UNITED STATES
DEPARTMENT OF VETERANS AFFAIRS

Department of Veterans Affairs

Benefits Integration Platform

Veterans Benefits Management System (VBMS)

eFolder v1.0 Service Contract Document

Release 15.1

Task Order 0005, Benefits Integration Platform (BIP)

T4NG Prime Contract No: VA118-16-D-1007

(PWS 5.2.2.d)

Version 8.0

August 2018

Submitted to:

Ryan VanVickle
Department of Veterans Affairs
OIT EPMO VBMS
ryan.vanvickle@va.gov
(202) 280-8409

Submitted by:

Booz Allen Hamilton Inc.
575 Herndon Parkway
Herndon, VA 20170
(571) 346-4000
(FAX) 346-4010

Booz | Allen | Hamilton

Revision History

Date	Version	Description	Author
08/03/2018	8.0	Release 15.1 – no content updates.	Booz Allen Hamilton
05/11/2018	7.0	Release 15.0 – Added findPagedDocumentSeriesReferences method to the ReadService, supported by the PagingCriteria, PagingReference, PagedDocumentSeriesReferences information models.	Booz Allen Hamilton
02/16/2018	6.0	Release 14.1 – Added Package Service Interface section; added the Error Types and Codes table; updates to Allowable Content Source Values; updates to document order and structure including modified section numbering	Booz Allen Hamilton
11/10/2017	5.0	Release 14.0 – Additions to initializeUpdate to include error message text, additions to Upload Service Accepted Metadata table; corrected positive offset to UTC time in Appendix D	Booz Allen Hamilton
08/18/2017	4.0	Release 13.1 – added service deprecation language to the Introduction section	Booz Allen Hamilton
05/26/2017	3.0	Updates to legacy matching in 4.3; updates to SearchCriteria information in 5.2.8, 5.2.10, 5.2.18, and 5.2.27; added metadata information in 5.2.9.1, Table 13 and 6.3.3.1, Table 35; added MovedItem and related in 5.2.23 and 6.3.11; updates to annotations description in 6.3.6; updates to accepted metadata in 7.4 including Tables 63 and 64; added MIME type matching in 7.5.1; description additions in 8.1 and 8.1.1; and added associateContention in 8.3.4; Numbering and editorial corrections throughout.	Booz Allen Hamilton
03/20/2017	2.2	Updates to introductory text in sections 5.2.23, 7.3.3, 10.3.2 & 10.3.3. Modifications to the source description in Table 12, the out description in Table 39 and the in notes for documentVersionReferenceID in Table 48. Minor updates to advance release version number throughout.	Booz Allen Hamilton
03/07/2017	2.1	Modifications to Sections 7.3 and 7.3.2 to reflect recommended file size for uploaded documents	Booz Allen Hamilton
03/03/2017	2.0	Updates to 5.2.26 (Table 26) to reflect supported metadata keys and 6.3.10 (Table 37) to clarify the use of DateTimeRange. Added Section 8: Upload eFolder Service Accepted Metadata and Section 9: Upload eFolder Service Accepted Document Types Updates to Appendix F (Table 53) to add allowable content source values	Booz Allen Hamilton
11/16/2016	1.5	Added CDM field descriptions in section 5, removed existing CDM tables, added Appendix G “Allowable Content Source Values”	Booz Allen Hamilton

CONTENTS

1 Introduction

The *eFolder Service Contract Document (SCD)* presents the information required to consume a service, to clarify whether it is appropriate for the service consumer's needs. eFolder Service v1.0 replaces Content Management Service (CMS) and eDocument Service, which have been deprecated for future integrations.

This document provides an information model, a behavior model, and descriptions of service operations, to define eFolder services 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 execution context, as it may be implemented and deployed in different environments.

1.1 Service Information

The following table provides the eFolder Service Version 1.0 administrative information.

Table 1: Service Administrative Information

Item	Details
Service Name	eFolder
Service Version	1.0
Business Domain	TBD
Business Owner	VBA
Development Owner	OIT, VA PMO
Support Owner	OIT, VA PMO
Technical Owner	OIT, VA PMO

1.2 Service Governance

The overall service governance for this and other VBMS hosted services, including the change management process, is performed by means of the VBMS requirements process and is subject to the project's CCB and dedicated scope management process.

1.3 References

- Universally Unique Identifier (UUID), http://www.ietf.org/rfc/rfc4122.txt
- IEPD Regex = \{[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}\}/I or
- Java Regex = [0-9a-f]{8}-[0-9a-f]{4}-4[0-9a-f]{3}-[ab89][0-9a-f]{3}-[0-9a-f]{12}/i

2 eFolder Service Overview

The eFolder Service encompasses several SOAP web services, allowing clients to manage documentation used in VBMS, including the ability to create, read, update and send documents. Included in the overarching eFolder Service are:

- **Association Service:** Allows claims and to be added (associated) to a document series or document version. Also allows VBMS to accept notifications of packages pending upload, and packages that have completed upload.
- **Package Service:** Aggregates document versions into named packages, and supports sending named packages to a specific recipient at a specified destination.
- **Read Service:** As the general read interface for the eFolder, this service provides operations that look up and fetch document data, including document version, find associations and call search operations. An identical read service is deployed at a slightly different endpoint for clients that need MTOM turned off (typically .NET clients).
- **Upload Service:** Provides operations for uploading new documents, updating existing documents, and renaming documents.

2.1 Dependencies

eFolder Service depends on or interacts with the following other services, systems, databases, etc., to perform its function:

- **Benefits Enterprise Platform (BEP):** The eFolder Service uses BEP for security purposes. At a high level, a calling client application should integrate with a trusted Identity Provider (IdP) within the BEP infrastructure.
- **Benefits Gateway Services (BGS):** The eFolder Service collaborates with BGS for certain data persistence operations.
- **FileNet:** The eFolder Service uses FileNet to retrieve and store documents of record.
- **VBMS Oracle Database:** The eFolder Service uses the VBMS Oracle database for any document metadata information required.

2.2 Technical Specification

The following specification allows developers to discover the service for run-time consumption:

Table 2: Technical Specification	
Item	**Details**
Service Invocation Type	SOAP over hypertext transfer protocol(HTTP)
Service Interface Type	WSDL via Web Service 2.0
Service Name	eFolder Service-v1
Interface	Association Service: http://<service_domain>/vbms-efolder-svc/association-v1/eFolderAssociationService?wsdl Package Service: http://<service_domain>/vbms-efolder-svc/package- v1/eFolderPackageService?wsdl Read Service: http://<service_domain>/vbms-efolder-svc/read-v1/eFolderReadService?wsdl No MTOM Read Service (.NET clients in particular might want to use this instead): http://<service_domain>/vbms-efolder-svc/read-v1/eFolderReadServiceInline?wsdl Upload Service: http://<service_domain>/vbms-efolder-svc/upload-v1/eFolderUploadService?wsdl
Schemas	Linked in the WSDL
End Points	Association Service: http://<service_domain>/vbms-efolder-svc/association-v1/eFolderAssociationService Package Service: http://<service_domain>/vbms-efolder-svc/association- v1/eFolderPackageService Read Service: http://<service_domain>/vbms-efolder-svc/read-v1/eFolderReadService No MTOM Read Service: http://<service_domain>/vbms-efolder-svc/read-v1/eFolderReadServiceInline Upload Service: http://<service_domain>/vbms-efolder-svc/upload-v1/eFolderUploadService

2.3 Mapping to Legacy Services (eDocument v4.0)

The following table provides information about replacements for legacy eDocument Service calls:

Table 3: Legacy Services Mapping

eDocument Service v4.0	eFolder Service v1.0	Notes
uploadDocument	Upload/uploadDocument	Call Upload/initializeUpload, then Upload/uploadDocument
uploadLetterDocument	—	Deprecated
fetchDocument	Read/findDocumentVersionReference Read/getDocumentContent	Call Read/find DocumentVersionReference to search for document attributes Call Read/getDocumentContent for binary content
getDocumentTypes	Read/listTypeCategories	
uploadFormDocument	Upload/uploadDocument	Call Upload/initializeUpload, then Upload/uploadDocument
getFormFieldsForDocumentType	Read/findMetadataKeysForTypeCategory	–
uploadDocumentWithAssociations	Upload/uploadDocument Associate/associateClaim or Associate/associateContention	Call Upload/initializeUpload then Upload/uploadDocument to upload Then call Associate/associateClaim or Associate/associateContention to create an association
fetchDocumentById	Read/getDocumentVersionReference Read/getDocumentContent	Call Read/getDocumentVersionReference for document attributes Call Read/getDocumentContent for the binary content
listDocuments	Read/findDocumentSeriesReference or Read/findDocumentVersionReference	Use Read/findDocumentVersionReference for current versions in eFolder Use Read/findDocumentSeriesReference for current and past versions
listDocumentsByDocumentTypeId	Read/findDocumentVersionReference	Specify the type id in the criteria parameter to findDocumentVersionReference

3 Information Model

A canonical model is a design pattern used to communicate between different data formats. A form of enterprise application integration, it is intended to reduce costs and standardize on agreed data definitions associated with integrating business systems.

The Canonical Data Model (CDM) helps to minimize dependencies when integrating applications and services that use different data formats. The CDM is independent from any specific application, requiring each application to produce and consume messages in a common format. If a new application is added to the integration solution, only the transformation between the CDM to the application's data model should be created, independent from the number of applications that already participate.

The following sections define the primary CDM data objects provided by eFolder Services as part of message payloads and service contracts. Only those aspects of data objects that are common across all service operations are defined. These objects do not define a database schema or class model, but define the objects that are passed between the service and the service consumers.

3.1 Document Series and Versions

Each DocumentSeriesReference is backed by versions of the document called DocumentVersionReference. A single DocumentSeriesReference may have multiple versions.

The first version is guaranteed to represent the original content sent from the client when invoking initializeUpload and uploadDocument. The service may then create subsequent DocumentVersionReferences, depending on the document content sent to the service, to optimize the document for viewing in the VBMS eFolder UI.

If a subsequent version is created, all metadata associated with the first version is copied to the subsequent version. When a single DocumentVersionReference is returned, it is considered metadata disconnected from its DocumentContent, which represents the actual binary content stored in the system.

Client applications that want to list the contents of a Veteran's eFolder would start with DocumentSeriesReference and get the latest DocumentVersionReference. If the client system then wishes to materialize the document content, it would call ReadService.getDocumentContent, passing the ID of the DocumentVersionReference.

The externalId element of DocumentSeriesReference has been deprecated and should not be used.

3.1.1 DocumentContent

DocumentContent is the class that contains the byte stream of the data for the document itself.

Table 4: DocumentContent Class Attributes			
Name	**Type**	**Constraints**	**Description**
bytes	base64Binary	—	The binary content of the document.
documentVersionReferenceId	GUID	Must be the GUID of an existing DocumentVersionReference.	The Id of the DocumentVersionReference the content represents.

3.1.2 DocumentPackage

Represents a package of scanned documents processed by a VA centralized mail-scanning vendor.

Table 5: DocumentPackage Class Attributes			
Name	**Type**	**Constraints**	**Description**
docCountByTypeCategory	TypeCategoryItem	—	Information about how many documents of a TypeCategory are in the package.
docCountTotal	—	Must equal the sum of all docCountByTypeCategory.count values.	The total number of documents in the package.
packageId	string	—	The unique identifier of the package.
veteran	VeteranIdentifier	—	The Veteran whose documents are in the package.

3.1.3 DocumentSeriesReference

DocumentSeriesReference is the class for describing a document series within VBMS. A document series contains a list of document versions. Only one document version in a series is considered the active one, meaning that it is the version visible within the eFolder.

Table 6: DocumentSeriesReference Class Attributes			
Name	**Type**	**Constraints**	**Description**
id	GUID	—	The UUID of the DocumentSeriesReference.
metadata	MapEntry	—	Key/value pairs describing the series. NOTE: In 13.0, this field is not supported. Attempts to use this field will be ignored.
seriesName	string	Avoid upper-case letters and special characters (except hypens and underscores).	A descriptive name of the document.
versions	DocumentVersion Reference	—	A list of all DocumentVersionReferences associated with this Document Series.

Table 6: DocumentSeriesReference Class Attributes			
Name	**Type**	**Constraints**	**Description**
veteranReference	VeteranReference	—	The Veteran the document belongs to.

3.1.4 DocumentVersionReference

DocumentVersionReference is the class for an individual version of a document. It contains all the attribute fields used to describe the document version, a UUID for the document series it belongs to, and a list of key/value pairs used to represent business form fields.

Table 7: DocumentVersionReference Class Attributes			
Name	**Type**	**Constraints**	**Description**
documentSeriesRefId	GUID	—	The ID of the DocumentSeries that the DocumentVersionReference belongs to.
documentVersionRefId	GUID	—	The ID of this DocumentVersionReference
externalId	string	—	An ID assigned by the VBMS-R system for this document. This value should be ignored by all other than VBMS-R.
metadata	MapEntry	The "Key" must be one of the values listed in table "Supported Metadata Fields."	Key/value pairs describing document data, intended to be used for describing forms filled out in the document.
mimeType	string	—	The MIME type of the document.
previousVersionId	GUID	—	The ID of the DocumentVersionReference that this DocumentVersionReference superceded. If null, this DocumentVersion is the original version.
source	source	—	The source of the content of the document. For legacy DocumentVersionReference objects (uploaded prior to Release 13.0) where no source was provided upon upload, this field will have value "Unknown".
subject	string	—	A value to describe the content of the document.
typeCategory	TypeCategory	—	Information about the type of form the document represents.
uploadedBy	Participant	—	Information about the VA user that uploaded the document into VBMS.
vaReceiveDate	date	—	The date of which VA came into possession of the document.
vbmsUploadDate	date	—	The date of which the document was uploaded into VBMS.
version	DocumentVersion Value	—	The number of the version within the DocumentSeriesReference.
versionName	string	—	This field is not supported in 13.0. All uses of this field will be ignored.

