Toshiba Software India Pvt Ltd
Document Solutions Engineering

BoxDocument Design Description

version: 1.23

Creation Date: 20/07/2016

Approvals

Div./Dept.	Title	Signature	Date
TSIP	VP/GM		
TSIP/Marketing	Director, Marketing and Administration		
	Product Manager		
TSIP/Engineering	Director, SW Engineering		
	Mgr. Software		
TSIP/Quality Control	Mgr. Software QA		

Table of Contents

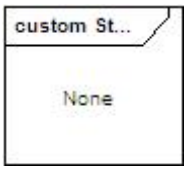
1. Introduction

1.1. PURPOSE

The purpose of this document is to describe the design of BoxDocument. This design specification should be for the design and development of BoxDocument.

1.2. DOCUMENT CONVENTIONS

Icons that contain the text None identify sections that do not require a graphical illustration



1.3. OVERVIEW OF THE DOCUMENT

This section presents the organization of this document, indicating the contents of each chapter.

Chapter 1 Introduction

Chapter 1 presents the purpose of the document its structure, revision history and reference information.

Chapter 2 Use Cases

Chapter 2 presents use cases that apply to the component.

Chapter 3 Design Methodology

Chapter 3 presents the necessary information content and organization of a Software Design Description.

Chapter 4 Interface Description View

Chapter 4 presents the details of the component's external and internal interfaces.

Chapter 5 Detailed Design Description View

Chapter 5 presents the component's internal details.

Note: Please be aware that the program interfaces, direct interface and messaging interface are now part of the architecture design document.

1.4. REVISION HISTORY

Version	Date	Author	Notes
1.18	20-07-2016	Shweta	Updated Section 5.1 to update specification of extract operation on different models (ebx, ebn, caspian). Removed the performance details from section 5.1
1.19	8/9/2017	Sudheer	Updated the Section 4.3.7 for GetViewPageList(), PutSystemFile(), PutSystemFile(), MovePage(), GetRotation(), Rotate(),
1.2	31/10/2017	Sudheer	Updated the Section 4.3.5 for PasteDocument(CString &documentname)
1.21	10/01/2018	Sudheer	Updated the Section 5.1 for Case study of EBX_DTFR_17328
1.22	10/05/2018	Atul	Updated the Section 4.3.3(BoxDocument::Initialize()) for Case study of EBX_DTFR_18627
1.23	13/08/2018	Atul	Updated Section: 5.1(Theory Of Operation) Section: 4.3.3 (Method name : CreateBox , CreateDocument) Section: 4.3.7 (Method name : CreatePage)

