocument

FormDocument is a CDM object that extends the *Document* object and contains a list of associated VBMS FormFields. All attributes in the base type (Document) are included by reference in *FormDocument*. Table 8 includes only attributes that are not part of the base type.

Table 8: CDM Object: FormDocument

Name	Element/ Attribute	Data Type	Comments
formFields	Element	List<FormField>	

dcsScanningComplete	Attribute	Boolean	
---------------------	-----------	---------	--

5.3.6 LetterDocument

LetterDocument is a CDM object that extends the Document type and adds a VBMS claimId and letterId. All attributes in the base type (Document) are included by reference in *LetterDocument*. Table 9 includes attributes that are not part of the base type.

Table 9: CDM Object: LetterDocument

Name	Element/ Attribute	Data Type	Comments
claimId	Attribute	string	
letterId	Attribute	string	

5.3.7 FormField

FormField is a CDM object representing a form field present in the document. Each document can have zero or more *FormField* objects.

Table 10: CDM Object: FormField

Name	Element/ Attribute	Data Type	Comments
name	Attribute	string	
value	Element	anyType	Supported Data Types: boolean, string, date, and long For the document upload to succeed; the provided data type MUST match the corresponding data type for the identically named field in eDocument Service Version 4.0
maxLength	Attribute	long	
required	Attribute	Boolean	

5.3.8 DocumentType

The *DocumentType* is a CDM object that type maps a document's docType value to a name and description for a particular type of document.

Table 11: CDM Object: DocumentType

Name	Element/ Attribute	Data Type	Comments
id	Attribute	long	
name	Attribute	string	
description	Attribute	string	
filenetDocumentClass	Attribute	string	

cdmDocumentClass	Attribute	string	
------------------	-----------	--------	--

5.3.9 ConvertedDocumentResponse

The *ConvertedDocumentResponse* object associates an original document with converted documents generated from the original.

Table 12: CDM Object: ConvertedDocumentResponse

Name	Element/ Attribute	Data Type	Comments
originalDocument	Element	cdm/document/v4:Document	Contains the VBMS document identifier for the uploaded document
convertedDocuments	Element	List<cdm/document/v4:Document>	Contains a list of converted <i>Document</i> objects resulting from VBMS document conversion logic

5.3.10 DocumentWithContentResponse

The *DocumentWithContentResponse* object associates a *Document* object with the binary contents delivered in the corresponding *DocumentContent* object.

Table 13: CDM Object: DocumentWithContentResponse

Name	Element/ Attribute	Data Type	Comments
document	Element	cdm/document/v4:Document	Contains VBMS document information for the returned document
content	Element	cdm/document/v4:DocumentContent	Contains document contents for the associated document

5.3.11 DocumentAssociation

DocumentAssociation is a CDM object that extends the *DocumentAssociationInput* object and provides additional association information for an associated *Document* object in the VBMS content store. This additional information is used to associate raw XML DBQ documents with associated PDF documents and vice versa.

Table 14: CDM Object: DocumentAssociation

Name	Element/ Attribute	Data Type	Comments
associatedDocType	Element	string	Contains the type identifier for the associated document type
documentId	Attribute	string	Contains the document identifier for the associated document

5.4 Behavior Model

A behavior model defines the supported eDocument Service Version 4.0 actions and processes. Actions and processes represented in the use cases and sequence diagrams are further defined by the operation contracts and the message payloads.

5.4.1 Use Cases

The primary eDocument Service Version 4.0 use cases, summarized in Figure 2, are:

- Uploading documents into VBMS
- Retrieving documents from VBMS

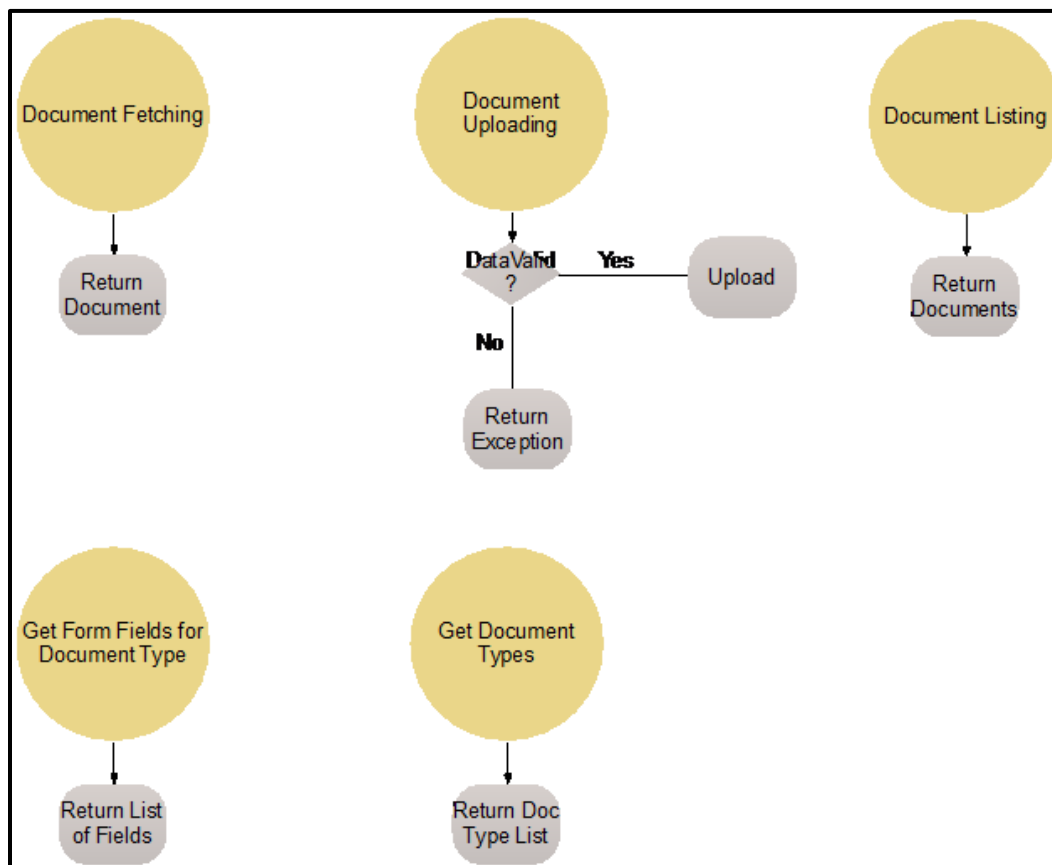


Figure 2: Use Cases

5.4.2 Sequence Diagrams

The eDocument Service Version 4.0 sequence diagram is straightforward for each operation. Business logic handled on the VBMS side of the service is irrelevant to consumers (i.e., a black box). Figure 3 shows the sequence of events for each operation.