3.1.4.1 DocumentVersionReference Metadata Keys

A DocumentVersionReference object may have zero or more metadata elements. Each metadata will have a key, as shown in the following table:

Table 8: DocumentVersionReference Supported Metadata Keys

Key	Description	Document Type Intended
altDocType	JSON String List of Cat - Type Strings representing altDocTypes	—
benefitTypeCompensation	"1" or "0", indicates whether the type of benefit being sought is Compensation ("1") or some other type ("0"), as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
benefitTypePension	"1" or "0", indicates whether the type of benefit being sought is Pension ("1") or some other type ("0"), as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
benefitTypeSurvivorsPensionOrDIC	"1" or "0", indicates whether the type of benefit being sought is Survivors ("1") or some other type ("0"), as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
claimPreviouslyFiledYN	"Y" or "N", indicates whether the Veteran has previously submitted a claim, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
claimantFirstName	The first name of the claimant, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
claimantLastName	The last name of the claimant, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
claimantMiddleName	The middle name of the claimant, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
claimantSSN	The SSN of the claimant, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
DCS	The ID of the Document Control Sheet to which this DocumentVersionReference was associated to upon upload.	—
dcsScanningComplete	Indicates whether this document closed a DCS.	—
documentRelocatorEmail	The e-mail of a VBMS user who moved the document.	—
documentRelocatorId	The user ID of a VBMS user who moved the document.	—
emailAddressPreferred	The preferred e-mail address of the claimant, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
EP	The EP Code the DocumentVersionReference is associated with, as appears in the VBMS Core eFolder UI.	—

Table 8: DocumentVersionReference Supported Metadata Keys		
Key	**Description**	**Document Type Intended**
isNewMail	"true" or "false", whether the document is considered new mail.	—
mailingAddressCurrent	The current mailing address of the claimant, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
MimeType	The MIME type of the DocumentVersionReference, as determined by the VBMS Core server during document upload.	—
poaCode	The POA organization the uploading user belongs to, as specified by an external system during upload.	—
poaOrganization	The POA code of the uploading user, as specified by an external system during upload.	—
sourceComment	The Source Comment of the DocumentVersionReference, as appears in the VBMS Core eFolder UI.	—
spouseIsVeteran	"true" or "false", whether the Veteran's spouse is a Veteran, as scanned from the document.	"VA 21-526 Veterans Application for Compensation or Pension", document type ID 131. "VA 21-526c Pre-Discharge Compensation Claim", document type ID 530. "VA 21-526b, Veteran Supplemental Claim", document type ID 532. "VA 21-526EZ, Fully Developed Claim (Compensation)", document type ID 533.
spouseFileNumber	The file number of the Veteran's spouse, as scanned from the document.	"VA 21-526 Veterans Application for Compensation or Pension", document type ID 131. "VA 21-526c Pre-Discharge Compensation Claim", document type ID 530. "VA 21-526b, Veteran Supplemental Claim", document type ID 532. "VA 21-526EZ, Fully Developed Claim (Compensation)", document type ID 533.
spouseRelationshipStartDate	The date at which the Veteran's marriage began, as scanned from the document.	"VA 21-526 Veterans Application for Compensation or Pension", document type ID 131. "VA 21-526c Pre-Discharge Compensation Claim", document type ID 530. "VA 21-526b, Veteran Supplemental Claim", document type ID 532. "VA 21-526EZ, Fully Developed Claim (Compensation)", document type ID 533.

Table 8: DocumentVersionReference Supported Metadata Keys		
Key	**Description**	**Document Type Intended**
spouseRelationshipEndDate	The date at which the Veteran's marriage ended, as scanned from the document.	"VA 21-526 Veterans Application for Compensation or Pension", document type ID 131. "VA 21-526c Pre-Discharge Compensation Claim", document type ID 530. "VA 21-526b, Veteran Supplemental Claim", document type ID 532. "VA 21-526EZ, Fully Developed Claim (Compensation)", document type ID 533.
subject	The Subject of the DocumentVersionReference, as appears in the VBMS Core eFolder UI.	—
systemSource	The system which uploaded the document.	—
telephoneNumberPreferred	The preferred telephone number of the claimant, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
userEmail	The e-mail of the user who uploaded the document.	—
userFacilityJob	The job title of the user who uploaded the document.	—
userFirstName	The first name of the user who uploaded the document.	—
userLastName	The last name of the user who uploaded the document.	—
userMiddleName	The middle name of the user who uploaded the document.	—
userName	The BEP user ID of the user who uploaded the document.	—
userRole	The role of the user that uploaded the DocumentVersionReference. If uploaded by a VBMS user, this value will be the roles associated with that user. If uploaded by an external system, this will be the external system.	—
userSARL	The Sensitive Record Access Level of the user who uploaded the document.	—
veteranDateOfBirth	The date of birth of the Veteran, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
veteranGenderMF	'M' or 'F', whether the Veteran is male or female, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
veteranRepresentative	The POA for the Veteran, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.
veteranSSN	The SSN of the Veteran, as scanned from the document.	"VA 21-0966 Intent to File", document type ID 718.

3.1.5 DocumentVersionValue

Version tracking for our Document Version Reference.

Table 9: DocumentVersionValue Class Attributes			
Name	**Type**	**Constraints**	**Description**
major	string	Must be a positive integer.	The major version of the DocumentVersionReference.
minor	string	—	The minor version of the DocumentVersionReference.

3.1.6 MapEntry

MapEntry is a key/value pair representing document series or version metadata.

Table 10: MapEntry Class Attributes			
Name	**Type**	**Constraints**	**Description**
key	string	—	The name of a metadata field.
value	string	—	The value of a metadata field.

3.1.7 MovedItem

MovedItem represents a document move event. A move event occurs when:

- a document is moved out of a Veteran A's folder and into Veteran B's folder
- a document is moved out of Veteran A's folder and into the Unassociated folder

A MovedItem element contains the date and time of the move event, the from and to VBMS efolder identifiers, the Document Version Reference Identifier, and the Document Series Reference Identifier.

Table 11: MovedItem Class Attributes			
Name	**Type**	**Constraints**	**Description**
moveDate	Datetime	—	N/A
fromEfolder	FileNumber	An eight or nine digit number.	The file number of the Veteran.
toEfolder	FileNumber	An eight or nine-digit number.	The file number of the Veteran.
movedDocVersion RefId	GUID	—	The ID of the DocumentVersionReference that moved
movedDocSeries RefId	GUID	—	The ID of the DocumentSeriesReference that the DocumentVersionReference belongs to.

3.1.8 VersionAssociation

VersionAssociation is the class for an association to a specific document version. It is a sub-class of DocumentSeriesAssociation and contains a link to the DocumentVersionReference that has this association. DocumentSeriesAssociation is the base class for all associations to a document series. It contains the UUID of the document series that has this association.

Table 12: VersionAssociation Class Attributes			
Name	**Type**	**Constraints**	**Description**
versionReference	DocumentVersionReference	—	The ID of the DocumentVersionReference in the association.
documentSeriesRef	GUID	—	The ID of the DocumentSeriesReference that the DocumentVersionReference belongs to.

3.1.9 PagedDocumentSeriesReferences

PagedDocumentSeriesReferences is a container for returning DocumentSeriesReference types from a query that takes PagingCriteria.

Table 13: PagedDocumentSeriesReferences Class Attributes			
Name	**Type**	**Constraints**	**Description**
documentSeriesReferences	DocumentSeriesReference	—	Collection of DocumentSeriesReference types that represent the page of results.
pagingReference	PagingReference	—	Information describing the results returned in the page.

3.2 Claim Associations

A claim may be associated with one or more document versions, and each document version may be associated with one or more claims. You cannot navigate directly between a DocumentVersionReference and a ClaimReference. Instead, use the Read interface findClaimReferencesForDocumentVersionReference operation to find ClaimReferences for a given DocumentVersionReference.

3.2.1 ClaimAssociation

ClaimAssociation is the class for an association of a document version to a claim. It is a sub-class of VersionAssociation, containing the reference to the document version that has this claim association.

Table 14: ClaimAssociation Class Attributes			
Name	**Type**	**Constraints**	**Description**
claimReference	ClaimReference	—	A reference to the claim associated with a DocumentVersionReference.

3.2.2 ClaimReference

A lightweight model containing only immutable claim data.

Table 15: ClaimReference Class Attributes			
Name	**Type**	**Constraints**	**Description**
claimId	long	Must match the ID of a claim associated with the specified file number.	The ID of the claim.

Table 15: ClaimReference Class Attributes			
Name	**Type**	**Constraints**	**Description**
establishDate	date	—	The date when the claim was established.
fileNumber	FileNumber	—	The file number of the Veteran that filed the claim.

3.3 Document Associations

Documents can be associated with other objects in the data model.

3.3.1 FileNumber

A file number is used as a unique identifier for a Veteran.

Table 16: FileNumber Tagged Values		
Name	**Type**	**Value**
xs:restriction/xs:maxLength@value	Text	9
xs:restriction/xs:minLength@value	Text	8
xs:restriction/xs:pattern@value	Text	[0-9]*
xs:restriction@base	Text	xs:string

3.3.2 FormField

Represents an individually accessible data field on a document.

Table 17: FormField Class Attributes			
Name	**Type**	**Constraints**	**Description**
maxLength	long	Must be a positive integer.	The maximum length that can be fit in field "value".
name	string	—	A name describing the form field.
required	boolean	—	True if the field is required for a particular document, false otherwise.
value	value	—	The value of the document's form field.

3.3.3 GUID

GUID is a globally unique identifier.

Table 18: GUID Tagged Values		
Name	**Type**	**Value**
xs:restriction/xs:pattern@value	Text	\{[a-zA-Z0-9\-_]*\}
xs:restriction@base	Text	xs:string

3.3.4 TypeCategory

TypeCategory is the class for a document category and type value taken from the Master Document Type Categories spreadsheet contained in ACR 156.

Table 19: TypeCategory Class Attributes			
Name	**Type**	**Constraints**	**Description**
categoryDescriptionText	string	—	A description of a document category.
categoryId	long	—	The unique ID of the document category.
subCategoryText	string	—	A sub-description of a document that distinguishes multiple categories with the same "categoryDescriptionText".
typeDescriptionText	string	—	A description of the document type.
typeId	string	—	The unique ID of the document type.
typeLabelText	string	Must follow the regular expression L[0-9]{3}, e.g. L001.	A legacy value uniquely identifying the document type, separate from typeId.

3.3.5 VeteranIdentifier

VeteranIdentifier is the type for containing all unique identifiers for a Veteran. These unique identifiers are values that are stored within MVI.

Table 20: VeteranIdentifier Class Attributes			
Name	**Type**	**Constraints**	**Description**
edipi	string	—	The EDIPI of the Veteran. NOTE: In 13.0, this field is not supported. All attempts to use this field will be ignored.
fileNumber	FileNumber	An eight or nine-digit number.	The file number of the Veteran.
ssn	ssn	A nine-digit number.	The SSN of the Veteran.

3.3.6 VeteranReference

VeteranReference is a sub-class of Participant that represents a Veteran. It contains an identifier for the Veteran. Participant is the abstract base type for all person or organization types in the VBMS CDM whose source data originates from non-VBMS systems. Typically, the types would be Veteran or Organization that originate in the BGS system.

Table 21: VeteranReference Class Attributes			
Name	**Type**	**Constraints**	**Description**
id	VeteranIdentifier	—	The VeteranIdentifier for the Veteran.
veteranSensitivityLevel	veteranSensitivityLevel	A number between 0 and 9, if provided.	The sensitivity level of the Veteran.
externalParticipantId	string	—	The VA participant ID of the Veteran.

3.3.7 VeteranSensitivityLevel

VA records are assigned a sensitivity level from 1-9.

Table 22: VeteranSensitivityLevel Tagged Values		
Name	**Type**	**Value**
xs:simpleType/xs:restriction/xs:maxInclusive@value	Text	9

Table 22: VeteranSensitivityLevel Tagged Values		
Name	**Type**	**Value**
xs:simpleType/xs:restriction/xs:minInclusive@value	Text	1
xs:simpleType/xs:restriction@base	Text	xs:long

3.4 Search Criteria

To use the service's search capabilities, populate a search criteria object with fields you are interested in and pass it to the appropriate operation. The operation will return a list of objects that match the search criteria.

See each service operation description in Read Service Interface for further details about the SearchCriteria metadata keys supported by each operation.

3.4.1 DateRange

DateRange is a type that represents a range of dates. It is useful for searching elements that appear between two dates, before one date, or after one date.

Table 23: DateRange Class Attributes			
Name	**Type**	**Constraints**	**Description**
begin	date	Must be before "end" if both are specified.	The beginning of the date range, inclusive. If not supplied, the date range beginning is unbounded.
end	date	Must be after "begin" if both are specified.	The end of the date range, inclusive. If not supplied, the date range end is unbounded.