1.5. REFERENCES

Document	Version	Notes

1.6. TERMS AND DEFINITIONS

Term	Definition

2. Use Case Realization

This chapter presents use cases that apply to this component

2.1. USE CASES

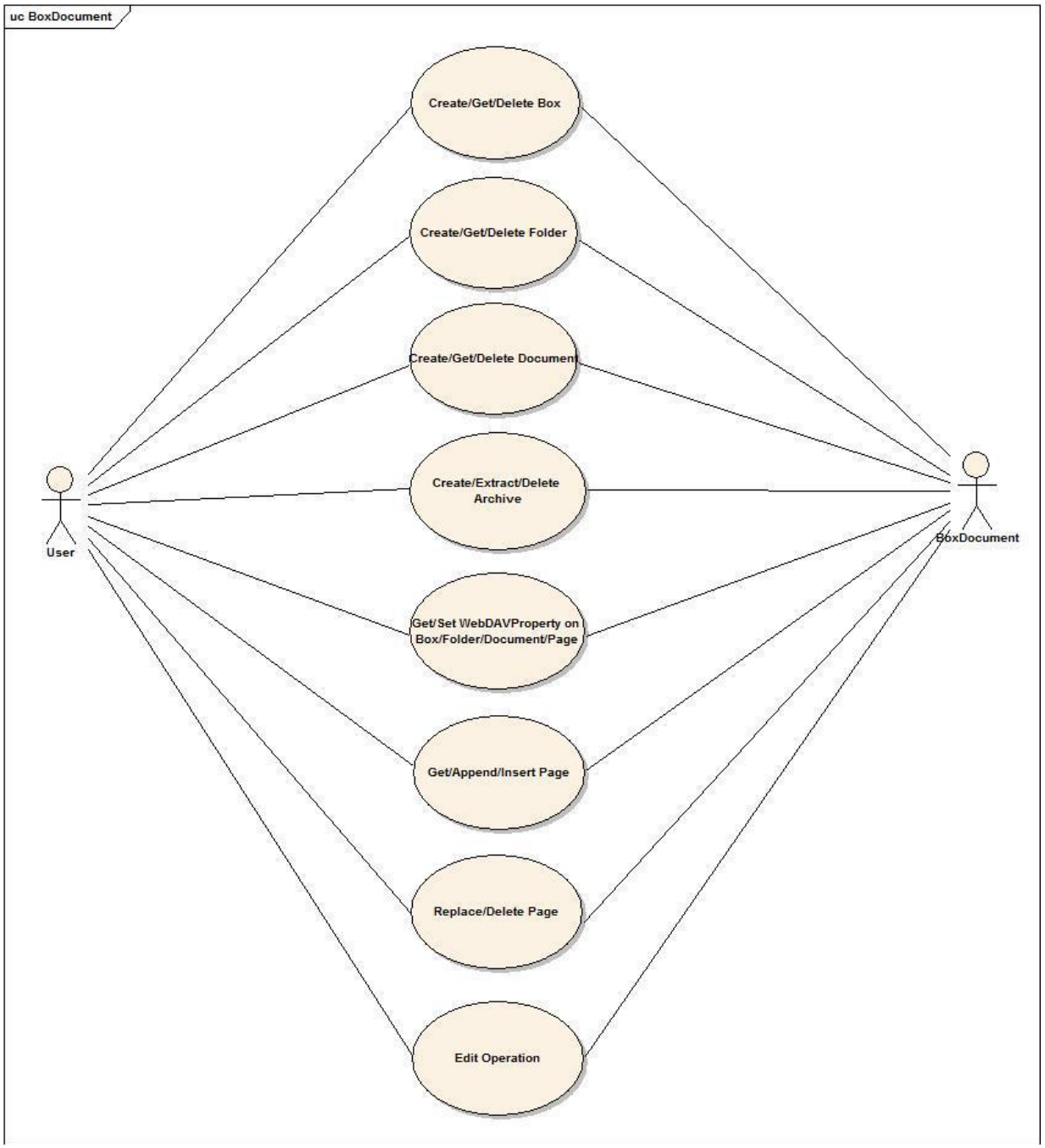


Figure 1: BoxDocument

BoxDocument interface provides a set of API's to store the documents on the WebDAV server in a particular folder structure. It can be used to create and manage Box/Folder/ Document/Page. It uses DocumentStore to interact with the WebDAV server.

2.1.1. CI::BoxDocument::Create/Extract/Delete Archive

Pre Conditions

For Create Archive, Document should be present.
For Extract/Delete Archive the archived file should exist
Server should be up and running

Realization

```
1) Archive  documents in a box/folder into a zip file. EFB file format is used to zip a document, which
is a specialized form of zip .The main zip file consists of the EFB files.
2) Extract the archive file to any box/folder. All the documents are extracted.
3) Delete the archive file
```

Post-conditions

On CreateArchive, an archive file is created.
On DeleteArchive, the archive file is deleted.
On ExtractArchive, the archive file is extracted to a box/folder.