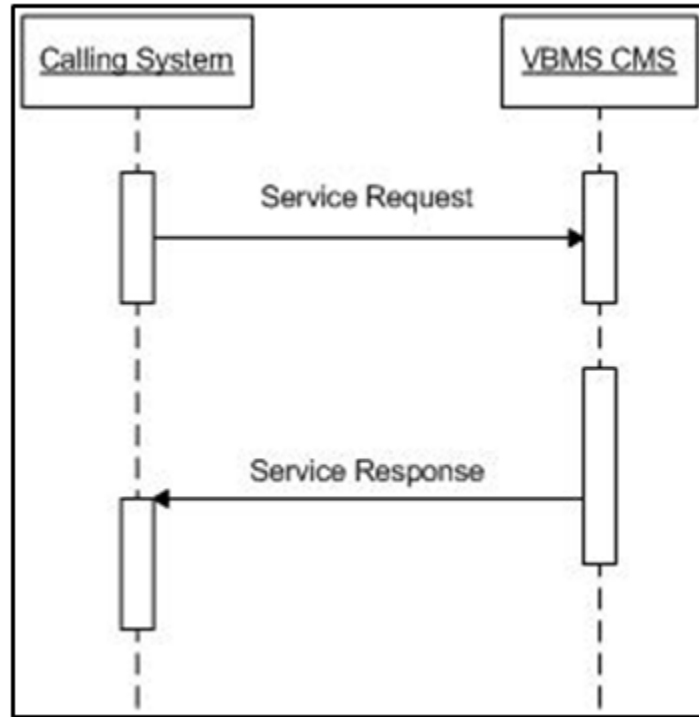


Figure 3: Request – Response Sequence Diagram

5.4.3 Message Headers

Message headers encapsulate information sent to and returned from the service in a common data structure. A common data structure is a consistent way of interacting with services, and reduces the necessary amount of rework when versioning services. Message headers are used to specify WSS information. Additional WSS details for the eDocument Service Version 4.0 are found in Section 6. Also message headers are used to receive the user identifier from the service consumer. The user identifier header (userId) is an optional header and contains a string with a 39 character limit. If the userId header exists in a request but has empty contents, it is treated as if the header was not present. An example of the userId header is as follows:

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:v4="http://vbms.vba.va.gov/external/eDocumentService/v4"
  xmlns:vbmsext="http://vbms.vba.va.gov/external">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      .....
    </wsse:Security>
    <vbmsext:userId>USERID_OF_SERVICE_CONSUMER</vbmsext:userId>
  </soapenv:Header>
  <soapenv:Body/>
</soapenv:Envelope>
  
```

5.4.4 Operation Contracts

Operation contracts define how service consumers interact with the eDocument Service Version 4.0. Information communicated between the service and service consumer is sent in the message payload to call an operation or method. The following sub-sections contain detailed information for each of the service operation contracts the service consumer may invoke.

5.4.4.1 *uploadDocumentWithAssociations*

```
wsdl:ConvertedDocumentResponse uploadDocumentWithAssociations (
    cdm/document/v4:Document document
    List<cdm:DocumentAssociationInput> documentAssociations
    cdm/document/v4:DocumentContent documentContent
) throws VbmsServiceException
```

Table 15: uploadDocumentWithAssociations

Method Name	uploadDocumentWithAssociations
Description	<p>Uploads an MTOM-streamed document with optional <i>documentAssociations</i> objects. The <i>fileNumber</i> must be provided to the <i>Document</i> object and the corresponding Veteran must be retrievable from CorpDB by that file number to invoke a successful operation.</p> <p>This is the preferred service operation for uploading documents without associated corresponding form fields. The only accepted values for <i>associationType</i> are 'CLAIM' and 'CONTENTION'. The 'DOCUMENT' value for <i>associationType</i> is not supported by this operation. Document associations are NOT required elements.</p> <p>VBMS business rules require that all document upload requests also provide the following fields:</p> <ul style="list-style-type: none"> • Veteran First Name • Veteran Middle Name (if available) • Veteran Last Name • Veteran File Number • New Mail Indicator • Date of Receipt (i.e., the date stamped on the document or the date the document was generated in the calling system. This is NOT the date of the document upload) • Document Source (VBMS-assigned name for document uploading institution) • Document Type • Document File Name • Document Content <p>The document file name should be unique for each upload operation, unless the document is identical to a previously uploaded document that must be reloaded for quality assurance purposes. Refer to the <i>VBMS PDF Specification Document</i> (VBMS-SPEC-00021) for details on these expected file naming conventions.</p>

Method Name	uploadDocumentWithAssociations
Required Fields	document.veteranFirstName document.veteranMiddleName document.veteranLastName document.fileNumber document.newMail document.receivedDt document.source document.docType document.filename documentContent.data
Message Exchange Pattern	In-Out
Return Parameter	uploadDocumentWithAssociationsResponse
Input Parameter	uploadDocumentWithAssociations
Error Codes:	vbmsServiceException
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="No record found with file number [fileNumber]"> or <faultDetailBean:faultDetailBean message="Veteran object missing fileNumber field"> or <faultDetailBean:faultDetailBean message="Claim does not exist for this file number">
Error Description	Possible Cause: fileNumber is provided in a valid format, but does not exist in the CorpDB Possible Cause: fileNumber is not provided Possible Cause: fileNumber is provided but is blank Possible Solution: Provide a valid fileNumber already existing in the CorpDB with at least one claim association
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="Unable to save document">
Error Description	Possible Cause: filename contains invalid characters Possible Cause: filename is not provided Possible Cause: filename is provided but is blank Possible Solution: Provide a valid file name for the VBMS eDocument Service and ensure it follows the naming conventions provided in the <i>VBMS PDF Specification Document</i> (VBMS-SPEC-00021)
Error Text	Fault Message: <faultDetailBean:faultDetailBean

Method Name	uploadDocumentWithAssociations
	exceptionClassName=" java.lang.NullPointerException">
Error Description	<p>Possible Cause: A <i>documentAssociations</i> element was provided with an empty <i>associationValue</i> element</p> <p>Possible Solution: Either do not provide <i>documentAssociations</i> elements, or ensure that each <i>documentAssociations</i> element has an <i>associationType</i> attribute of CLAIM or CONTENTION</p>
Error Text	<p>Fault Message:</p> <pre><faultstring>org.dozer.MappingException: Source object must not be null</faultstring></pre> <p>or</p> <pre><ns2:faultDetailBean exceptionClassName="gov.va.vba.vbms.bl.exceptions.BusinessException" message="Failed to saved document"></pre> <p>or</p> <pre><ns2:faultDetailBean exceptionClassName="gov.va.vba.vbms.bl.exceptions.BusinessException" message="Unknown file type for conversion. Document not saved."></pre>
Error Description	<p>Possible Cause: A <i>documentContent</i> element was not provided</p> <p>Possible Cause: A <i>documentContent</i> element was provided but a data element was not</p> <p>Possible Cause: A <i>documentContent</i> and data element were both provided; however, no <i>xop:Include</i> element was included referring to the MTOM attachment</p> <p>Possible Solution: Ensure that a complete <i>documentContent</i> element with all of the supporting sub-elements is specified</p>
Error Text	<p>Fault Message:</p> <pre><faultstring>java.lang.IllegalStateException: The current event is not START_ELEMENT but 4</faultstring></pre>
Error Description	<p>Possible Cause: The <i>xop:Include</i> element's href attribute refers to a nonexistent XOP attachment</p> <p>Possible Solution: Ensure the method your code is using to generate a MTOM attachment is conformant with the published MTOM specification at the w3.org web site: http://www.w3.org/TR/soap12-mtom/</p>
Error Text	<p>Fault Message:</p> <pre><faultDetailBean message="Invalid date for field receivedDt - Dates before Dec 31 19:00:00 EST 1752 are not allowed."></pre>
Error Description	<p>Possible Cause: The receivedDt set for the document is before the minimum date of 12/31/1752</p> <p>Possible Solution: Ensure that the receivedDt being set is after the minimum allowed date</p>

Table 16: uploadDocumentWithAssociations Request Element

Type	Input Parameter Name
cdm/document/v4:Document	document
List<cdm:DocumentAssociationInput>	documentAssociations

Type	Input Parameter Name
cdm/document/v4:DocumentContent	documentContent

Table 17: uploadDocumentWithAssociationsResponse Response Element

Type	Return Parameter Name
wsdl:ConvertedDocumentResponse	Result

Table 18: vbmsServiceException

Type	Parameter Name
xs:string	message
xs:string	uid
xs:boolean	serverException
xs:string	exceptionClassName

5.4.4.2 *uploadDocument*