3.4.2 DateTimeRange

DateTimeRange is a type that represents a range of datetimes. It is useful for more precise searching of elements that appear between two dates, before one date, or after one date.

Table 24: DateTimeRange Class Attributes			
Name	**Type**	**Constraints**	**Description**
from	dateTime	Must be before "to" if both are specified.	The beginning of the dateTime range, inclusive
to	dateTime	Must be after "from" if both are specified.	The end of the dateTime range, exclusive

3.4.3 DocumentSeriesReferenceSearchCriteria

DocumentSeriesReferenceSearchCriteria is a sub-class of DocumentSearchCriteria used in search operations for a DocumentSeriesReference.

DocumentSearchCriteria is a sub-class of SearchCriteria used for operations that perform a search for Documents. This class has a required field veteran of type VeteranIdentifier. All searches for documents within VBMS must be scoped to an individual Veteran.

SearchCriteria is the base class for all operations to search for documents within VBMS. This class has optional field metadata that can be used for searches on metadata keys.

Table 25: DocumentSeriesReferenceSearchCriteria Class Attributes			
Name	**Type**	**Constraints**	**Description**
seriesName	string	—	Only the Document Series with seriesName matching this value will be returned. No results are returned if there is no match.

3.4.4 DocumentVersionReferenceSearchCriteria

DocumentVersionReferenceSearchCriteria is a sub-class of DocumentSearchCriteria used for operations that perform a search for DocumentVersionReferences.

DocumentSearchCriteria is a sub-class of SearchCriteria used for operations that perform a search for Documents. This class has a required field veteran of type VeteranIdentifier. All searches for documents within VBMS must be scoped to an individual Veteran.

SearchCriteria is the base class for all operations to search for documents within VBMS. This class has optional field metadata that can be used for searches on metadata keys.

Table 26: DocumentVersionReferenceSearchCriteria Class Attributes			
Name	**Type**	**Constraints**	**Description**
categoryId	long	Must be a value of a Category ID as listed the Master Document Type Categories, found in ACR156.	The ID of a document category to search for.
externalId	string	—	An externalId to search for. Intended to be used by VBMS-R.
parentDocumentSeriesRefId	GUID	—	The ID of the DocumentSeries to limit DocumentVersionReference results to.
typeId	long	—	The ID of a document type to search for.
uploadedBy	Participant	—	This field is not supported in 13.0. All uses of this field will be ignored.
vaReceiveDate	DateRange	—	A date range defining the period of time interested in for when the VA received the document.
vbmsUploadDate	DateRange	—	A date range defining the period of time interested in for when the document was uploaded to VBMS.
metadata	MapEntry	The "Key" must be one of the values listed in table "Supported Metadata Fields."	Key/value pairs that the DocumentVersionReference must have to be included in the result.
veteran	VeteranIdentifier	EDIPI is not supported in Release 13.0.	The Veteran whose documents should be returned, if all other criteria are matched.

3.4.5 TypeCategoryItem

TypeCategoryItem allows you to search documents via type and/or category.

Table 27: TypeCategoryItem Class Attributes			
Name	**Type**	**Constraints**	**Description**
count	—	—	The number of documents that match this document type.
typeCategory	TypeCategory	—	The type and category of the documents.

3.4.6 TypeCategorySearchCriteria

TypeCategorySearchCriteria is a sub-class of SearchCriteria used for operations that perform a search using TypeCategory. SearchCriteria is the base class for all operations to search for documents within VBMS.

Table 28: TypeCategorySearchCriteria Class Attributes			
Name	**Type**	**Constraints**	**Description**
dateTimeRange	DateTimeRange	—	The dateTime range defining the period of time of VBMS uploads to constrain the search to.
typeCategory	TypeCategory	—	The type and category information of documents to constrain the search to.
metadata	MapEntry	—	This field is not supported in 13.0. All attempts to use this field will be ignored.

3.4.7 PagingCriteria

PagingCriteria is used to limit the number of rows that are returned from a search, as well as to continue search results starting from a given result location.

Table 29: PagingCriteria Class Attributes			
Name	**Type**	**Constraints**	**Description**
pageSize	Integer	—	The maximum number of rows to return in the response. Defaults to 5000.
startIndex	Integer	—	The row number to start the return results as. Defaults to 0, which is the first row.

3.4.8 PagingReference

PagingReference provides information for results returned by a service that utilizes PagingCriteria.

Table 30: PagingReference Class Attributes			
Name	**Type**	**Constraints**	**Description**
filteredResults	Integer	—	The number of results that are hidden on this page due to security discriminators or other applied filtering.
nextStartIndex	Integer	—	The row number to use to continue the search for the next page of results. If the current set of results is the last page of results, a -1 will be returned.
totalResultCount	Integer		The total number of rows available for the search across all pages.

4 Association Service Interface

The Association Service provides an API by which claims, contentions, annotations, links, and bookmarks can be added (associated) to a document series or document version. Currently, the Association Service only supports associating claims and contentions to documents.

Additionally, the Association Service provides an API to notify VBMS of document packages pending upload, and document packages that have completed upload. The term *package* in the context of the Association Service should not be equated to its usage within the Package Service.

4.1 Association Service Technical Specification

The following specification allows developers to discover the service for run-time consumption:

Table 31: Association Service Technical Specification	
Item	**Details**
Service Invocation Type	SOAP over Hypertext Transfer Protocol(HTTP)
Service Interface Type	Web Services Description Language (WSDL) via Web Service 2.0
Service Name	eFolderAssociationService
Interface	http://<service_domain>/vbms-efolder-svc/association-v1/eFolderAssociationService?wsdl
Schema	Linked in the WSDL
End Points	http://<service_domain>/vbms-efolder-svc/association- v1/eFolderAssociationService

4.2 Association Service Interface Operations

Methods in this interface allow associating documents to other related entities.

4.2.1 associateClaim

Associates a document to one or more claims. A document version may be associated with many claims.

- associateClaim (documentVersionReference : UUID, claimReference : ClaimReference) : void

Table 32: associateClaim				
	Name	**Type**	**Required?**	**Notes**
In:	documentVersionReferenceId	string	Yes	The UUID of the DocumentVersionReference to associate claims to. If the DocumentVersionReference is not the current version of its associate DocumentSeriesReference, the claim association will be created on the current version as well as the version supplied in this field.
In:	claimIdList	string	Yes	Cardinality: 1 - 5000. The IDs of the claims to associate to this document. All claim IDs should be numeric.
Out:	—			

Table 33: associateClaim Faults		
Name	**Message**	**Notes**
versionNotFoundError	EFolderVersionNotFoundException	No DocumentVersionReference exists with the specified UUID.
claimNotFoundError	EFolderClaimNotFoundException	No claim exists with the specified ID. This error is thrown if at least one of the IDs does not correspond to claims associated with the Veteran the document belongs to.
invalidClaimError	EFolderInvalidClaimException	A value supplied in claimIdList is not a valid claim ID.
error	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

4.2.2 notifyPackageScanned

When documents arrive at an RO, the worker boxes up the Veteran's documents, inserts a DCS cover sheet and ships the documents to the scanning vendor. When the documents arrive at the scanning vendor, the worker will scan the Veteran's documents and when the final page is complete, they call the eFolder/eDocument association service method 'notifyPackageScanned', completing the process.

Call notifyPackageScanned to provide the package ID and a manifest of documents included in a package. The manifest should include a list of document types with the number of each type included in the package. When scanning is complete, call notifyPackageComplete with the package ID.

When notifyPackageScanned is invoked by an external client, the "Pending Upload Indicator" will be set for the Veteran within the VBMS Work Queue and the VBMS eFolder. This icon can be cleared manually by a VBMS NWQ Administrator or through a subsequent call to notifyPackageComplete.

- notifyPackageScanned (documentPackage:DocumentPackage) : String

Table 34: notifyPackageScanned				
	Name	**Type**	**Required?**	**Notes**
In:	documentPackage	DocumentPackage	packageId - A unique identifier for the package. docCountTotal - An integer equalling the total number of documents in the package docCountByTypeCategory - Cardinality 0-5000. A TypeCategory of the document inside the package, with the count of documents of that TypeCategory. Veteran - A VeteranIdentifier the package is associated to	Represents a package of scanned documents processed by a VA centralized mail scanning vendor.
Out:	"success"			

Table 35: notifyPackageScanned Faults		
Name	**Message**	**Notes**
veteranNotFoundFault	EFolderVeteranNotFoundException	No Veteran exists with the specified Veteran identifier.
packageAlreadyExists	EFolderPackageAlreadyExists	A package with the ID already exists in VBMS.
sumOfTypeCategoryCountsDoesNotMatchTotalPackageCount	EFolderSumOfTypeCategoryCountsDoesNotMactchTotalPackageCount	The value of docCountTotal does not match the sum of all TypeCategoryItem.count values.
error	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

4.2.3 notifyPackageComplete

When scanning is complete and all documents in a package have been uploaded, call notifyPackageComplete with the package ID. When notifyPackageComplete is invoked by an external client, the "Pending Upload Indicator" will be cleared for this package. If no other packages are pending upload, then the "Pending Upload Indicator" will no longer appear in the VBMS eFolder or Work Queue for this Veteran.

- notifyPackageComplete (packageId : UUID)

Table 36: notifyPackageComplete				
	Name	**Type**	**Required?**	**Notes**
In:	packageId	string	Yes	The ID of the package to close.
Out:	"success"			

Table 37: notifyPackageComplete Faults		
Name	**Message**	**Notes**
invalidPackageError	EFolderPackageNotFoundException	No package was found with the supplied packageId.
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

4.2.4 associateContention

Associates a document to one or more contentions. A document version may be associated with many contentions.

- associateContention (documentVersionReference : UUID, contentionIdList : List<String>) : void

Table 38: associateContention				
	Name	**Type**	**Required ?**	**Notes**
In:	documentVersion ReferenceId	string	Yes	The UUID of the DocumentVersionReference to which contention associations will be created. If the DocumentVersionReference is not the current version of its associated DocumentSeriesReference, the contention association will be created on the current version as well as the version supplied in this field.
In:	contentionIdList	string	Yes	Cardinality: 1 - 5000. The IDs of the contentions to associate to this document. All contention IDs should be numeric.
Out:	—			

Table 39: associateContention Faults		
Name	**Message**	**Notes**
versionNotFoundError	EFolderVersionNotFound Exception	No DocumentVersionReference exists with the specified UUID.
contentionNotFoundError	EFolderContentionNotFoundException	A value in contentionIdList does not correspond to a contention.
invalidContentionError	EFolderInvalidContention Exception	A value supplied in contentionIdList is not a valid contention ID.
error	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

5 Package Service Interface

The Package Service provides an API by which collections of documents can be packaged and sent to recipients through centralized mailing services. The term *package* in the context of the Package Service should not be equated to its usage within the Association Service.

The Package Service aggregates document versions into named packages, and supports sending named packages to a specific recipient (person, organization, or system) at a specified mailing address destination.

Future enhancements will add Power of Attorney recipients and email addresses as destinations, as well as adding other destinations that are automatically derived based on available metadata.

5.1 Package Service Technical Specification

The following specification allows developers to discover the service for run-time consumption:

Table 40: Association Service Technical Specification	
Item	**Details**
Service Invocation Type	SOAP over Hypertext Transfer Protocol(HTTP)
Service Interface Type	Web Services Description Language (WSDL) via Web Service 2.0
Service Name	eFolderPackageService
Interface	http://<service_domain>/vbms-efolder-svc/package- v1/eFolderPackageService?wsdl
Schema	Linked in the WSDL
End Points	http://<service_domain>/vbms-efolder-svc/association- v1/eFolderPackageService

5.2 Package Service Interface Operations

Methods in this interface allow adding documents to named packages and sending documents to specified recipients through the centralized mailing service.

5.2.1 createPackage

Creates a named communication package.

- createPackage (fileNumber: String, packageName: String, documentVersionReferences[] : DocID) : UUID

Table 41: createPackage				
	Name	**Type**	**Required?**	**Notes**
In:	fileNumber	string	Yes	The fileNumber of the Veteran who the documentVersionReferences are associated to.

Table 41: createPackage				
	Name	**Type**	**Required?**	**Notes**
In:	packageName	string	Yes	The name to give the package. This is required to differentiate the package to users of the packaging functionality.
In:	documentVersionReferences	string	Yes	The eFolder DocIDs of the document versions to be included in the package. Cardinality: 1 - 5000
Out:	UUID	UUID	YES	The UUID of the package that has been created. This UUID is used when sending the package in a distribution.

5.2.2 distributePackage

Once a package is created, it can then be sent to recipients through the centralized mailing services.

- distributePackage (packageUuid: UUID, recipient: RecipientType, destinations[]: DestinationType) : DistributionType

Table 42: distributePackage				
	Name	**Type**	**Required?**	**Notes**
In:	packageUuid	UUID	yes	The UUID of a package to send. This is the value returned from the createPackage method.
In:	recipient	RecipientType	yes	The recipient receiving the package.
In:	Destinations[]	DestinationType	yes	The location to send the document package. Currently, only mailing addresses (MailingDestinationTypes) are supported. Future enhancements will support electronic delivery.
Out:	DistributionType Provides the distribution that was sent to the CBCM mailing services. Contains all relevant UUIDs and timestamps. If a distribution is returned, that means the CBCM has successfully received the request, and that the request is queued for processing (printing and mailing).			

6 Read Service Interface

The Read interface provides operations that look up and fetch document data. Use this service to get binary content for a Document Series, add annotations and bookmarks, find associations, and call search operations. Version 1.0 does not yet support all of the intended functionality.