2.1.2. CI::BoxDocument::Create/Get/Delete Box

Pre Conditions

For Create Box, Box should not exist on the server
For Get/Delete Box, Box should be present on the server
Server should be up and running

Realization

```
1) Create a box on the server and set the properties on the box.
2) Get the Box, if it exists on the server and load the properties of the Box.
3) Delete the Box, if present on the server.
```

Post-conditions

Box is Created/Retrieved/Deleted from the server

2.1.3. CI::BoxDocument::Create/Get/Delete Document

Pre Conditions

For Create Document, document should not exist on the server
For Get/Delete Document, document should be present on the server
Server should be up and running

Realization

```
1) Create a document on the server and set the properties on the document.
2) Get the document, if it exists on the server and load the properties of the document.
3) Delete the document, if present on the server.
```

Post-conditions

Document is Created/Retrieved/Deleted from the server

2.1.4. CI::BoxDocument::Create/Get/Delete Folder

Pre Conditions

For Create Folder, folder should not exist on the server
For Get/Delete Folder, folder should be present on the server
Server should be up and running

Realization

```
1) Create a folder on the server and set the properties on the folder.
2) Get the folder, if it exists on the server and load the properties of the folder.
3) Delete the folder, if present on the server.
```

Post-conditions

Folder is Created/Retrieved/Deleted from the server

2.1.5. CI::BoxDocument::Edit Operation

Pre Conditions

Documents/Pages to be Cut/Copied/Pasted should exist
For Cut/Paste Page operation, the document should be in Edit Mode
Server should be up and running

Realization

```
1)  Cut/Copy/Paste Documents - The documents are copied to a temporary path upon Copy/Cut. On Paste, the
Documents in the temporary path are copied to the new location. The document cut is deleted from the
original location.
2) Edit/CancelEdit/Save/SaveAs Document - A Document has to be in Edit mode for Pages to be Cut/Pasted.
The Cut/Copy/Paste operations can be saved or canceled. On Edit the document is copied to temporary
path. Further operations are performed on this document. Upon CancelEdit, the document in the original
is restored and the temporary document is deleted. Upon Save/SaveAs the document in the temporary path
is overwritten on the original document.
3) Cut/Copy/Paste Pages - The pages are copied to a temporary path upon Copy/Cut. On Paste, the Pages in
the temporary path are copied to the new location. The Pages cut is deleted from the original location.
```

Post-conditions

Document is Cut/Copied/Pasted
Document is Saved/Canceled upon Editing
Pages are Cut/Copied/Pasted

2.1.6. CI::BoxDocument::Get/Append/Insert Page

Pre Conditions

For Get Page, page should exist
For Insert Page, position should not be at the end of the document
Server should be up and running
The Document should exist

Realization

```
1) Get the Page, if it exists on the server and load the properties of the page.
2) Append the new page at the end and update the document and page properties.
3) Insert the new page at the specified position. Sequence the other pages accordingly. Update the page
properties
```

Post-conditions

Page is Retrieved/Appended/Inserted

2.1.7. CI::BoxDocument::Get/Set WebDAVProperty on Box/Folder/Document/Page

Pre Conditions

Box/Folder/Document/Page should be present on the server
Server should be up and running
To Get a property, the property should exist on the Box/Folder/Document/Page

Realization

```
1) Set the property for Box/Folder/Document/Page on the server
2) Get the property of the Box/Folder/Document/Page present on the server
```

Post-conditions

Property is got from the Box/Folder/Document/Page
Property is set on the Box/Folder/Document/Page

2.1.8. CI::BoxDocument::Replace/Delete Page

Pre Conditions

Document should exist
Page to be replaced/deleted should exist
Server should be up and running

Realization

```
1) Replace a page with a new page and update the page properties of the document.
2) Delete a page on the server and sequence the other pages. Update the page properties of the document
```

Post-conditions

Page is replaced/deleted

3. Design Methodology

The design description is presented in terms of two views, the Interface Description view and the Detail Design Description view, as described in the IEEE standard 1016-1998, IEEE Recommended Practice for Software Design Descriptions.

The Decomposition Description and Dependency Description views, which are mentioned in the same IEEE standard, are relegated to the product's architecture specification.

3.1. COMPONENT ATTRIBUTES

A component attribute is a named characteristic or property of a component. It provides a statement of fact about the component. The design view of a component is presented in terms of the attributes characteristic to the respective view. The following attributes are relevant to the design views presented in this document

Identification

This attribute represents the name of the component. Two components shall not have the same name and these names may be selected to characterize their nature. This will simplify referencing and tracking in addition to providing identification.