```
xs:string uploadDocument (
    cdm:Veteran veteran,
    cdm/document/v4:Document document
    cdm/document/v4:DocumentContent documentContent
) throws VbmsServiceException
```

Table 19: uploadDocument

Method Name	uploadDocument
Description	<p>! DEPRECATED !</p> <p>This service operation has been deprecated in favor of the uploadDocumentWithAssociations operation.</p> <p>The uploadDocument operation provides basic document uploading capabilities for a Veteran's eFolder. The MTOM-streamed document content is referenced in the <i>Document</i> object and the file number referencing an eFolder is provided in the <i>Veteran</i> object. The Veteran MUST be retrievable from CorpDB by that file number in order to invoke a successful operation.</p> <p>VBMS business rules require each document upload request to also provide the following fields:</p> <ul style="list-style-type: none"> • Veteran First Name • Veteran Middle Name (if available) • Veteran Last Name • Veteran File Number • New Mail Indicator • Date of Receipt (date stamped on document or date document was generated within the calling system. NOT the date of the document upload)

Method Name	uploadDocument
	<ul style="list-style-type: none"> Document Source (VBMS assigned name for document uploading institution) Document Type Document File Name Document Content <p>The document File Name should be unique for each upload operation, unless the document is identical to a previously uploaded document being reloaded for quality assurance purposes. Refer to the <i>VBMS PDF Specification Document</i> (VBMS-SPEC-00021) for details on the expected file naming conventions.</p> <p>This service operation has been deprecated in favor of the uploadDocumentWithAssociations operation.</p> <p>! DEPRECATED !</p>
Required Fields	veteran.fileNumber document.veteranFirstName document.veteranMiddleName document.veteranLastName document.newMail document.receivedDt document.source document.docType document.filename document.docContent document.externalID documentContent.data
Message Exchange Pattern	In-Out
Return Parameter	uploadDocumentResponse
Input Parameter	uploadDocument
Error Codes:	vbmsServiceException
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="ShareException thrown in findCorporateRecordByFileNumber.">
Error Description	<p>Possible Cause: fileNumber is not provided</p> <p>Possible Cause: fileNumber is provided but is blank</p> <p>Possible Cause: fileNumber is provided, but contains invalid characters instead of digits only</p> <p>Possible Solution: Provide a valid fileNumber already existing in the CorpDB</p>
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="Document object missing externalID field">
Error Description	<p>Possible Cause: externalID is not provided</p> <p>Possible Cause: externalID is provided but is blank</p>

Method Name	uploadDocument
	Possible Solution: Provide an externalID attribute with a specified value
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="Document object missing filename field"> or <faultDetailBean:faultDetailBean message="Unable to save document">
Error Description	Possible Cause: fileName is not provided Possible Cause: fileName contains invalid characters Possible Solution: Provide a valid fileName that can be stored in the VBMS eDocument Service and ensure the file name is conformant with the <i>VBMS PDF Specification Document</i> (VBMS-SPEC-00021)
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="Document object missing docType field">
Error Description	Possible Cause: docType is not provided Possible Cause: docType is provided but is blank Possible Solution: Provide a valid docType that is returned from the getDocumentTypes operation
Error Text	Fault Message: <faultstring>org.dozer.MappingException: Source object must not be null</faultstring> or <ns2:faultDetailBean exceptionClassName="gov.va.vba.vbms.bl.exceptions.BusinessException" message="Failed to saved document"> or <ns2:faultDetailBean exceptionClassName="gov.va.vba.vbms.bl.exceptions.BusinessException" message="Unknown file type for conversion. Document not saved.">
Error Description	Possible Cause: A <i>documentContent</i> element was not provided Possible Cause: A <i>documentContent</i> element was provided but no data element was provided within it. Possible Cause: A <i>documentContent</i> and data element were both provided, but no <i>xop:Include</i> element was provided that referred to the MTOM attachment Possible Solution: Ensure that a complete <i>documentContent</i> element with all of the supporting sub-elements is specified
Error Text	Fault Message: <faultstring>java.lang.IllegalStateException: The current event is not START_ELEMENT but 4</faultstring>
Error Description	Possible Cause: The <i>xop:Include</i> element's href attribute refers to a nonexistent XOP attachment Possible Solution: Ensure that the method your code is using to generate MTOM

Method Name	uploadDocument
	attachments properly conforms to the MTOM specification
Error Text	Fault Message: <faultDetailBean message="Invalid date for field receivedDt - Dates before Dec 31 19:00:00 EST 1752 are not allowed.">
Error Description	Possible Cause: The receivedDt set for the document is before the minimum date of 12/31/1752 Possible Solution: Ensure that the receivedDt being set is after the minimum allowed date

Table 20: uploadDocument Request Element

Type	Input Parameter Name
cdm:Veteran	veteran
cdm/document/v4:Document	document
cdm/document/v4:DocumentContent	documentContent

Table 21: uploadDocumentResponse Response Element

Type	Return Parameter Name
xs:string	message

Table 22: vbmsServiceException

Type	Parameter Name
xs:string	message
xs:string	uid
xs:boolean	serverException
xs:string	exceptionClassName

5.4.4.3 fetchDocumentById (UPDATED)