The Read Service has three basic behavior patterns: getting, finding, and listing. Which one you use depends on the parameter you pass. In general, if you have a UUID for something, and all you want is that thing, then you will use a getter. If you want to find all instances of a thing that match a certain set of parameters, you pass a Criteria object into a find operation. List operations do not require a parameter and simply return a list.

6.1 Read Service Technical Specification

The following specification allows developers to discover the service for run-time consumption:

Table 43: Read Service Technical Specification

Item	Details
Service Invocation Type	SOAP over Hypertext Transfer Protocol(HTTP)
Service Interface Type	Web Services Description Language (WSDL) via Web Service 2.0
Service Name	eFolderReadService
Interface	http://<service_domain>/vbms-efolder-svc/read-v1/eFolderReadService?wsdl If you need a version with no MTOM please use: http://<service_domain>/vbms-efolder-svc/read-v1/eFolderReadServiceInline?wsdl
Schema	Linked in the WSDL
End Points	http://<service_domain>/vbms-efolder-svc/read-v1/eFolderReadService If you need a version with no MTOM please use: http://<service_domain>/vbms-efolder-svc/read-v1/eFolderReadServiceInline

6.2 Read Service Interface Operations

All operations follow the convention of wrapping input and output parameters in an envelope named according to the operation. For example, the input parameter for the findDocumentSeriesReference operation is named "findDocumentSeriesReference" and the output parameter is named "findDocumentSeriesReferenceResponse." For clarity's sake, these wrapper names are omitted below.

6.2.1 findClaimReferencesForDocumentVersionReference

Returns a list of claim references associated with a particular document version.

- findClaimReferencesForDocumentVersionReference (documentVersionRefId : UUID) : List<ClaimReference>

Table 44: findClaimReferencesForDocumentVersionReference				
	Name	**Type**	**Required?**	**Notes**
In:	documentVersionRefId	UUID	N/A	Document Version Reference Id (could be provided by results of a search method, for example)
Out:	List of ClaimReference			

Table 45: findClaimReferencesForDocumentVersionReference Faults		
Name	**Message**	**Notes**
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

6.2.2 findDocumentSeriesReference

This call returns a list of DocumentSeries objects containing the metadata for documents that match search criteria. Search criteria will be verified and rules may be applied to ensure the search operation has the correct combination of fields as per our governance TBD.

- findDocumentSeriesReference (criteria : DocumentSeriesReferenceSearchCriteria) : List<DocumentSeriesReference

Table 46: findDocumentSeriesReference				
	Name	**Type**	**Required?**	**Notes**
In:	DocumentSeriesReference SearchCriteria	DocumentSeriesReference SearchCriteria	Veteran is a required criterion	Criteria include veteran, seriesName, and supported metadata elements. seriesName expects a concatenation of Veteran's file number and name of file
Out: List of DocumentSeriesReference				

Table 47: findDocumentSeriesReference Faults		
Name	**Message**	**Notes**
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

6.2.3 findDocumentVersionReference

This call returns a list of document version references matching the search criteria.

- findDocumentVersionReference (criteria : DocumentVersionReferenceSearchCriteria) : List>DocumentVersionReference

Table 48: findDocumentVersionReference				
	Name	**Type**	**Required ?**	**Notes**
In:	documentVersionReference SearchCriteria	DocumentVersionReference SearchCriteria	–	Criteria keys include veteran, vaReceiveDate, parentDocumentSeriesRefId, vbmsUploadDate, category, uploadedBy, and the supported metadata elements. externalId is a legacy key to support a VBMS-R use case.
Out:	List of FormField elements. The name attribute of the FormField represents the key that can be used for metadata Map Entry elements.			

Table 49: findDocumentVersionReference Faults		
Name	**Message**	**Notes**
invalidDocTypeError	EFolderInvalidDocTypeException	The document type supplied is not recognized by VBMS
invalidDocCategoryError	EFolderInvalidDocCategoryException	The document category supplied is not recognized by VBMS.
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

6.2.3.1 findDocumentService Metadata Fields

findDocumentVersionReference will return DocumentVersionReference objects that match all metadata supplied.

For example, consider a client wanting to search for PDF documents associated to a particular claim. The client would create a DocumentVersionReferenceSearchCriteria object with a "veteran" element whose ssn attribute is set to a 9-byte numeric string. The object would have a metadata element whose key attribute is "MimeType" and whose value is "application/pdf," and another metadata element whose key is "ClaimAssociation" and whose value is the claim ID.

The following table shows possible metadata keys to use in findDocumentVersionReference searches:

Table 50: findDocumentVersionReference Supported Metadata Fields	
Key	**Description**
ContentionAssociation	ID of contention associated to the document, as created during invocations of eFolder Association Service method associateContention
ClaimAssociation	ID of claim associated to the document, as created during invocations of eFolder Association Service method associateClaim
DocumentAssociation	GUID of DocumentVersionReference associated to this document. This field is only used for associating DBQ XML to DBQ PDF files. It is intended for use by VBMS-R only.

Table 50: findDocumentVersionReference Supported Metadata Fields	
Key	**Description**
userRole	The role of the user that uploaded the DocumentVersionReference. If uploaded by a VBMS user, this value will be the roles associated with that user. If uploaded by an external system, this will be the external system.
systemSource	The system which uploaded the document.
subject	The Subject of the DocumentVersionReference, as appears in the VBMS Core eFolder UI.
sourceComment	The Source Comment of the DocumentVersionReference, as appears in the VBMS Core eFolder UI.
poaOrganization	The POA organization the uploading user belongs to, as specified by an external system during upload.
poaCode	The POA code of the uploading user, as specified by an external system during upload.
EP	The EP Code the DocumentVersionReference is associated with, as appears in the VBMS Core eFolder UI.
DCS	The ID of the Document Control Sheet to which this DocumentVersionReference was associated to upon upload.
Contention	The value of a "Contention" form field scanned and uploaded as metadata for a 526 document.
MimeType	The MIME type of the DocumentVersionReference, as determined by the VBMS Core server during document upload.

6.2.4 findMetadataKeysForTypeCategory

This call returns a list of document metadata keys for a given Type Category. These keys can be used in a subsequent call to Upload interface's initilizeUpload or initializeUpdate operation as the key attributes in the versionMetadata map entries.

- findMetadataKeysForTypeCategory (typeCategory : TypeCategorySearchCriteria) : String

Table 51: findMetadataKeysForTypeCategory				
	Name	**Type**	**Required?**	**Notes**
In:	typeCategory	TypeCategorySearchCriteria	categoryId	Only the typeId field of the typeCategory object is used; all other fields are ignored.
Out:	List of FormField elements. The name attribute of the FormField represents the key that can be used for metadata Map Entry elements.			

Table 52: findMetadataKeysForTypeCategory Faults		
Name	**Message**	**Notes**
invalidDocCategoryError	EFolderInvalidDocCategoryException	The document type supplied is not recognized by VBMS
eFolderFault	EFolderServiceExcepetion	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.Informational message

6.2.5 getDocumentContent

getDocumentContent gets the binary document content for a specific Document Version.

- getDocumentContent (documentVersionRefId : UUID) : DocumentContent

Table 53: getDocumentContent				
	Name	**Type**	**Required?**	**Notes**
In:	documentVersionRefId	UUID	N/A	Id of the document version reference of interest.
Out:	Returns the binary DocumentContent for a specific Document Version.			

Table 54: getDocumentContent Faults		
Name	**Message**	**Notes**
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

6.2.6 getDocumentContentAnnotations

getDocumentContentAnnotations XML containing the annotations for a document version in Adobe's XML Forms Data Format (XFDF version 3.0). The XFDF is returned as an MTOM attachment if using the MTOM-enabled endpoint, otherwise it is returned as base64 encoded inside the SOAP response.

- getDocumentContentAnnotations (documentVersionRefId : UUID) : DocumentContent

Table 55: getDocumentContentAnnotations				
	Name	**Type**	**Required?**	**Notes**
In:	documentversionRefId	UUID	N/A	This is the UUID of the DocumentVersionReference for which you want associated annotations.
Out:	Returns the DocumentVersionReferences for a specific Claim Reference.			

Table 56: getDocumentContentAnnotations		
Name	**Message**	**Notes**
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

6.2.7 getDocumentSeriesReference

Gets a single DocumentSeriesReference object, which is the basic Document metadata object. The DocumentSeriesReference is disconnected from binary PDF content. To materialize the content for a document version within the series, call getDocumentContent() using one of the Document Version Reference IDs in the DocumentSeriesReference object.

- getDocumentSeries Reference (id : UUID) : DocumentSeriesReference

Table 57: getDocumentSeriesReference				
	Name	**Type**	**Required?**	**Notes**
In:	Id	UUID	N/A	The UUID of the DocumentSeriesReference to obtain.
Out:	DocumentSeriesReference			

Table 58: getDocumentSeriesReference Faults		
Name	**Message**	**Notes**
seriesNotFoundError	EFolderSeriesNotFoundException	No document series with a matching UUID
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

6.2.8 getDocumentVersionReference

getDocumentVersionReference gets the document version metadata object with the given ID, disconnected from its actual content.

- getDocumentVersionReference (id: UUID) : DocumentVersionReference

Table 59: getDocumentVersionReference				
	Name	**Type**	**Required?**	**Notes**
In:	Id	UUID	N/A	The UUID of the DocumentVersionReference to obtain.
Out:	DocumentVersionReference			

Table 60: getDocumentVersionReference Faults		
Name	**Message**	**Notes**
versionNotFoundError	EFolderVersionNotFoundException	No document version with a matching UUID
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

6.2.9 listTypeCategory

listTypeCategory returns list of all type categories within eFolder Service.

- listTypeCategories () : TypeCategorySearchCriteria

Table 61: listTypeCategory				
	Name	**Type**	**Required?**	**Notes**
In:	N/A	N/A	N/A	
Out:	List of TypeCategory			

Table 62: listTypeCategory Faults		
Name	**Message**	**Notes**
serviceError	EFolderServiceException	Empty list was returned by server.

Table 62: listTypeCategory Faults		
Name	**Message**	**Notes**
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

6.2.10 findVeteransWithFolderModifications

findVeteransWithFolderModifications searches for Veterans that have had at least one document uploaded matching the TypeCategory information within the supplied DateTimeRange. For safety, the service will return a maximum of 5000 results. If the corresponding error message is received for the scenario where the maximum has been exceeded, please narrow your DateTimeRange and try again.

- findVeteransWithFolderModifications(criteria:TypeCategorySearchCriteria)

Table 63: findVeteransWithFolderModifications				
	Name	**Type**	**Required?**	**Notes**
In:	criteria	TypeCategory SearchCriteria	See next box-->	typeCategory: Search for folders that have had documents with the corresponding TypeCategory information added. Only folders with documents that match all supplied TypeCategory fields will be returned. DateTimeRange.from: Search inclusively for documents added on or after this datetime. This field must be supplied, and it must be before the value for 'DateTimeRange.to". See Appendix for DateTimeRange formatting. DateTimeRange.to: Search exclusively for documents added before this datetime. This field must be supplied, and it must be after the value for "DateTimeRange.from". See Appendix for DateTimeRange formatting.
Out:	List of VeteranIdentifier elements that have had documents added to the efolder that match the typeCategory and occurred on or after the from date and before the to date parameters.			

Table 64: findVeteransWithFolderModifications Faults		
Name	**Message**	**Notes**
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error. When the maximum result size is exceeded, the error message is returned: "Requested result set exceeds acceptable size"

6.2.11 findMovedItemsForVeteran

findMovedItemsForVeteran searches a Veteran's EFolder for move events. All non-expired moved events will be returned as a list of MovedItem elements. A move event is captured when a document is moved into or out of a Veteran's EFolder. A move event expires two years after the move date. Move events for restricted document types will not be reported by this service.

- findVeteransWithFolderModifications(criteria:TypeCategorySearchCriteria)

Table 65: findMovedItemsForVeteran				
	Name	**Type**	**Required?**	**Notes**
In:	Criteria	Veteranidentifier	One of either Veteran file number or Veteran SSN.	Veteran EDIPI is not supported in VBMS Release 13.0.
Out:	A list of MovedItem elements. An empty list will be returned when there are no move events for the requested Veteran.			

Table 66: findMovedItemsForVeteran Faults		
Name	**Message**	**Notes**
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

6.2.12 findPagedDocumentSeriesReferences

Searches for DocumentSeriesReference's, allowing for the results to be paged in order to limit the size and have more control over the response. Some Veterans have hundreds of thousands of documents in eFolder, making returning all documents at once cumbersome and untimely.

- findPagedDocumentSeriesReference (searchCriteria : DocumentSeriesReferenceSearchCriteria, pagingCriteria: PagingCriteria) : PagedDocumentSeriesReferences

Table 67: findPagedDocumentSeriesReference				
	Name	**Type**	**Required?**	**Notes**
In:	searchCriteria	DocumentSeriesReference SearchCriteria	Veteran is a required criterion	Criteria include veteran, seriesName, and supported metadata elements. seriesName expects a concatenation of Veteran's file number and name of file
In	pagingCriteria	PagingCriteria	Defaults are used if none are set.	
Out: PagedDocumentSeriesReferences				

Table 68: findPagedDocumentSeriesReference Faults		
Name	**Message**	**Notes**
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

7 Upload Service Interface

This service interface provides operations for uploading and updating documents. The upload service can be used to upload, update, and rename.