Function

This attribute represents a statement of what the component does. The function attribute shall state the transformation applied by the component to inputs to produce the desired output. In the case of a data component, this attribute shall state the type of information stored or transmitted by the component.

Interface

This attribute represents a description of how other components interact with this component. The interface attribute shall describe the methods of interaction and the rules governing those interactions. The methods of interaction include the mechanisms for invoking or interrupting the component, for communicating through parameters, common data areas or messages, and for direct access to internal data. The rules governing the interaction include the communications protocol, data format, acceptable values, and the meaning of each value.

This attribute shall provide a description of the input ranges, the meaning of inputs and outputs, the type and format of each input or output, and output error codes. For information systems, it should include inputs, screen formats, and a complete description of the interactive language.

Processing

This attribute represents a description of the rules used by the component to achieve its function. The processing attribute shall describe the algorithm used by the component to perform a specific task and shall include contingencies. This description is a refinement of the function attribute. It is the most detailed level of refinement for this component.

This description should include timing, sequencing of events or processes, prerequisites for process initiation, priority of events, processing level, actual process steps, path conditions, and loop back or loop termination criteria. The handling of contingencies should describe the action to be taken in the case of overflow conditions or in the case of a validation check failure.

Data

This attribute represents a description of data elements internal to the component. The data attribute shall describe the method of representation, initial values, use, semantics, format, and acceptable values of internal data.

The description of data may be in the form of a data dictionary that describes the content, structure, and use of all data elements. Data information shall describe everything pertaining to the use of data or internal data structures by this component. It shall include data specifications such as formats, number of elements, and initial values. It shall also include the structures to be used for representing data such as file structures, arrays, stacks, queues, and memory partitions.

The meaning and use of data elements shall be specified. This description includes such things as static versus dynamic, whether it is to be shared by transactions, used as a control parameter, or used as a value, loop iteration count, pointer, or link field. In addition, data information shall include a description of data validation needed for the process

3.2. DESIGN VIEWS

The description of a design can be organized in views that reveal design details from different perspectives and viewpoints. Together these views provide a comprehensive description of the design in a comprehensive and concise manner.

Example of a parameter table:

Operations

Method name: Acquire

Documentation:
Returns a new Client interface instance for a unique service name parameter. @param MsgPortRef - Service port that the Client interface uses to send messages to SSM. @param CString - Service name. @return Client interface instance.

Return type:
ClientRef

Parameters:
CI::MessagingSystem::MsgPortRef servicePort
CI::OperatingEnvironment::CString serviceName

4. Interface Description View

Scope

The interface description provides everything designers, programmers, and testers need to know to correctly use the functions provided by a component. This description includes the details of external and internal interfaces not provided in the software requirements specification.

Use

The interface description serves as a binding contract among designers, programmers, customers, and testers. It provides them with an agreement needed before proceeding with the detailed design of components. In addition, the interface description may be used by technical writers to produce customer documentation or may be used directly by customers. In the latter case, the interface description could result in the production of a human interface view.

Representation

The interfaces in this section are documented according to the types found in the master UML model.

4.1. COMPONENT IDENTIFICATION

CI::BoxDocument

4.2. FUNCTION

BoxDocument interface provides a set of API's to store the documents on the WebDAV server in a particular folder structure. It can be used to create and manage Box/Folder/Document/Page. It uses DocumentStore to interact with the WebDAV server.

4.3. INTERFACES

4.3.1. CI::BoxDocument::Archiver

Archiver class provides facilities to create archive.