```
wsdl:DocumentWithContentResponse fetchDocumentById (
    xs:string documentId
) throws vbmsServiceException
```

Table 23: fetchDocumentById

Method Name	fetchDocumentById
Description	<p>The fetchDocumentById operation returns a <i>DocumentWithContentResponse</i> object from the provided documentId. The actual document content is returned in an MTOM-streamed attachment. The documentId may be a document identifier returned from the uploadDocumentWithAssociations, listDocuments, or listDocumentsByDocumentTypeId operation.</p> <p>If no document can be found with the provided document identifier, a</p>

Method Name	fetchDocumentById
	VbmsServiceException SOAP fault is returned.
Required Fields	documentId
Message Exchange Pattern	In-Out
Return Parameter	fetchDocumentResponse
Input Parameter	fetchDocumentById
Error Codes:	vbmsServiceException
Error Text	Fault Message: <faultstring>Could not get document with content for docID: null</faultstring> or <faultstring>Could not get document with content for docID: [documentId]</faultstring>
Error Description	Possible Cause: documentId is provided and in a valid format, but does not exist in the eDocument Service Version 4.0 Possible Cause: documentId is not provided Possible Cause: documentId is provided but is blank Possible Solution: Provide a valid documentId existing in eDocument Service Version 4.0

Table 24: fetchDocumentById Request Element

Type	Input Parameter Name
xs:string	documentId

Table 25: fetchDocumentResponse Response Element

Type	Return Parameter Name
wsdl:DocumentWithContentResponse	result

Table 26: vbmsServiceException

Type	Parameter Name
xs:string	message
xs:string	uid
xs:boolean	serverException
xs:string	exceptionClassName

5.4.4.4 fetchDocument

```
wsdl:DocumentWithContentResponse fetchDocument (
    cdm:Veteran veteran
    cdm/document/v4:Document query
) throws ServiceException, VbmsWssException
```

Table 27: fetchDocument

Method Name	fetchDocument
Description	The fetchDocument operation returns a <i>DocumentWithContentResponse</i> object from the fileNumber associated with a <i>Veteran</i> object and an externalId field associated with the <i>Document</i> object. Actual document content is returned in an MTOM-streamed attachment. The externalId must be the same externalId provided on input to the uploadFormDocument or the uploadDocument operation's Document input parameter.
Required Fields	veteran.fileNumber query.externalId
Message Exchange Pattern	In-Out
Return Parameter	fetchDocumentResponse
Input Parameter	fetchDocument
Error Codes:	vbmsServiceException
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="Document object missing externalID field"> or <faultDetailBean:faultDetailBean message="No document found matching file number [fileNumber] and external ID [externalId]">
Error Description	Possible Cause: externalId is provided and in a valid format, but does not exist in the VBMS eDocument Service for the specified fileNumber Possible Cause: externalId is not provided Possible Cause: externalId is provided but is blank Possible Solution: Provide a valid externalId that exists in the VBMS eDocument Service for the specified fileNumber
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="Veteran object missing fileNumber field"> or <faultDetailBean:faultDetailBean message="No document found matching file number [fileNumber] and external ID [externalId]">
Error Description	Possible Cause: fileNumber is provided and in a valid format, but does not exist in the VBMS eDocument Service Possible Cause: fileNumber is not provided Possible Cause: fileNumber is provided but is blank Possible Solution: Provide a valid fileNumber that exists in the VBMS eDocument Service

Table 28: fetchDocument Request Element

Type	Input Parameter Name
cdm:Veteran	Veteran
cdm/document/v4:Document	Query

Table 29: fetchDocumentResponse Response Element

Type	Return Parameter Name
cdm/document/v4:DocumentWithContentResponse	Result

Table 30: vbmsServiceException

Type	Parameter Name
xs:string	message
xs:string	uid
xs:boolean	serverException
xs:string	exceptionClassName

5.4.4.5 uploadFormDocument (UPDATED)