7.1 Upload Service Technical Specification

The following specification allows developers to discover the service for run-time consumption:

Table 69: Upload Service Technical Specification	
Item	**Details**
Service Invocation Type	SOAP over Hypertext Transfer Protocol(HTTP)
Service Interface Type	Web Services Description Language (WSDL) via Web Service 2.0
Service Name	eFolderUploadService
Interface	http://<service_domain>/vbms-efolder-svc/upload-v1/eFolderUploadService?wsdl
Schema	Linked in the WSDL
End Points	http://<service_domain>/vbms-efolder-svc/upload-v1/eFolderUploadService

7.2 Upload Service Interface Operations

The Upload Service operates in a two-phase approach. To upload a document, call initializeUpload with metadata to receive a token used in the second call, uploadDocument. To provide a hash of your content, one way would be to use Apache Commons Codec (version 1.7+) to do this job for you:

```
DigestUtils.sha1Hex(stringToConvertToSHexRepresentation)
```

Likewise, to update a document (provide a new version to a document series), call initializeUpdate with metadata to receive a token used in the second call, updateDocument.

VBMS eDocument and eFolder services do not enforce a document size limitation; however, document size has a dramatic impact on end user experience for VA personnel and external users. Systems uploading documents should attempt to keep documents as small as possible to reduce negative impact to both system resources and end users. For eDocument Service and eFolder Service uploads, 25MB files and below are recommended. Documents above 25MB have a higher chance to fail during these service operations or cause problems for VBMS end users when attempting to view these documents. VBMS may incorporate file size validations in a future release to enforce these recommendations.

7.2.1 initializeUpload

Accepts document metadata and returns a token (formatted as a UUID) that can be used as a parameter to call uploadDocument, which uploads the actual document content.

- initializeUpload (…): GUID

Table 70: initializeUpload Input Parameters			
Type	**Parameters**	**Description**	**Notes**
string	contentHash	The SHA-1/SHA-256 hash of your document content	Either SHA-1 or SHA-256 is required.
string	fileName	The document name intended to use as the destination file name in the eFolder	Required. Refer to the VBMS PDF Specification Document for details on document naming conventions.
VeteranIdentifier	veteranIdentifier	Veteran identifying information containing EDIPI, SSN, and/or fileNumber	Required.
string	docType	Document type information	Required. The document type ID, as listed in the Master Document Type and Category List, is the source of valid values.
Source	source	The source of the content that is being uploaded	Required. Represents the system that produced the document being uploaded (e.g., VDC, EBENEFITS, or SEP). Governed by ACR 635.
Date	vaReceiveDate	When did the VA receive the document	Required. Format: yyyy-mm-dd
MapEntry[]	seriesMetadata	Identifying characteristics of the document	For future use.
MapEntry[]	versionMetadata	Identifying characteristics of the document version	Optional. See Upload Service Accepted Metadata for further details on how to use this field.
Out:	uploadToken of type GUID This token must be used in a subsequent call to uploadDocument within 3 minutes of the client making a request to initializeUpload, otherwise the token expires and will be rejected when input into uploadDocument.		

Table 71: initializeUpload Faults		
Name	**Message**	**Notes**
		The document version {0} has been expunged and cannot be modified.
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

7.2.2 uploadDocument

Uploads the binary document content.

- uploadDocument (content : byte[], uploadToken : GUID) : uploadDocumentResponse

Table 72: uploadDocument				
	Name	**Type**	**Required?**	**Notes**
In:	content	xs:base64Binary	Yes	Recommended max file size is 25MB
In:	uploadToken	GUID	Yes	Must match the token returned by a previous call to initializeUpload
Out:	veteranReference	VeteranReference	—	VeteranReference containing the Veteran Identifier the document is associated to
Out:	typeCategory	TypeCategory	—	The type and category of the document
Out:	metadata	MapEntry	—	Contains Key/Value pairs; cardinality = 0 - 5000
Out:	vaReceiveDate	xs:date	—	Date VA received the document.
Out:	vbmsUploadDate	xs:date	—	Date uploaded into VBMS
Out:	source	Source	—	Represents the source of the document content (e.g., VDC, EBENEFITS, or SEP).
Out:	newDocumentVersionRefId	GUID	—	UUID of the DocumentVersionReference that will be created by the content sent by the client
Out:	newDocumentSeriesRefId	GUID	—	UUID of the DocumentSeriesReference that will be created by the content sent by the client
Out:	mimeType	string	—	The MIME type the service determined from the document content

Table 73: uploadDocument Faults		
Name	**Message**	**Notes**
veteranNotFoundFault	EFolderVeteranNotFoundException	No Veteran found matching the VeteranIdentifier supplied during initializeUpload
invalidMimeTypeFault	EFolderInvalidMimeTypeException	The MIME type determined from the content is not allowed to be uploaded into VBMS.
invalidSourceFault	EFolderInvalidSourceException	The source of the document content is not one recognized by VBMS.
invalidDocTypeFault	EFolderInvalidDocTypeException	The document type information supplied during initializeUpload is not recognized by VBMS. The list of acceptable values is governed by the Master Document Type Categories list, as found in ACR156.
invalidDateFault	EFolderInvalidDateException	A date field supplied during initializeUpload is not a valid date.
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

7.2.3 initializeUpdate

Accepts document metadata and returns a token (formatted as a UUID) that can be used as a parameter to call updateDocument. To update a document (i.e., provide a new version to a document series), call initializeUpdate with metadata to receive a token used in the second call, updateDocument.

If an initializeUpdate() request is called against a document in a document series that has been remediated by production support, the initializeUpdate() will fail with the following fault message: *The document version {0} has been expunged and cannot be modified*

- initializeUpdate (contentHash : String, existingVersion : String, vaReceiveDate : Date, versionMetadata : MapEntry) : GUID

Table 74: initializeUpdate

	Name	Type	Required?	Notes
In:	contentHash	string	Yes	The SHA-1/SHA-256 hash of your document content. One or the other is required.
In:	documentVersionReferenceId	string	Yes	ID of existing DocumentVersionReference that will be updated. Formatted as a UUID.
In:	vaReceiveDate	xs:date	Yes	Format: yyyy-mm-dd
In:	versionMetadata	MapEntry	No	Cardinality: 1 - 5000
Out:	uploadToken of type GUID			

Table 75: initializeUpdate Faults

Name	Message	Notes
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error.

7.2.4 updateDocument

Uploads the binary document content for a new version of a document series.

- updateDocument (content : byte[], updateDocumentToken : String) : updateDocumentResponse

Table 76: updateDocument

	Name	Type	Required?	Notes
In:	content	xs:base64Binary	Yes	Document content
In:	updateDocumentToken	GUID	Yes	Must match the token returned by a previous call to initializeUpload
Out:	veteranReference	VeteranReference	—	VeteranReference containing the Veteran Identifier the document is associated to
Out:	typeCategory	TypeCategory	—	The type and category of the document
Out:	metadata	MapEntry	—	element; cardinality = 0 - 5000

Table 76: updateDocument				
	Name	**Type**	**Required?**	**Notes**
Out:	vaReceiveDate	xs:date	—	Date VA received document
Out:	vbmsUploadDate	xs:date	—	Date VBMS uploaded document
Out:	source	Source	—	optional element. Represents the source of the document content (e.g., VDC, EBENEFITS, or SEP).
Out:	newDocumentVersionRefId	GUID	—	UUID of the DocumentVersionReference that will be created by the content sent by the client
Out:	documentSeriesRefId	GUID	—	UUID of the DocumentSeriesReference that this new document version belongs to.
Out:	mimeType	string	—	The MIME type the service determined from the document content

Table 77: updateDocument Faults		
Name	**Message**	**Notes**
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error

7.2.5 updateDocTitle

Change the name of an existing document version reference. This is possible only with documents you have uploaded. Attempts to edit the document title for a document your system did not upload will be rejected.

- updateDocTitle (documentVersionReferenceId : string, docTitle : String) : void

Table 78: updateDocTitle				
	Name	**Type**	**Required?**	**Notes**
In:	DocumentVersionReferenceId	string	Yes	UUID of the DocumentVersionReference you'd like to update
In:	DocTitle	string	Yes	Target title for document
Out:	—			

Table 79: updateDocTitle Faults		
Name	**Message**	**Notes**
eFolderFault	EFolderServiceException	This is a catch-all fault that will contain a UUID for identifying the fault and a message describing the error For attempts to update a document title you are not authorized to update, you will receive the message "Unauthorized to update document title"

7.3 Upload Service Accepted Metadata

Upload eFolder Service allows certain metadata to be specified when uploading or updating a document. For a list of metadata keys allowed during an upload/update of a document, you may invoke the eFolder Read Service operation findMetadataKeysForTypeCategory. Note that the allowable types of Metadata keys can change based on the document type, as specified in the field "docType" of initializeUpload.

Each key in the response of findMetadataKeysForTypeCategory has a maximum allowable length, as specified in attribute "maxLength" of the FormField object. Additionally, certain metadata keys have an expected pattern for their values. If any metadata passed into initializeUpload or initializeUpdate violate the maximum length or pattern requirements, then the service will respond with a SOAP fault indicating: The metadata value for key XXXX is in an invalid format. The following table shows metadata keys with pattern requirements:

Table 80: Metadata Keys with Pattern Requirements

Metadata Key	Max Length	Regular Expression
altDocType	200	^[a-zA-Z0-9\s\-/\`\~\=\+\[\]\{\}\#\^*<>\!\@\$\%&\(\)_|\;\:\"\'\,\.\?]*$
benefitTypeCompensation	1	^[01]*$
benefitTypePension	1	^[01]*$
benefitTypeSurvivorsPensionOrDIC	1	^[01]*$
Certified	96	--
claimantFirstName	64	^[a-zA-Z\s\-/]*$
claimantLastName	64	^[a-zA-Z\s\-/]*$
claimantMiddleName	64	^[a-zA-Z\s\-/]*$
claimantSSN	11	^[0-9]*$
claimPreviouslyFiledYN	1	^[yYnN]*$
Contention	1200	^[a-zA-Z0-9\s.\-_|\/\\@#~=%,;?!'"`():$+*\^\[\]&<>{}]*$
dateOfReceipt	40	^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])?$
dateOnDocument	10	^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])?$
DCS	20	^[a-zA-Z0-9\-]*$
dcsScanningComplete	5	(?i)^(t(rue)?|f(alse)?|[0-1]|y(es)?|n(o)?|on|off)$

Table 80: Metadata Keys with Pattern Requirements		
Metadata Key	**Max Length**	**Regular Expression**
documentRelocatorEmail	60	((^$))\|(^((([a-zA-Z]\|\d\|[!#\$%&'*\+\-\/=\^_`{\|}~]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])+(\.([a-zA-Z]\|\d\|[!#\$%&'*\+\-\/=\^_`{\|}~]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])+)*)\|((\x22)((((\x20\|\x09)*(\x0d\x0a))?(\x20\|\x09)+)?(([\x01-\x08\x0b\x0c\x0e-\x1f\x7f]\|\x21\|[\x23-\x5b]\|[\x5d-\x7e]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\|(\\([\x01-\x09\x0b\x0c\x0d-\x7f]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))*(((\x20\|\x09)*(\x0d\x0a))?(\x20\|\x09)+)?(\x22)))@((([a-zA-Z]\|\d\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\|(([a-zA-Z]\|\d\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-zA-Z]\|\d\|-\|\.\|_\|~\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-zA-Z]\|\d\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.)+(([a-zA-Z]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\|(([a-zA-Z]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-zA-Z]\|\d\|-\|\.\|_\|~\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-zA-Z]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.?$)
documentRelocatorId	64	^[0-9]*$
documentSeriesId	38	^(.)([0-9A-F]{8})-([0-9A-F]{4})-([0-9A-F]{4})-([0-9A-F]{4})-([0-9A-F]{12})(.)$
documentTypeId	10	^[0-9\:]*$
documentVersionId	38	^(.)([0-9A-F]{8})-([0-9A-F]{4})-([0-9A-F]{4})-([0-9A-F]{4})-([0-9A-F]{12})(.)$
emailAddressPreferred	60	((^$))\|(^((([a-zA-Z]\|\d\|[!#\$%&'*\+\-\/=\^_`{\|}~]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])+(\.([a-zA-Z]\|\d\|[!#\$%&'*\+\-\/=\^_`{\|}~]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])+)*)\|((\x22)((((\x20\|\x09)*(\x0d\x0a))?(\x20\|\x09)+)?(([\x01-\x08\x0b\x0c\x0e-\x1f\x7f]\|\x21\|[\x23-\x5b]\|[\x5d-\x7e]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\|(\\([\x01-\x09\x0b\x0c\x0d-\x7f]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))*(((\x20\|\x09)*(\x0d\x0a))?(\x20\|\x09)+)?(\x22)))@((([a-zA-Z]\|\d\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\|(([a-zA-Z]\|\d\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-zA-Z]\|\d\|-\|\.\|_\|~\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-zA-Z]\|\d\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.)+(([a-zA-Z]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\|(([a-zA-Z]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-zA-Z]\|\d\|-\|\.\|_\|~\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-zA-Z]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.?$)
EP	64	^[a-zA-Z0-9\s.\\\-_|\/@&><\(\)'\+\,;\$]*$
existingVersionId	38	^(.)([0-9A-F]{8})-([0-9A-F]{4})-([0-9A-F]{4})-([0-9A-F]{4})-([0-9A-F]{12})(.)$
fileName	255	^[\w`~=+#^@$\&\-_.{}{};\[\]]+[^\\\/:*?"><\|%,!]\.[\w]{3,4}$
fileNumber	64	^[0-9]{8,9}$
isNewMail	5	(?i)^(t(rue)?|f(alse)?|[0-1]|y(es)?|n(o)?|on|off)$
mailingAddressCurrent	200	^[a-zA-Z0-9\s+#@%&()_:',.\-/]*$
MimeType	96	--
poaCode	64	^[a-zA-Z0-9\s.\-_|\/\\@#~=%,;?!'"`():$+*\^\[\]&<>{}]*$
poaOrganization	64	^[a-zA-Z0-9\s.\-_|\/\\@#~=%,;?!'"`():$+*\^\[\]&<>{}]*$
ShippingNumber	64	^[0-9]*$

Table 80: Metadata Keys with Pattern Requirements		
Metadata Key	**Max Length**	**Regular Expression**
source	64	^[a-zA-Z\(\)\s']*$
sourceComment	255	--
spouseFileNumber	9	^[0-9]{8,9}$
spouseIsVeteran	5	(?i)^(t(rue)?|f(alse)?|[0-1]|y(es)?|n(o)?|on|off)$
spouseRelationshipEndDate	10	^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])?$
spouseRelationshipStartDate	10	^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])?$
subject	256	^[\w`'~=+#^@$\&\-_.,(){};\[\]]+[^\\\/:*?"><\|%,!]*$
systemSource	64	^[a-zA-Z\(\)\s']*$
telephoneNumberPreferred	15	^([1]?\D?(\d{3})\D?\D?(\d{3})\D?(\d{4}))?$
typeCategory	10	--
userEmail	64	((^$))\|(^((([a-zA-Z]\|\d\|[!#\$%&'*\+\-\/=\^_`{\|}~]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])+(\.([a-zA-Z]\|\d\|[!#\$%&'*\+\-\/=\^_`{\|}~]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])+)*)\|((\x22)((((\x20\|\x09)*(\x0d\x0a))?(\x20\|\x09)+)?(([\x01-\x08\x0b\x0c\x0e-\x1f\x7f]\|\x21\|[\x23-\x5b]\|[\x5d-\x7e]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\|(\\([\x01-\x09\x0b\x0c\x0d-\x7f]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))*(((\x20\|\x09)*(\x0d\x0a))?(\x20\|\x09)+)?(\x22)))@((([a-zA-Z]\|\d\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\|(([a-zA-Z]\|\d\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-zA-Z]\|\d\|-\|\.\|_\|~\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-zA-Z]\|\d\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.)+(([a-zA-Z]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\|(([a-zA-Z]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-zA-Z]\|\d\|-\|\.\|_\|~\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-zA-Z]\|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.?$)
userFacilityJob	64	^[A-Za-z\\\/\-\.\,'\(\)\s]*$
userFirstName	64	^[A-Za-z\/\-]*$
userId	64	^[A-Za-z0-9\\\/\-\.\,'\(\)\s]*$
userLastName	64	^[A-Za-z\/\-]*$
userMiddleName	64	^[A-Za-z\/\-]*$
userName	64	^[A-Za-z0-9\\\/\-\.\,'\(\)\s]*$
userRole	64	^[A-Za-z0-9\\\/\-\.\,'\(\)\s]*$
userSARL	1	^[0-9]*$
vbmsUploadDate	10	^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])?$
veteranDateOfBirth	10	^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])?$
VeteranFirstName	64	^[A-Za-z\/\-]*$
veteranGenderMF	1	^[mMfF]*$
VeteranLastName	64	^[A-Za-z\/\-]*$
VeteranMiddleName	64	^[A-Za-z\/\-]*$

Table 80: Metadata Keys with Pattern Requirements		
Metadata Key	**Max Length**	**Regular Expression**
veteranRepresentative	200	^[a-zA-Z0-9\s\-/\`\~\=\+\[\]\{\}\#\^*<>\!\@\$\%&\(\)_|\;\:\"\'\,\.\?\\]*$
veteranSSN	11	^[0-9]*$
VeteranSuffix	3	^[A-Za-z\/\-]*$
vvaDateOfStorage	10	^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])?$

In addition to the keys specified in the response to findMetadataKeysForTypeCategory, all uploads support a standard set of keys, as shown below. You may use these values if needed for your use case.

Table 81: eFolder Service Accepted Metadata		
Metadata Key	**Description**	**Notes**
Contention	The value scanned from "Contention" fields of a 526 form.	This can be specified up to 9 times on an individual upload. Each occurrence can be no longer than 128 characters.
DCS	The ID of the Document Control Sheet associated with this document.	If the value provided does not match the ID of an open Document Control Sheet, the initializeUpload or initializeUpdate call is rejected with error message "Document Control Sheet does not exist within the system."
dcsScanningComplete	true/false, indicating whether this upload is the last document associated with the Document Control Sheet identified with metadata "DCS".	If dcsScanningComplete is provided, DCS must also be provided. If this metadata is not provided, but "DCS" is, then this value is defaulted to "false".
isNewMail	true/false, indicating whether this document being uploaded is a new document that requires action by the VBMS user.	If this value is not provided, the default is "true".
documentTypeId	The ID of the document type from the Master Document Type Category List for the document being uploaded	This value is only recognized on initializeUpdate. It is ignored in initializeUpload.
sourceComment	The Source Comment value of the document.	—
spouseIsVeteran	Whether the Veteran's spouse is a Veteran, as scanned from the document.	Intended for use with forms VA 21-526, VA 21-527, VA 21-527EZ, and VA 21-686c only.
spouseFileNumber	The file number of the Veteran's spouse, as scanned from the document.	Intended for use with forms VA 21-526, VA 21-527, VA 21-527EZ, and VA 21-686c only. This or "spouseSSN" must be specified (but not both) if this is the first document for the Veteran with metadata key "spouseIsVeteran" and value "true". If a Veteran is not found with file number matching the metadata value, the upload is rejected.

Table 81: eFolder Service Accepted Metadata		
Metadata Key	**Description**	**Notes**
spouseRelationship StartDate	The date at which the Veteran's marriage began, as scanned from the document.	Intended for use with forms VA 21-526, VA 21-527, VA 21-527EZ, and VA 21-686c only. Must be in 'yyyy-mm-dd' format. If the date is in the future, or if 'spouseRelationshipEndDate' is specified with a date before this metadata value, the upload is rejected.
spouseRelationship EndDate	The date at which the Veteran's marriage ended, as scanned from the document.	Intended for use with forms VA 21-526, VA 21-527, VA 21-527EZ, and VA 21-686c only. Must be in 'yyyy-mm-dd' format. If the date is in the future, or if 'spouseRelationshipStartDate' is specified with a date after this metadata value, the upload is rejected.
spouseSSN	The SSN of the Veteran's spouse, as scanned from the document.	Intended for use with forms VA 21-526, VA 21-527, VA 21-527EZ, and VA 21-686c only. This or "spouseFileNumber" must be specified (but not both) if this is the first document for the Veteran with metadata key "spouseIsVeteran" and value "true". If a Veteran is not found with SSN matching the metadata value, the upload is rejected.

7.4 Upload Service Accepted Document Types

MimeType is the identifier for the file format. eDocument Service places limitations on the binary content that can be uploaded. The service will only accept content that represents acceptable MIME types, as listed in the table below. If the service determines the content is of a type other than one found in this list, the service will reject the upload.

Before making the document visible inside the VBMS UI, the service will convert all non-PDF documents into PDF.

Table 82: Accepted Document Types	
File Type	**MIME Types**
BMP	image/x-ms-bmp; image/bmp
JPG	image/jpeg; image/jp2
PDF	application/pdf
PNG	image/x-png; image/png
RTF	application/x-rtf; application/rtf; text/richtext
TEXT	text/plain
TIFF	image/tiff

7.4.1 MIME Type Matching Enforcement

If a file extension is provided in the value of the field *fileName* in the service method initializeUpload, the server will validate the binary content submitted during the corresponding

invocation of uploadDocument. If a mismatch is discovered between a file extension and the MIME type of the binary content, during the response to uploadDocument the server will respond with a fault, as follows:

Table 83: File Extension / MIME Type Mismatch Fault Properties	
Name	**Message**
code	EFSERR0020
message	Mime type of file does not match file extension

Whenever the fileName field of initializeUpload does not contain a file extension (i.e., it does not end with regular expression "\..*"), there is no enforcement of MIME type matching. However, the service will still enforce that the binary content is one of the allowed MIME types.

The following table represents the recognized file extensions and the allowable MIME types for that extension. If the binary content does not match any of the allowable MIME types for the file extension, then the uploadDocument call is rejected.

Table 84: Allowable MIME Types for each File Extension	
File Type	**MIME Types**
.bmp	image/bmp, image/x-ms-bmp
.jpg	image/jpeg
.pdf	application/pdf
.png	image/png
.rtf	application/rtf, text/richtext, application/x-rtf
.tif	image/tiff
.tiff	image/tiff
.txt	text/plain

8 eFolder Service Error Messages

Error messages are returned as SOAP faults to the service caller. The faultstring element within the fault can be reported to VBMS-Core support to facilitate troubleshooting.

Consult the sample XML below to understand the anatomy of an eService SOAP fault:

```
<soap:Fault>
  <faultcode>soap:Server</faultcode>
  <faultstring>Validation Errors GUID: b37257d1-c792-4e9c-848a-
f4dfadaaa759</faultstring>
  <detail>
    <ns1:EFolderFault xmlns:ns1="http://service.efolder.vbms.vba.va.gov/common">
      <ns1:code>EFSERR0001</ns1:code>
      <ns1:message>EFSERR0005 - Missing required field: Veteran Identifier.
</ns1:message>
      <ns1:type>API</ns1:type>
      <ns1:uuid>b37257d1-c792-4e9c-848a-f4dfadaaa759</ns1:uuid>
    </ns1:EFolderFault>
  </detail>
</soap:Fault>
```

Table 85: Error Types and Codes

Type	Code
SYSTEM_ERROR	EFSERR1001
SYSTEM_DETAIL_ERROR	EFSERR1002
FILE_NUMBER_NOT_FOUND	EFSERR1003
INVALID_DOC_TYPE	EFSERR1004
INVALID_DOC_CATEGORY	EFSERR1005
INVALID_DOC_SERIES_REFERENCE_ID	EFSERR1006
DOCUMENT_NOT_FOUND	EFSERR1007
VERSION_NOT_FOUND	EFSERR1008
CLAIM_NOT_FOUND	EFSERR1009
INVALID_STATE_CHANGE	EFSERR1010
INVALID_STATE_CHANGE_PARAMETER	EFSERR1011
INVALID_FILE_UPLOAD	EFSERR1012
DOCUMENT_ALREADY_UPLOADED	EFSERR1013
DOCUMENT_DELETED	EFSERR1067
UNKNOWN_BUSINESS	EFSERR1014
UNKNOWN_SYSTEM	EFSERR1015
INVALID_BGS_VETERAN	EFSERR1017
INVALID_DCS_ID	EFSERR1018
VA_RECEIVE_DATE_NOT_VALID	EFSERR1019
VA_RECEIVE_DATE_FUTURE_DATE	EFSERR1020
INVALID_DOCUMENT_SOURCE	EFSERR1021

Table 85: Error Types and Codes	
Type	**Code**
INVALID_LIFECYCLE_STATE	EFSERR1022
DOCUMENT_FAILURE	EFSERR1023
IN_PROCESS	EFSERR1024
PERMISSION_DENIED	EFSERR1025
INVALID_UNKNOWN_STATE	EFSERR1026
NO_OUTPUT_STREAM	EFSERR1027
DOCUMENT_SERIES_STAGING_ERROR	EFSERR1028
DOCUMENT_SERIES_NOT_FOUND	EFSERR1029
DOCUMENT_SERIES_CONTENT_NOT_FOUND	EFSERR1030
PDF_STREAM_NOT_FOUND	EFSERR1031
PDF_STREAM_UNABLE_TO_PROCESS	EFSERR1032
RETRIEVING_FILE_NUMBER	EFSERR1033
RETRIEVING_CLAIM_IDS	EFSERR1034
RETRIEVING_CLAIM_SUMMARY	EFSERR1035
DOCUMENT_FAILURE_ECM	EFSERR1036
INVALID_CONTENT_HASH	EFSERR1037
EMPTY_PACKAGE_ID	EFSERR1038
PACKAGE_NOT_FOUND	EFSERR1039
PACKAGE_ALREADY_COMPLETE	EFSERR1040
EMPTY_LIST	EFSERR1041
PACKAGE_ALREADY_EXISTS	EFSERR1042
PACKAGE_COUNT_MISMATCH	EFSERR1043
DUPLICATE_PROCESSING_ATTEMPT	EFSERR1044
REQUEST_TOO_LARGE	EFSERR1045
UPLOAD_OPERATION_MISMATCH	EFSERR1046
MISMATCH_SYSTEM_SOURCE_DOC_TITLE	EFSERR1047
INVALID_UPDATE_STATE	EFSERR1048
SPOUSE_IS_VET_START_DATE	EFSERR1049
SPOUSE_IS_VET_END_DATE	EFSERR1050
SPOUSE_IS_VET_DATE_IS_INVALID	EFSERR1051
SUPERSEDED_DOCUMENT	EFSERR1052
SPOUSE_VETERAN_NOT_FOUND	EFSERR1053
SPOUSE_CANNOT_BE_SELF	EFSERR1054
SPOUSE_IDENTIFIER_REQUIRED	EFSERR1055
CONTENTION_NOT_FOUND	EFSERR1056