```
xs:string uploadFormDocument (
```

```
    cdm/document/v4:FormDocument formDocument
```

```
    cdm/document/v4:DocumentContent documentContent
```

```
) throws VbmsServiceException, VbmsDataValidationException
```

Table 31: uploadFormDocument

Method Name	uploadFormDocument
Description	<p>The uploadFormDocument operation allows an MTOM-streamed document to be uploaded into the VBMS content store along with any associated <i>FormField</i> elements collected from the end-user or via OCR from the uploaded document. A list of valid form fields for the <i>FormDocument</i> element's specified docType attribute may be retrieved from the getFormFieldsForDocumentType service operation.</p> <p>In certain instances, a valid form field for the specified document type overlaps with a similarly named attribute of the <i>FormDocument</i> element. In these instances, the form field value will take precedence over the form document attribute value when stored in the VBMS eDocument Service. Known form fields that supersede document attributes are:</p> <ul style="list-style-type: none"> • Veteran First Name • Veteran Middle Name

Method Name	uploadFormDocument
	<ul style="list-style-type: none"> • Veteran Last Name • Document Type • File Number <p>VBMS business rules require each document upload request provide the following fields:</p> <ul style="list-style-type: none"> • Veteran First Name • Veteran Middle Name (if available) • Veteran Last Name • Veteran File Number • New Mail Indicator • Date of Receipt (date stamped on document or date document was generated within the calling system, NOT the date of the document upload) • Document Source (VBMS assigned name for document uploading institution) • Document Type • Document File Name • DCS Id • DCS Scanning Complete indicator • Document Content <p>The document file name should be unique for each upload operation unless the document being uploaded is the same as a previously uploaded document being reloaded for quality assurance purposes. Refer to the <i>VBMS PDF Specification Document</i> (VBMS-SPEC-00021) for details on the expected file naming conventions.</p> <p>The uploadFormDocument operation requires that an eFolder for the specified File Number already be created within the VBMS content store. If the eFolder does not exist, a VbmsServiceException SOAP fault will be returned.</p>
Required Fields	<p>document.veteranFirstName (or corresponding <i>FormField</i> element)</p> <p>document.veteranMiddleName (or corresponding <i>FormField</i> element)</p> <p>document.veteranLastName (or corresponding <i>FormField</i> element)</p> <p>document.fileNumber (or corresponding <i>FormField</i> element)</p> <p>document.newMail</p> <p>document.receivedDt</p> <p>document.source</p> <p>document.docType (or corresponding <i>FormField</i> element)</p> <p>document.filename</p> <p>document.dcsId</p> <p>document.dcsScanningComplete</p> <p>documentContent.data</p>
Message Exchange Pattern	In-Out

Method Name	uploadFormDocument
Return Parameter	uploadFormDocumentResponse
Input Parameter	uploadFormDocument
Error Codes:	vbmsServiceException, vbmsDataValidationException
Error Text	Fault Message: <faultstring>org.springframework.context.NoSuchMessageException: No message found under code 'missing.requiredField' for locale 'en_US'.</faultstring> or <faultstring>java.lang.NullPointerException</faultstring>
Error Description	Possible Cause: Some required fields were left blank Possible Solution: Please ensure that all required fields listed in the "Required Fields" row are specified on the request
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="Form field [formFields.name] is not supported"/>
Error Description	Possible Cause: A specified form field is not valid for the document type being uploaded Possible Solution: Ensure all form fields that are passed have a name that is on the list of valid form fields for the uploaded document type. The list of valid form fields for a particular document type may be retrieved from the getFormFieldsForDocumentType service operation.
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="Failed to saved document"/>
Error Description	Possible Cause: No document content was specified in the request Possible Solution: Ensure that document content is passed within the data element of the <i>DocumentContent</i> object
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="Unable to save document"/>
Error Description	Possible Cause: Invalid filename was specified in the request Possible Cause: No filename was specified in the request Possible Solution: Please ensure that a valid file name is passed for the filename attribute of the <i>Document</i> object
Error Text	Fault Message: <faultDetailBean:faultDetailBean message="Invalid data

Method Name	uploadFormDocument
	for form field FileNumber"/>
Error Description	<p>Possible Cause: Invalid fileNumber was specified in the request</p> <p>Possible Cause: No filename was specified in the request</p> <p>Possible Solution: Please ensure that a valid and properly formatted fileNumber is passed for the fileNumber attribute of the <i>Document</i> object</p>
Error Text	<p>Fault Message:</p> <pre><faultstring>org.dozer.MappingException: Source object must not be null</faultstring></pre> <p>or</p> <pre><ns2:faultDetailBean exceptionClassName="gov.va.vba.vbms.bl.exceptions.Busines sException" message="Failed to saved document"></pre> <p>or</p> <pre><ns2:faultDetailBean exceptionClassName="gov.va.vba.vbms.bl.exceptions.Busines sException" message="Unknown file type for conversion. Document not saved."></pre>
Error Description	<p>Possible Cause: A <i>DocumentContent</i> element was not provided</p> <p>Possible Cause: A <i>documentContent</i> element was provided but it was blank</p> <p>Possible Cause: A <i>documentContent</i> and data element were both provided, but no <i>xop:Include</i> element was provided that referred to the MTOM attachment</p> <p>Possible Solution: Ensure that a complete <i>documentContent</i> element with all of the supporting sub-elements is specified</p>
Error Text	<p>Fault Message:</p> <pre><faultstring>java.lang.IllegalStateException: The current event is not START_ELEMENT but 4</faultstring></pre>
Error Description	<p>Possible Cause: The <i>xop:Include</i> element's href attribute refers to a nonexistent XOP attachment</p> <p>Possible Solution: Ensure that the method your code is using to generate MTOM attachments properly conforms to the MTOM specification</p>
Error Text	<p>Fault Message:</p> <pre><faultDetailBean message="Invalid data for form field "DateofReceipt" - DateofReceipt must be in MM/dd/yyyy format."></pre> <p>or</p> <pre><faultDetailBean message="Invalid data for form field "DateofReceipt" - Dates before Dec 31 19:00:00 EST 1752 are not allowed."></pre>

Method Name	uploadFormDocument
Error Description	<p>Possible Cause: The DateofReceipt form field is not in the correct format of mm/dd/yyyy.</p> <p>Possible Cause: The DateofReceipt form field is set to a date before the minimum date of 12/31/1752.</p> <p>Possible Solution: Ensure that the DateofReceipt form field is in the correct format and the date being set is after the minimum allowed date.</p>
Error Text	<p>Fault Message:</p> <p><faultDetailBean message="Invalid date for field receivedDt - Dates before Dec 31 19:00:00 EST 1752 are not allowed."></p>
Error Description	<p>Possible Cause: The receivedDt set for the document is before the minimum date of 12/31/1752</p> <p>Possible Solution: Ensure that the receivedDt being set is after the minimum allowed date</p>

Table 32: uploadFormDocument Request Element

Type	Input Parameter Name
cdm/document/v4:FormDocument	formDocument
cdm/document/v4:DocumentContent	documentContent

Table 33: uploadFormDocumentResponse Response Element

Type	Return Parameter Name
xs:string	message

Table 34: vbmsServiceException

Type	Parameter Name
xs:string	message
xs:string	uid
xs:boolean	serverException

5.4.4.6 getFormFieldsForDocumentType

```
List<cdm:FormField> getFormFieldsForDocumentType (
    xs:string documentType
) throws VbmsServiceException
```

Table 35: getFormFieldsForDocumentType

Method Name	getFormFieldsForDocumentType
Description	<p>The getFormFieldsForDocumentType operation returns a list of valid form fields that may be submitted when uploading a particular document type document passed in the documentType string on input. Available document types that may be specified on input can be retrieved from the getDocumentTypes operation.</p>

Method Name	getFormFieldsForDocumentType
Required Fields	documentType
Message Exchange Pattern	In-Out
Return Parameter	getFormFieldsForDocumentTypeResponse
Input Parameter	getFormFieldsForDocumentType
Error Codes:	vbmsServiceException
Error Text	Fault Message: <faultstring>Could not get form fields for document type</faultstring>
Error Description	Possible Cause: Invalid documentType was specified in the request Possible Cause: documentType was not specified in the request Possible Solution: Ensure that a valid documentType is passed according to the results from the getDocumentTypes service operation

Table 36: getFormFieldsForDocumentType Request Element

Type	Input Parameter Name
xs:string	documentType

Table 37: getFormFieldsForDocumentTypeResponse Response Element

Type	Return Parameter Name
List<cdm:FormField>	result

Table 38: vbmsServiceException

Type	Parameter Name
xs:string	message
xs:string	uid
xs:boolean	serverException
xs:string	exceptionClassName

Table 39: vbmsDataValidationException

Type	Parameter Name
List<wsdl:FormFieldError>	formFieldErrors

Table 40: validationError

Type	Parameter Name
xs:string	formField
xs:string	errorCode
xs:string	errorMessage

5.4.4.7 uploadLetterDocument

cdm/document/v4:LetterDocument document
cdm/document/v4:DocumentContent documentContent

) throws VbmsServiceException

Table 41: uploadLetterDocument

Method Name	uploadLetterDocument
Description	Note: The uploadLetterDocument operation was built for a specific VBMS use case and should not be used for general document uploading. Your VBMS Point of Contact (POC) can provide more information if you believe you should be uploading documents using this service operation.
Required Fields	document.fileNumber document.docType document.filename documentContent.data
Message Exchange Pattern	In-Out
Return Parameter	uploadLetterDocumentResponse
Input Parameter	uploadLetterDocument
Error Codes:	vbmsServiceException, vbmsDataValidationException
Error Text	Fault Message: <faultstring>For input string: "?"</faultstring> or <faultDetailBean:faultDetailBean message="Error getting Corp DB Veteran by File Number"/> or <faultDetailBean:faultDetailBean message="Failed to saved document"/>
Error Description	Possible Cause: fileNumber is not provided Possible Cause: fileNumber is provided but is blank Possible Cause: fileNumber is provided, but contains invalid characters instead of digits only Possible Solution: Provide a valid fileNumber that already exists in the CorpDB Possible Cause: filename is not provided Possible Cause: filename contains invalid characters Possible Solution: Provide a valid filename that can be stored within the VBMS eDocument Service and ensure that the file name follows standard conventions for the filename value (containing only alphanumeric characters, underscores, hyphens and periods)

Method Name	uploadLetterDocument
Error Text	Fault Message: <faultstring>org.dozer.MappingException: Source object must not be null</faultstring> or <ns2:faultDetailBean exceptionClassName="gov.va.vba.vbms.bl.exceptions.BusinessException" message="Failed to saved document"> or <ns2:faultDetailBean exceptionClassName="gov.va.vba.vbms.bl.exceptions.BusinessException" message="Unknown file type for conversion. Document not saved.">
Error Description	Possible Cause: A <i>documentContent</i> element was not provided Possible Cause: A <i>documentContent</i> element was provided but no data element was provided within it Possible Cause: A <i>documentContent</i> and data element were both provided, but no <i>xop:Include</i> element was provided that referred to the MTOM attachment Possible Solution: Ensure that a complete <i>documentContent</i> element with all of the supporting sub-elements is specified
Error Text	Fault Message: <faultstring>java.lang.IllegalStateException: The current event is not START_ELEMENT but 4</faultstring>
Error Description	Possible Cause: The <i>xop:Include</i> element's href attribute refers to a nonexistent XOP attachment Possible Solution: Ensure that the method your code is using to generate MTOM attachments properly conforms to the MTOM specification
Error Text	Fault Message: <faultDetailBean message="Invalid date for field receivedDt - Dates before Dec 31 19:00:00 EST 1752 are not allowed.">
Error Description	Possible Cause: The receivedDt set for the document is before the minimum date of 12/31/1752 Possible Solution: Ensure that the receivedDt being set is after the minimum allowed date

Table 42: uploadLetterDocument Request Element

Type	Input Parameter Name
cdm/document/v4:LetterDocument	Document
cdm/document/v4:DocumentContent	documentContent

Table 43: uploadLetterDocumentResponse Response Element

Type	Return Parameter Name
xs:string	message

Table 44: vbmsServiceException

Type	Parameter Name
xs:string	message
xs:string	uid
xs:boolean	serverException
xs:string	exceptionClassName

5.4.4.8 *getDocumentTypes*