Table 85: Error Types and Codes	
Type	**Code**
DATE_ON_DOC_INVALID	EFSERR1057
FILENET_CREATION_FAILED	EFSERR1058
DOCUMENT_ID_FILE_NUMBER_MISMATCH	EFSERR1059
SET_NOT_UNIQUE	EFSERR1060
ITEM_NOT_FOUND	EFSERR1061
RESTRICTED_DOCUMENT	EFSERR1065
MULTIPLE_CURRENT_DOC_VERSIONS	EFSERR1068
MULTIPLE_BRANCHES_FOR_CURRENT_DOC	EFSERR1069
METHOD_NOT_YET_IMPLEMENTED	EFSERR0000
MULTIPLE_VALIDATION_ERRORS	EFSERR0001
INVALID_FORMAT	EFSERR0002
INVALID_DATE_ORDER	EFSERR0003
INVALID_DATE_COMBINATION	EFSERR0004
REQUIRED_FIELD_MISSING	EFSERR0005
REQUIRED_FIELD_EMPTY	EFSERR0006
INVALID_METADATA_KEY	EFSERR0007
INVALID_CLAIM_ID	EFSERR0008
UNKNOWN_API	EFSERR0009
METADATA_KEY_BLANK	EFSERR0010
INVALID_FILENAME	EFSERR0011
EDIPI_NOT_SUPPORTED	EFSERR0012
NO_VALID_TOKEN	EFSERR0013
DOCUMENT_SERIES_STAGING_INVALID_INPUTS	EFSERR0014
INVALID_MIME_TYPE	EFSERR0015
VA_RECEIVE_DATE_NOT_FOUND	EFSERR0016
MULTIPLE_VETERAN_IDENTIFIERS	EFSERR0017
METADATA_VALUE_INVALID_FORMAT	EFSERR0018
DUPLICATE_DOC_TITLE	EFSERR0019
MIME_TYPE_MISMATCH	EFSERR0020
INVALID_EMPTY_DOCUMENT	EFSERR0021
PROBLEMATIC_TOKEN	EFSERR0022
METADATA_VALUES_INVALID_FORMAT	EFSERR0023
INVALID_CONTENTION	EFSERR0024
INVALID_CONTENTION_ID	EFSERR0025
INVALID_CONTENTION_COUNT	EFSERR0026

Table 85: Error Types and Codes	
Type	**Code**
METADATA_KEYS_DUPLICATES	EFSERR0027
INVALID_DATE_BOUNDS	EFSERR0028
INVALID_CONTENTION_DUPLICATE	EFSERR0029
INVALID_CONTENTION_CHARACTER	EFSERR0030
METADATA_IN_INCORRECT_LIST	EFSERR0031
UNABLE_TO_CONVERT_TO_PDF	EFSERR0032
DATE_FUTURE_DATE	EFSERR1062
DATE_NOT_FOUND	EFSERR1063
DATE_NOT_VALID	EFSERR1064
NUMBER_RANGE_INVALID	EFSERR1066

9 Policies and Qualities of Service

Please ensure you can meet VBMS security standards for system-to-system interactions.

9.1 Service Policies

To conform to VBMS security standards for system-to-system interactions, the following guidelines must be met for each interaction:

Double Encryption: Messages must be encrypted at least twice. For most messages, the first level of encryption is at the transport layer using two-way SSL between the calling application and the service host. The second level of encryption is performed at the application layer using standard WS-Security message encryption techniques.

Double encryption must not be used on operations uploadDocument and updateDocument.

Request and Response Signing: Messages are signed by the calling application to achieve non-repudiation and to enable preservation of message integrity. This signature encompasses the entire contents of the message body and the timestamp portion of the message header. Therefore, it is above and beyond any signatures that exist for particular elements of the message itself (such as the signature for the SAML assertion in the message headers). Standard eXtensible Markup Language (XML) Digital Signatures are used for signing contents of the XML request information. Response messages sent back to the client are signed by the web service application.

Message signatures must not be used on operations uploadDocument and updateDocument.

User Authentication via SAML: SAML assertions are used for validating prior authentication of the end-user or system responsible for the invocation of the web service operation. When the interaction is driven by an end-user action, SAML assertions should always be on behalf of the currently logged on user, and NOT on behalf of the system invoking the web service operation in response to the user's request. SAML assertions received by the calling application during a Web Single Sign-On (SSO) request while authenticating the end-user will be reused and sent in the WS-Security headers.

9.2 SAML Assertion Guidelines

SAML assertions generated by the IdP and delivered to the Web Service require the following guidelines. The BEP IdP is already configured to deliver assertions according to this specification and does not need to be changed to conform to these guidelines.

9.2.1 SAML 2.0 Profile

SAML assertions will conform to SAML 2.0 specifications for Web SSO, using the "urn:oasis:names:tc:SAML:2.0:assertion" namespace for all elements of the assertion.

9.3 Usage of "Bearer" Method

This method is currently used for Web SSO with the BEP IdP and is expected to provide adequate levels of security with the additional message signing and encryption laid out in the above guidelines. "Holder of Key" and "Sender Vouches" SAML profiles are not needed for our current approach.

9.4 Key Elements

Attributes delivered within the SAML assertion for end-users will include the following key elements needed for VBMS P2 applications:

- **Station Id:** Identifies the station used by the end-user for accessing the originating VBMS application.

Example:

```
<ns2:Attribute FriendlyName="stationId" Name="http://vba.va.gov/css/common/stationId">
<ns2:AttributeValue xmlns:xs=http://www.w3.org/2001/XMLSchema
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">317</ns2:AttributeValue>
</ns2:Attribute>
```

- **Security Level:** Identifies the security level associated with the end-user who is initiating the request from the calling application.

Example:

```
<ns2:Attribute FriendlyName="securityLevel"
Name="http://vba.va.gov/css/common/securityLevel">
<ns2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">7</ns2:AttributeValue>
</ns2:Attribute>
```

- **Role(s):** Identifies one or more roles granted to the end-user who is initiating the request from the calling application.

Example:

```
<ns2:Attribute FriendlyName="role" Name="http://vba.va.gov/css/vbms-core/role">
<ns2:AttributeValue xmlns:xs=http://www.w3.org/2001/XMLSchema
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
        xsi:type="xs:string">Rater </ns2:AttributeValue>
</ns2:Attribute>
<ns2:Attribute FriendlyName="role" Name="http://vba.va.gov/css/vbms-r/role">
<ns2:AttributeValue xmlns:xs=http://www.w3.org/2001/XMLSchema
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
        xsi:type="xs:string">Rater/Trainee</ns2:AttributeValue>
</ns2:Attribute>
```

The set of attributes within the SAML assertion for system-to-system interactions has not yet been finalized. VBMS-Core system administrators are responsible for generating and securely delivering long-term SAML tokens for remote systems.

9.5 Performance

Service levels may vary between services based on the specific technical and functional characteristics and requirements of the service.

9.5.1 Standards Compliance

The following table lists standards and versions as applicable.

Table 86: Standards Compliance

Standard	Version	Comments
Technical standards	N/A	
Business standards	N/A	
Regulatory standards	N/A	
Security standards	SAML 2.0	
SOAP	1.2	
TLS/SSL	TLS v1.1 and above	This update was an effort during 11.1.

9.5.2 VBMS-Provided SAML Tokens

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.

9.5.3 SAML Token Injection

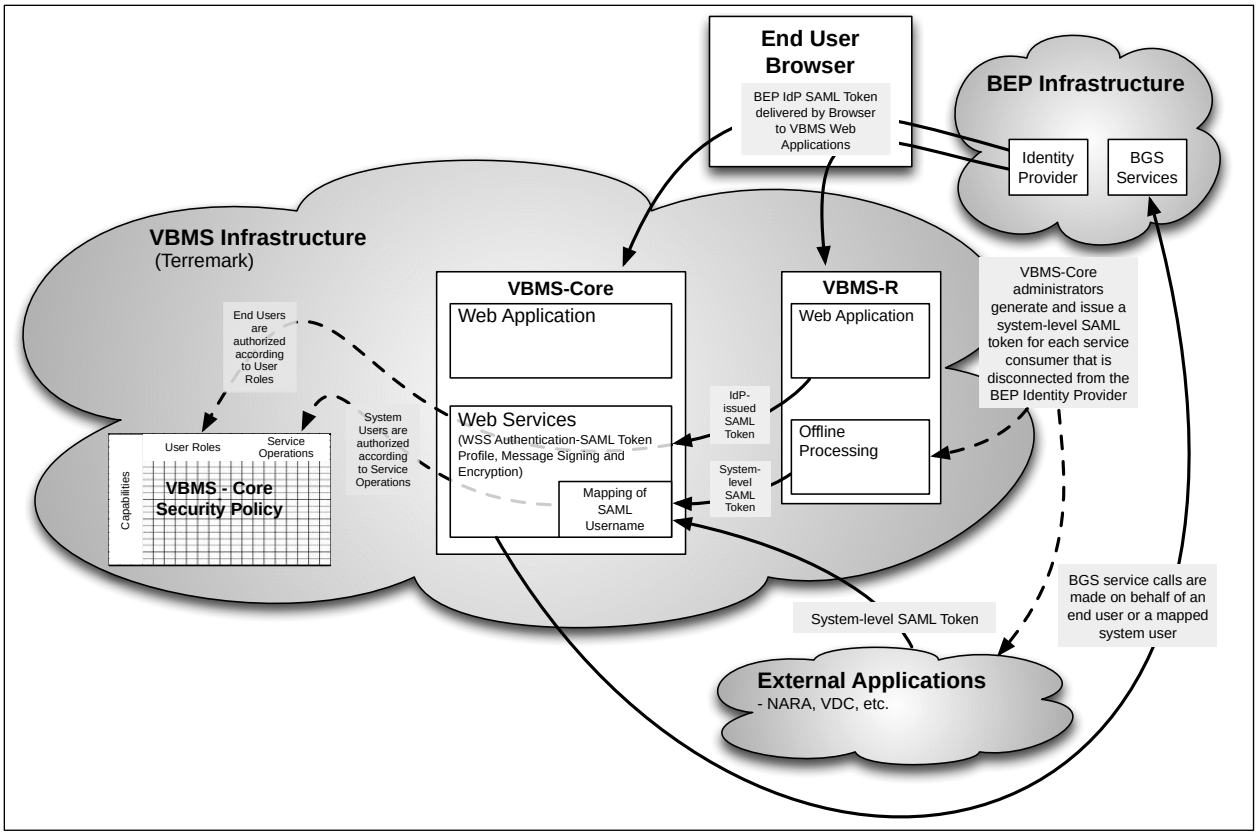
The eFolder Service Version 1.0 recognizes BEP-generated SAML tokens for authentication on behalf of end-users of client applications. The interaction method varies based on the type of service consumer:

- **Interactions Driven by End-users of the VBMS Web Applications:** VBMS web applications that invoke eFolder Service v1.0 must integrate with the trusted IdP within the BEP infrastructure (i.e., the BEP IdP). The IdPcollects 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.

The following figure shows the information flow for both interaction styles:

Figure 1: eFolder Service Credential Flow



Appendix A: Document Types and Roles

Information about current document types and the corresponding roles that are required to access those documents is governed by ACR 461. The master Document Type Category spreadsheet is attached to the Document Type Category Updates scope item epic for each release.

R14.1-ACR156v14-Document Type Updates (669698)

Appendix B: MTOM

MTOM is the W3C Message Transmission Optimization Mechanism, a method of efficiently sending binary data to and from Web services. MTOM is usually used with the XOP (XML-binary Optimized Packaging). VBMS uses MTOM to transmit file content through the eFolder & eDoc services.

There is a known development issue with .NET & MTOM. This issue will occur when attempting to interface with Read Service operations getDocumentContent and getDocumentContentAnnotations. To avoid this issue, .NET clients should interface with the Read Service exposed at http://<service_domain>/vbms-efolder-svc/read-v1/eFolderReadServiceInline?wsdl as described in the .NET Client Notes section.

.Net Client Notes

B-1 MTOM Problems

If you need MTOM turned off for document read, substitute the Inline read service for the read service. That is, use Identical Read Service with no MTOM:

http://<service_domain>/vbms-efolder-svc/read-v1/eFolderReadServiceInline?wsdl

B-2 Extra Fault Contract Attribute

The one thing specific to the current wsdl used to generate the client's service reference is when Visual Studio uses the read service wsdl to generate the service reference. It creates two identical fault contract attributes of type EFolderFault, which causes a compile error.

The extra fault contract attribute needs to be manually deleted every time the service reference is updated. This is found in the references file for the read service where the listTypeCategoryResponse object is defined. It will be obvious, as there are two identical lines defining this fault. This may change in the future with edits to the wsdl, but for now it is causing problems with .NET clients.

B-3 Problematic attributes within a complexType

Our upload service returns a complexType object that holds document specific attributes when completing an upload. The problem is that the data contract serializer used by WCF does not support the use of attributes within a complexType, which normally is not a problem because WCF will use the XmlSerializer instead, except that it can't because the DataContractSerializer is necessary for the design type of our wsdl.

Therefore, in order to capture the attributes of the uploaded document, we need to remove all occurrences of the individual attributes and replace them with a single array. The following needs to be changed in the reference file for the upload service (e.g. Reference.cs):

1. Remove the newDocumentVersionRefId, documentSeriesRefId, and mimeType variable declarations from the uploadDocumentResponse class.
2. Add a new XmlAttribute array in place of the three removed in step 1. (e.g. public System.Xml.XmlAttribute[] xmlAttributes;)
3. Change the uploadDocument and uploadDocumentResponse methods to accommodate these changes in steps 1 and 2.
4. When the call to the upload service is made it will have several parameters passed by reference, one of which is now the XmlAttribute array. After the uploadDocument is complete, iterate through the array to get the newDocumentVersionRefId, documentSeriesRefId, and mimeType.
5. Repeat steps 1-4 for the update methods.

Appendix C: SOAP Examples

initializeUpload request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:efol="http://service.efolder.vbms.vba.va.gov/eFolderUploadService"
xmlns:v5="http://vbms.vba.va.gov/cdm/document/v5">
   <soapenv:Header/>
   <soapenv:Body>
      <efol:initializeUpload>
         <contentHash>bf734c2b06caeb6308460e39874d2705653d2ca6</contentHash>
         <fileName>iot.pdf</fileName>
         <veteranIdentifier fileNumber="redacted"/>
         <docType>123</docType>
         <source>VBMS</source>
         <vaReceiveDate>2016-04-21-04:00</vaReceiveDate>
      </efol:initializeUpload>
   </soapenv:Body>
</soapenv:Envelope>
```

initializeUpload response:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
   <env:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
         <xenc:EncryptedKey Id="EK-5AF2272435F577F3BD147672522020428"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
            <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
1_5"/>
            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
               <wsse:SecurityTokenReference>
                  <ds:X509Data>
                     <ds:X509IssuerSerial>
                        <ds:X509IssuerName>CN=VBMS
Client,OU=VBA,O=VA,L=Charleston,ST=SC,C=US</ds:X509IssuerName>

<ds:X509SerialNumber>14400794291942239722</ds:X509SerialNumber>
                     </ds:X509IssuerSerial>
                  </ds:X509Data>
               </wsse:SecurityTokenReference>
            </ds:KeyInfo>
            <xenc:CipherData>

<xenc:CipherValue>KbUdhR0jj7Wk45T3DyHVD4mZAjKK/MWsxrmtRUdbWMC4Pae8lhZsJ8pZEN7RrInoW677
w39/ADPmArUMLVP8ScFQSfUP44TGU6vqnOLjrYJMk80Q2qcqoJ412BcVKdSGcmhWllLe4+c1ENJdYUSvUKTXaQ
MW2qYfbGLHrKoD3v2y0bSJsDPI1hUx/rtCA6gawHHedRp2lVkJ9MymsaLWo/DiKyEUHK28Q/+QxC7EX4Lqzh/o
XSO+nR7eA81YbuEZV8/gTsyACppPIeaxMDBQDAxqFwyyGIccFmUbuMSQjtpl+cDmS51QOPHt7l668EsV3d5W0x
t9GobKetoCbBHnyg==</xenc:CipherValue>
            </xenc:CipherData>
            <xenc:ReferenceList>
               <xenc:DataReference URI="#ED-28"/>
            </xenc:ReferenceList>
         </xenc:EncryptedKey>
         <wsu:Timestamp wsu:Id="TS-25">
            <wsu:Created>2016-10-17T17:27:00.188Z</wsu:Created>
            <wsu:Expires>2016-10-17T17:32:00.188Z</wsu:Expires>
         </wsu:Timestamp>
         <ds:Signature Id="SIG-27" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:SignedInfo>
               <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#">
```

```
                  <ec:InclusiveNamespaces PrefixList="env"
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
               </ds:CanonicalizationMethod>
               <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
               <ds:Reference URI="#TS-25">
                  <ds:Transforms>
                     <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#">
                        <ec:InclusiveNamespaces PrefixList="wsse env"
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                     </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <ds:DigestValue>RiN93hj5aAF+Lm9niIubOfej1BI=</ds:DigestValue>
               </ds:Reference>
               <ds:Reference URI="#id-26">
                  <ds:Transforms>
                     <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#">
                        <ec:InclusiveNamespaces PrefixList=""
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                     </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <ds:DigestValue>/qbpgCetS3o0QzOFKUML6gDbw70=</ds:DigestValue>
               </ds:Reference>
            </ds:SignedInfo>

<ds:SignatureValue>fhq7G86UJop5aOkS5y3TLwjyGxDQHUd2Fda1Dw7T2M7bKKBbEj4clzLIRM6NqmEA8tp
Nk9CyQsxR
rNVWiATKlcEOMfwkLrnDTjH0Z8kb5emM0EQUj/qFlsCFSsLMy4f3xA0AedblINu1uXEG639b8uGF
iKHwXSkclhjR8j+xqvfDyuiifaBh8406NLGwNKWSvWxjD5Tyc36DkLYCAwh8BvBa1J6fue/HgXCS
xiY4VzrKzNOav8NeZo9mCkmYvEI88TBsZ3l/D0bGhXSJEoLKG/eYRa8DjbVPiJQ83xjGbsLOP3kv
JiAZBj0vTaLpS3n02hXFnejZwMtoig0VZ5pk+g==</ds:SignatureValue>
            <ds:KeyInfo Id="KI-5AF2272435F577F3BD147672522018926">
               <wsse:SecurityTokenReference wsu:Id="STR-
5AF2272435F577F3BD147672522018927">
                  <ds:X509Data>
                     <ds:X509IssuerSerial>
                        <ds:X509IssuerName>CN=AIDE
CA2,OU=CA2,O=AIDE,L=Culpeper,ST=Virginia,C=US</ds:X509IssuerName>
                        <ds:X509SerialNumber>236</ds:X509SerialNumber>
                     </ds:X509IssuerSerial>
                  </ds:X509Data>
               </wsse:SecurityTokenReference>
            </ds:KeyInfo>
         </ds:Signature>
      </wsse:Security>
   </env:Header>
   <env:Body wsu:Id="id-26" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd">
      <initializeUploadResponse
xmlns="http://service.efolder.vbms.vba.va.gov/eFolderUploadService"
xmlns:com="http://vbms.vba.va.gov/cdm/common/v4"
xmlns:doc="http://vbms.vba.va.gov/cdm/document/v5"
xmlns:efc="http://service.efolder.vbms.vba.va.gov/common"
xmlns:efu="http://service.efolder.vbms.vba.va.gov/eFolderUploadService"
xmlns:part="http://vbms.vba.va.gov/cdm/participant/v4">
         <uploadToken xmlns="">{417B5CA6-BD8C-4976-90CD-271A6644D47D}</uploadToken>
      </initializeUploadResponse>
   </env:Body>
</env:Envelope>
```

uploadDocument request (with file as attachment):

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:efol="http://service.efolder.vbms.vba.va.gov/eFolderUploadService"
xmlns:ser="http://service.efolder.vbms.vba.va.gov/"
xmlns:v5="http://vbms.vba.va.gov/cdm/document/v5">
   <soapenv:Header/>
   <soapenv:Body>
      <efol:uploadDocument>
         <content>cid:iot.pdf</content>
         <uploadToken>{417B5CA6-BD8C-4976-90CD-271A6644D47D}</uploadToken>
      </efol:uploadDocument>
   </soapenv:Body>
</soapenv:Envelope>
```

uploadDocument response:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
   <env:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
         <wsu:Timestamp wsu:Id="TS-29">
            <wsu:Created>2016-10-17T17:27:22.042Z</wsu:Created>
            <wsu:Expires>2016-10-17T17:32:22.042Z</wsu:Expires>
         </wsu:Timestamp>
         <ds:Signature Id="SIG-31" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:SignedInfo>
               <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#">
                  <ec:InclusiveNamespaces PrefixList="env"
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
               </ds:CanonicalizationMethod>
               <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
               <ds:Reference URI="#TS-29">
                  <ds:Transforms>
                     <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#">
                        <ec:InclusiveNamespaces PrefixList="wsse env"
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                     </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <ds:DigestValue>Kjyjh56m1bI0q3k2KMMsPvDCFZI=</ds:DigestValue>
               </ds:Reference>
               <ds:Reference URI="#id-30">
                  <ds:Transforms>
                     <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#">
                        <ec:InclusiveNamespaces PrefixList=""
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                     </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <ds:DigestValue>2xOWTdPHKU0KX87cqV/eysdKdSk=</ds:DigestValue>
               </ds:Reference>
            </ds:SignedInfo>

<ds:SignatureValue>AnsIQD/ES1oo3szaM9QC1lRz9M6+eEF6YdalbOJac60oeazPoipbWR7Btpg3hXZncqY
I8Q3x38aR
UTWPSIaPUeM1ZL+I2j+rz24Sm2YmykuR0DN/MME+31pcwi8zsOudswrBb7LjxF/rfU9IW5IxhQqZ
gp3VEddfxvMY/fzmbCFA8769V5BHF/GPyr8mL0NXnmaRg+c7hd6TBxrB8MHTpnQtHCWd8Iee2bUt
RaXRfJm9zfhV35sAUQTPGzoSFxmdMQFkOD7KhtCzHHbgnOFV6Z79VsenHKyolIlTrVZgoULiUZzQ
```

```
UsMiC+gagynNqUZR8x8UoB/Rmrj3IfQNecZJ/Q==</ds:SignatureValue>
            <ds:KeyInfo Id="KI-5AF2272435F577F3BD147672524204230">
               <wsse:SecurityTokenReference wsu:Id="STR-
5AF2272435F577F3BD147672524204231">
                  <ds:X509Data>
                     <ds:X509IssuerSerial>
                        <ds:X509IssuerName>CN=AIDE
CA2,OU=CA2,O=AIDE,L=Culpeper,ST=Virginia,C=US</ds:X509IssuerName>
                        <ds:X509SerialNumber>236</ds:X509SerialNumber>
                     </ds:X509IssuerSerial>
                  </ds:X509Data>
               </wsse:SecurityTokenReference>
            </ds:KeyInfo>
         </ds:Signature>
      </wsse:Security>
   </env:Header>
   <env:Body wsu:Id="id-30" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd">
      <uploadDocumentResponse documentSeriesRefId="{F8C05DD7-6B91-48DB-B5A1-
0F0C2E6EF069}" mimeType="application/pdf" newDocumentVersionRefId="{33A3E379-BFD8-
4E02-87A2-C32B3C8EE601}"
xmlns="http://service.efolder.vbms.vba.va.gov/eFolderUploadService"
xmlns:com="http://vbms.vba.va.gov/cdm/common/v4"
xmlns:doc="http://vbms.vba.va.gov/cdm/document/v5"
xmlns:efc="http://service.efolder.vbms.vba.va.gov/common"
xmlns:efu="http://service.efolder.vbms.vba.va.gov/eFolderUploadService"
xmlns:part="http://vbms.vba.va.gov/cdm/participant/v4">
         <veteranReference xmlns="">
            <doc:id fileNumber="redacted"/>
         </veteranReference>
         <typeCategory categoryDescriptionText="Income" categoryId="29"
typeDescriptionText="VA 21-4185 Report of Income from Property or Business"
typeId="123" typeLabelText="L114" xmlns=""/>
         <vaReceiveDate xmlns="">2016-04-21-04:00</vaReceiveDate>
         <vbmsUploadDate xmlns="">2016-10-17-04:00</vbmsUploadDate>
      </uploadDocumentResponse>
   </env:Body>
</env:Envelope>
```

Appendix D: DateTimeRange Formatting

Should be in the format one of the following:

- UTC Time: YYYY-MM-DDThh:mm:ssZ" or
- Negative Offset to UTC Time: YYYY-MM-DDThh:mm:ss-hh:mm"
- Positive Offset to UTC Time: YYYY-MM-DDThh:mm:ss+hh:mm"

Where:

- YYYY indicates the year
- MM indicates the month
- DD indicates the day
- T indicates the start of the required time section
- hh indicates the hour
- mm indicates the minute
- ss indicates the second
- Z indicates UTC time

Appendix E: Allowable Content Source Values

The following is a list of the allowable values for content source. When filling the source value of an initializeUpload call, one of the following values must be used, otherwise the upload will be rejected.

- Accurint
- Arcis
- BDN
- BFILite
- CACI
- CAPRI
- CAPRI Integration
- CAPRI User IEN
- CAPRI VISTA
- CEN
- County Court Records
- CRM
- CRMUD
- CSRA (CBCM)
- CSRA (CM)
- CSRA (RMC-ESS)
- CSRA (SC)
- D2D
- DFAS
- DMC
- DOD Integration
- DPRIS
- EBN
- eBenefits
- Exam Track
- FAX
- IRIS
- Kofax
- LHI
- Military Services
- MSLA
- National Call Center (NCC)
- NPRC (PIES)
- Office of Workman's Compensation
- PCGL
- PCTC
- PMR Program
- PTSD Stressor Verification Site
- Public Contact Box
- QTC
- RBPS
- RMC-SCAN
- RMC-VALO
- RRC
- SCAN
- SCAN_S
- SEP
- SMRTS
- SMS
- SSA
- USER
- USER_EDIT
- USER_FORM
- USER_MDU
- VACOLS
- VADIR (VIS)
- VBMS
- VBMS-A
- VBMS-R
- VDC
- VERIS
- VISTA
- VHA_CAPRI
- VHA_CUI
- VONAPP
- Veteran
- VetFed
- Virtual VA
- VVABFI
- VVAUSER
- VRE