```
List<cdm:DocumentType> getDocumentTypes ( )
```

Table 45: getDocumentTypes

Method Name	getDocumentTypes
Description	The getDocumentTypes operation returns a list of valid document types that may be uploaded into the eDocument Service Version 4.0. Each document type may also be used as input to the getFormFieldsForDocumentType operation in order to retrieve a list of valid form fields that may be provided to the uploadFormDocument operation.
Required Fields	N/A
Message Exchange Pattern	In-Out
Return Parameter	getDocumentTypesResponse
Input Parameter	getDocumentTypes
Error Codes:	N/A

Table 46: getDocumentTypes Request Element

Type	Input Parameter Name
N/A	N/A

Table 47: getDocumentTypesResponse Response Element

Type	Return Parameter Name
List<cdm:DocumentType>	Result

5.4.4.9 *listDocuments*

```
List<cdm/document/v4:Document> listDocuments (
    xs:string fileName,
    List<cdm:DocumentAssociationInput> documentAssociations
)
```

Table 48: listDocuments

Method Name	listDocuments
-------------	---------------

Method Name	listDocuments
Description	<p>The listDocuments operation returns a list of document identifiers that may be retrieved from the eDocument Service Version 4.0 for the specified Veteran file number. The list may be filtered by optionally provided document associations on input. In eDocument Service Version 4.0, the only associationType values currently accepted are 'CLAIM', 'CONTENTION', and 'DOCUMENT'.</p> <p>Please note that if a fileNumber is not specified or is invalid, this operation will return an empty list of documents, rather than throwing an exception. The same is true if a <i>documentAssociations</i> element is present but the corresponding associationType or associationValue fields are empty or invalid.</p>
Required Fields	fileNumber
Message Exchange Pattern	In-Out
Return Parameter	listDocumentsResponse
Input Parameter	listDocuments
Error Codes:	N/A
Error Text	
Error Description	

Table 49: listDocuments Request Element

Type	Input Parameter Name
xs:string	fileNumber
List<cdm:DocumentAssociationInput>	documentAssociations

Table 50: listDocumentsResponse Response Element

Type	Return Parameter Name
List<cdm/document/v4:Document>	result

Table 51: vbmsServiceException

Type	Parameter Name
xs:string	message
xs:string	uid
xs:boolean	serverException
xs:string	exceptionClassName

5.4.4.10 listDocumentsByDocumentTypeId

List<cdm/document/v4:Document> listDocumentsByDocumentTypeId (

xs:string fileNumber,

xs:string documentTypeId
)

Table 52: listDocumentsByDocumentTypeId

Method Name	listDocumentsByDocumentTypeId
Description	<p>The listDocumentsByDocumentTypeId operation returns a list of document identifiers that may be retrieved from the eDocument Service Version 4.0 for the specified Veteran file number and the specified document type Id. In eDocument Service Version 4.0, all documentTypeId values that are returned by the getDocumentTypes operation are accepted.</p> <p>Note: If a fileNumber or documentTypeId is not specified, the operation will throw a VBMS Service Exception. If the fileNumber or documentTypeId is specified but not valid, this operation will return an empty list of documents, rather than throwing an exception.</p>
Required Fields	fileNumber documentTypeId
Message Exchange Pattern	In-Out
Return Parameter	listDocumentsByDocumentTypeId Response
Input Parameter	listDocumentsByDocumentTypeId
Error Codes:	N/A
Error Text	
Error Description	

Table 53: listDocumentsByDocumentTypeId Request Element

Type	Input Parameter Name
xs:string	fileNumber
xs:string	documentTypeId

Table 54: listDocumentsByDocumentTypeIdResponse Response Element

Type	Return Parameter Name
List<cdm/document/v4:Document>	result

Table 55: vbmsServiceException

Type	Parameter Name
xs:string	message
xs:string	uid
xs:boolean	serverException
xs:string	exceptionClassName

5.5 Policies and Qualities of Service

The following sub-sections describe the policies and qualities of service attributes for the eDocument Service Version 4.0. These policies are focused on ensuring the eDocument Service conforms to the currently defined VBMS security policies.

5.5.1 Service Policies

To conform to VBMS security standards, the following guidelines are in place for system-to-system interactions:

- **Double Encryption** – Messages are encrypted at least twice. The first level of encryption is at the transport layer using two-way Secure Socket Layer (SSL) between the calling application and the service host. The second level of encryption is at the application layer using standard WSS message encryption techniques. More detail is provided in Section 6.1.
- **Request and Response Signing** – Both request and response messages are digitally signed by the message sender to achieve non-repudiation and to verify message integrity. This signature is for the timestamp in the message header and for all contents of the message body EXCEPT for the streaming document reference in the *documentContent* element. An additional signature exists within the Security Assertion Markup Language (SAML) assertion in the message headers but is only used to verify authenticity of the SAML assertion. Standard XML (eXtensible Markup Language) digital signatures are used for signing SOAP messages.
- **User or System Authentication via SAML** – SAML assertions validate prior authentication of end-users or systems that invoke any VBMS Web Service operations. When interactions are driven by end-user actions with proper credentials, the provided SAML assertion is on behalf of the currently logged in user, and NOT on behalf of the calling system. A SAML assertion received by the calling application during a Web Single Sign On (SSO) interaction with the Business Enterprise Platform (BEP) Identity Provider (IdP) should be reused and sent in the WSS headers.

5.5.2 SAML Assertion Guidelines

The following attribute guidelines must be followed for each SAML assertion delivered to a VBMS Web Service. Attributes delivered within the SAML assertion include the following elements needed for VBMS Web Services SAML-based authentication (XML namespace references have been removed from the following examples to improve readability):

- **Subject.NameID** – Identifies either the username assigned to the application end-user or the calling system if the username is not meaningful in VBMS.
Example:
`<NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">VAEBENEFITS</NameID>`
- **AudienceRestriction.Audience** – Identifies the system where the SAML token was issued.
Example:
`<Audience>http://claims01.p2.vbms.va.gov:7001/vbmosp2</Audience>`
- **Station Id** – Identifies the station the end-user has authenticated into for the calling application. The Station Id value is "0" for a system-level SAML token.
Example:

```
<Attribute Name="http://vba.va.gov/css/common/stationId"
FriendlyName="stationId">
  <AttributeValue type="string">317</AttributeValue>
</Attribute>
```

- **Security Level** – Identifies the end-user security level for the individual initiating the request from the calling application. The security level is “0” for a system-level SAML token.

Example:

```
<Attribute Name="http://vba.va.gov/css/common/securityLevel"
FriendlyName="securityLevel">
  <AttributeValue type="string">7</AttributeValue>
</Attribute>
```

- **Role(s)** – Identifies zero or more roles granted either end-users initiating requests from the calling application or to the calling system if the end-user credential is not meaningful to VBMS. If no roles are specified, at least one Service Operation attribute must be provided.

Example:

```
<Attribute Name="http://vba.va.gov/css/vbms-core/role"
FriendlyName="role">
  <AttributeValue type="string">Rater </AttributeValue>
</Attribute>
<Attribute Name="http://vba.va.gov/css/vbms-r/role"
FriendlyName="role">
  <AttributeValue type="string">Rater/Trainee</AttributeValue>
</Attribute>
```

- **Service Operation(s)** – Identifies zero or more roles granted either to end-users initiating requests from the calling application or to the calling system if the end-user credential is not meaningful to VBMS. If no service operations are specified, at least one role attribute must be provided. Example:

```
<Attribute Name="http://vba.va.gov/vbms/service-operation">
  <AttributeValue type="string">Upload Form Document
Service</AttributeValue>
</Attribute>
<Attribute Name="http://vba.va.gov/vbms/service-operation">
  <AttributeValue type="string">Get Document Types
Service</AttributeValue>
</Attribute>
<Attribute Name="http://vba.va.gov/vbms/service-operation">
  <AttributeValue type="string">Get Form Fields For Document Type
Service</AttributeValue>
</Attribute>
```

When needed, a VBMS Web Services administrator provides a long-lasting SAML token for a service client that is unable to interact with a mutually trusted identity provider, such as the BEP IdP.

5.5.3 Performance

Information regarding availability, reliability, throughput, and response time will be described further in a forthcoming service contract document. Service levels may vary between services due to specific service technical and functional characteristic differences, and varying requirements.

5.5.4 Business Metrics

Currently no business metrics are captured or published by this service.

5.5.5 Standards Compliance

Table 56: Standards Compliance

Standard	Version	Comments
In-house Standards	N/A	
Technical Standards	N/A	
Business Standards	N/A	
Regulatory Standards	N/A	
Security Standards	WSS, SAML v2	
SOAP	1.2	
Transport Layer Security / Secure Sockets Layer (TLS/SSL)	1.0/3.0 Respectively	

6. Security

As described in Section 5.5.1, Web Service message contents are secured using specific signing, encryption, and SAML-injection techniques each service client must adhere to. A VBMS administrator provides the files referenced in the following sub-sections to each service client to securely access service operations in a specific VBMS environment. Details are provided in the following sections regarding the techniques used to secure Web Service messages. Additional detail is found in *VBMS Web Services – Getting Started Guide for Clients* (VBMS-GUI-00036).

6.1 Message-Level Security

All messages received and sent by VBMS Web Services are signed and encrypted using certificates and private keys specific to both VBMS Web Services and the calling application. The certificates and keys used for message encryption and signature generation vary across VBMS deployment environments, as do the certificate authorities (CAs).

Note: When importing VBMS certificates into the client keystore/truststore files, you must also import the corresponding CA certificates. Neglecting to import the CA certificates is a common Web Service client error that increases time to successfully interact with VBMS Web Services.

The following steps must be performed by the calling application against the raw SOAP request message before request submission:

1. Insert a WSS header into the message by adding a WSS timestamp to the message header.
2. Sign both the timestamp in the security header and all portions of message body EXCEPT for the *documentContent* element using the VBMS-provided client private key.

3. Encrypt the body of the message using the VBMS server certificate provided by a VBMS administrator.

WARNING: DO NOT use the client certificate to encrypt the message body...this common error slows down Web Service client integration efforts.

4. Add a SAML assertion obtained through Web SSO (VBMS applications) or provided by a VBMS administrator (non-VBMS applications).
5. Remove the MustUnderstand attribute from the security header.

A somewhat shorter reverse sequence of steps must be performed when parsing the response message sent by the eDocument Service Version 4.0:

6. Decrypt the body of the message using the private key that was used to sign the message request and that is associated with the certificate shared with VBMS system administrators.
7. Verify the signature of the timestamp and message body using the VBMS server certificate provided by a VBMS administrator.
8. Verify the timestamp of the security header is within the acceptable margin for error (+/- 10 minutes).

6.2 Document Content References

To maintain compatibility with a larger number of Web Service application frameworks, the *documentContent* element of the message body must neither be signed nor encrypted. The following sub-sections provide tables that list the specific elements and namespaces to use for signing message headers and for signing and encrypting the message body for service operations that upload documents into VBMS.

Any operation not listed in this section should have its entire message body element signed and encrypted as was previously done in the original eDocument Service.

6.2.1 uploadFormDocument

Table 57: uploadFormDocument Signature

Name	Namespace	Encode
Timestamp	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	Element
formDocument	http://vbms.vba.va.gov/external/eDocumentService/v4	Element

Table 58: uploadFormDocument Encryption

Name	Namespace	Encode
formDocument	http://vbms.vba.va.gov/external/eDocumentService/v4	Element

6.2.2 uploadDocument

Table 59: uploadDocument Signature

Name	Namespace	Encode
Timestamp	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	Element
document	http://vbms.vba.va.gov/external/eDocumentService/v4	Element
veteran	http://vbms.vba.va.gov/external/eDocumentService/v4	Element

Table 60: uploadDocument Encryption

Name	Namespace	Encode
document	http://vbms.vba.va.gov/external/eDocumentService/v4	Element
veteran	http://vbms.vba.va.gov/external/eDocumentService/v4	Element

6.2.3 uploadDocumentWithAssociations

Table 61: uploadDocumentWithAssociations Signature

Name	Namespace	Encode
Timestamp	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	Element
document	http://vbms.vba.va.gov/external/eDocumentService/v4	Element
documentAssociations (only if provided)	http://vbms.vba.va.gov/external/eDocumentService/v4	Element

Table 62: uploadDocumentWithAssociations Encryption

Name	Namespace	Encode
document	http://vbms.vba.va.gov/external/eDocumentService/v4	Element
documentAssociations (only if provided)	http://vbms.vba.va.gov/external/eDocumentService/v4	Element

6.2.4 uploadLetterDocument

Table 63: uploadLetterDocument Signature

Name	Namespace	Encode
Timestamp	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	Element
document	http://vbms.vba.va.gov/external/eDocumentService/v4	Element

Table 64: uploadLetterDocument Encryption

Name	Namespace	Encode
document	http://vbms.vba.va.gov/external/eDocumentService/v4	Element

6.3 SAML Token Injection

The eDocument Service Version 4.0 recognizes BEP-generated SAML tokens for authentication on behalf of end-users of client applications. The interaction method varies based on the type of service consumer:

- **Interactions Driven by End-users of the VBMS Web Applications** – VBMS web applications that invoke eDocument Service Version 4.0 are expected to integrate with the trusted IdP within the BEP infrastructure (i.e., the BEP IdP). The IdP collects the username and password from the end-user and delivers a SAML token back to the client application confirming the user's application authentication. The client application must inject this SAML token into the SOAP request security headers. The web service tests the validity of the SAML token received from the client application, and either allows access to the requested service operation or returns a SOAP fault to the client application.
- **System-to-system Interactions with Non-VBMS Applications** – A VBMS system administrator provides a client system-specific long-lasting SAML token to the administrators of each remote system (National Archives and Records Administration [NARA], VonApp Direct Connect [VDC], etc.). When invoking the desired web service operation, the remote system must inject this SAML token into the SOAP request security headers, and the service will process the token in a similar manner as described in the previous bullet. System level SAML tokens are correlated with the public/private key pair issued to the service consumer, ensuring the consumer has submitted the correct SAML token with each request.

Figure 4 shows the flow of information for both interaction styles:

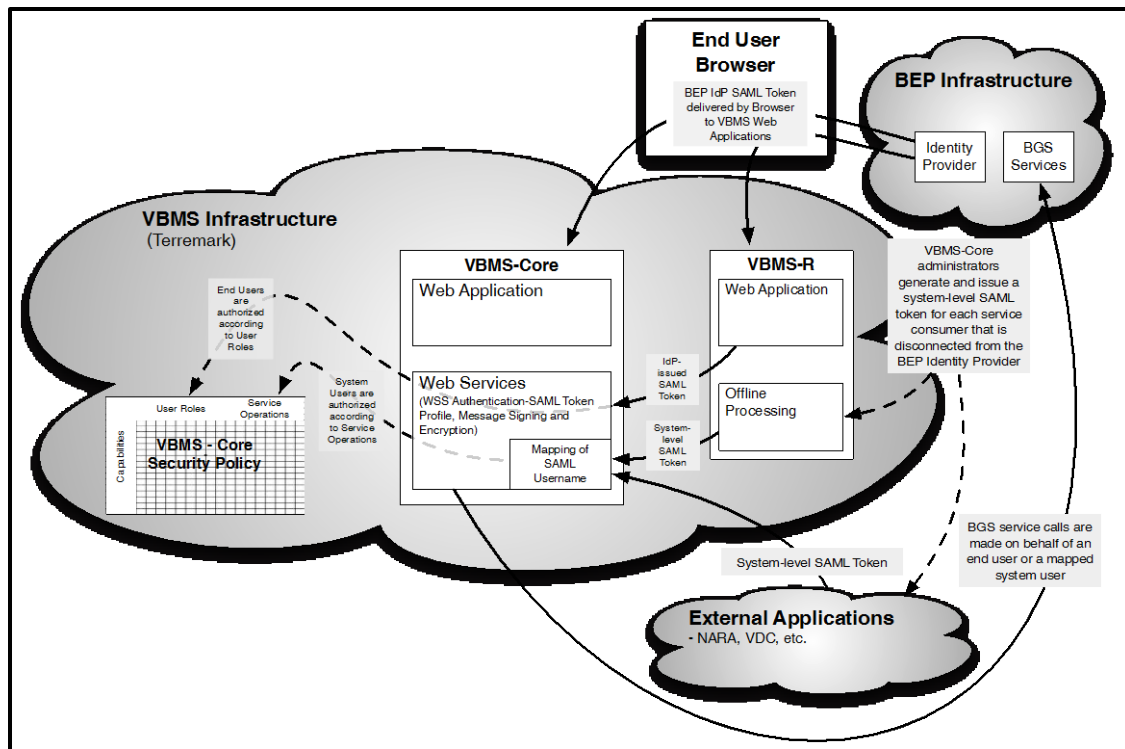


Figure 4: eDocumentService Credential Flow

Appendix

Appendix A: Acronym List

Acronym	Definition
BEP	Benefits Enterprise Platform
BGS	Benefits Gateway Services
C & P	Compensation and Pension
CA	Certificate Authorities
CDM	Common Data Model
COTS	Commercial Off-the-Shelf Software
DBQ	Disability Benefit Questionnaire
DCS	Document Control Sheet
HTTP	Hypertext Transfer Protocol
IAA	Interagency Agreement
IdP	Identity Provider
IM/IT	Information Management/Information Technology
IT	Information Technology
MTOM	Message Transmission Optimization Mechanism
NARA	National Archives and Records Administration
OCR	Optical Character Recognition
OIT	Office of Information and Technology
PD	Product Development
PDF	Portable Document Format
PMO	Program Management Office
POC	Point of Contact
SAML	Security Assertion Markup Language
SDD	Service Description Document
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPAWARSSYSCEN	Space and Naval Warfare Systems Center
SSL	Secure Sockets Layer
SSO	Single Sign-on
VA	Veterans Administration
VBA	Veterans Benefits Administration
VBMS	Veterans Benefit Management System
VDC	VonApp Direct Connect
WSDL	Web Services Description Language

Acronym	Definition
WSS	Web Service Security
XML	Extensible Markup Language

Appendix B: Approval Signatures

This section is used to record the approval of the *VBMS eDocument Service Version 4.0 Service Description Document* during the Formal Review. Conduct the review face-to-face where signatures can be obtained 'live' during the review. If unable to conduct a face-to-face meeting, conduct the review via a teleconferencing medium and capture concurrence during the meeting. The Scribe should add names by each position cited.

The following members governing the document are required to sign. Please annotate signature blocks accordingly.

REVIEW DATE:

SCRIBE:

Signed:

Date:

James Barr

SPAWAR Project Manager

Signed:

Date:

Clayton Coleman

SPAWAR Lead System Engineer