Operations	
Method name:	~Archiver
Method name:	Cancel
Documentation:	

Operations

```
cancel to archive
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:

Method name: GetPath

Documentation:
```
get a path of created archive
@param[out] path - archive path
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& path

Method name: GetProgress

Documentation:
```
get archiving progress
@param[out] progress - archiving progress [0-100] %
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
int& progress

Method name: GetStatus

Documentation:
```
get archiving status
@param[out] status - archiving status
@return STATUS_OK on success,
STATUS_FAILED on failure.
STATUS_ARCHIVE_SIZE_ERROR if the size of file goes beyond 2GB.
```

Return type:
Status

Parameters:
ArchiveStatus& status

4.3.2. CI::BoxDocument::Box

```
Box class provides facilities of BOX operation.
```

Operations

Method name: ~Box

Method name: GetCreationDate

Documentation:
```
get the boxcreation date
@param[out] date - creation date of the box
@return STATUS_OK on success,
        STATUS_FAILED on failure.
@NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
CString date

Method name: GetDateFormat

Documentation:
```
 get the date format for the received forwarding
 @param[out] df - get the date for the received forwarding
@return STATUS_OK on success,
         STATUS_FAILED on failure.
 @NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
RFDateFormat & df

Method name: GetDocumentCount

Documentation:
```
get the number of document present inside a box
@param[out] numDoc - get the number of documents in a box
@return STATUS_OK on success,
        STATUS_FAILED on failure.
@NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
unsigned int numDoc

Method name: GetFileNameFormat

Documentation:
```
get the file name format for the received forwarding
 @param[out] fnf - get the file name for the received forwarding
 @return STATUS_OK on success,
        STATUS_FAILED on failure.
 @NOTE - It is available via GetBoxList
```

Return type:

Operations

Status

Parameters:
RFFileNameFormat & fnf

Method name: GetFolderCount

Documentation:
```
get the number of folder inside a box
@param[out] numFolder - get the number of folder in a box
@return STATUS_OK on success,
         STATUS_FAILED on failure.
@NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
unsigned int numFolder

Method name: GetName

Documentation:
```
get a name of the container
@param[out] name - a name of the container
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& name

Method name: GetNotificationEmailId

Documentation:
```
get the notification email id
@param[out] emailId - get the nofication email id
@return STATUS_OK on success,
       STATUS_FAILED on failure.
@NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
CString emailId

Method name: GetNumber

Documentation:
```
get a number of the box
@param[out] number - a number of the box
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Operations

Return type:
Status

Parameters:
CString& number

Method name: GetPageNumberFormat

Documentation:
```
get the page number format for the received forwarding
@param[out] pnf - get the page number format for the received forwarding
@return STATUS_OK on success,
         STATUS_FAILED on failure.
@NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
RFPageNumberFormat pnf

Method name: GetPageNumberFormat

Documentation:
```
get the page number format for the received forwarding
@param[out] pnf - get the page number format for the received forwarding
@return STATUS_OK on success,
         STATUS_FAILED on failure.
@NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
RFPageNumberFormat & pnf

Method name: GetRFComment

Documentation:
```
get comment for the received forwarding..
@param[out] comment - return the comment
@return STATUS_OK on success,
         STATUS_FAILED on failure.
 @NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
CString & comment

Method name: GetSubID

Documentation:
```
 get the sub ID for the received forwarding
 @param[out] sid - get the sub ID for the received forwarding
```

Operations

```
 @return STATUS_OK on success,
         STATUS_FAILED on failure.
 @NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
RFSubID & sid

Method name: GetWEPDocument

Documentation:

```
 get WEP document ID
  BoxDocument create HDB document and return document ID.
 @param[out] documentID - HDB document ID
 @return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
 @note user need to delete this document using document ID.
```

Return type:
Status

Parameters:
CString& documentID

Method name: SetDateFormat

Documentation:

```
 set the date format for the received forwarding
 @param[in] df - get the date for the received forwarding
 @return STATUS_OK on success,
        STATUS_FAILED on failure.
         STATUS_DISK_FULL if there is not enough space in the partition.
 @NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
RFDateFormat df

Method name: SetFileNameFormat

Documentation:

```
 set the file name format for the received forwarding
 @param[in] fnf - set the file name for the received forwarding
 @return STATUS_OK on success,
         STATUS_FAILED on failure.
        STATUS_DISK_FULL if there is not enough space in the partition.
 @NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:

Operations

RFFileNameFormat fnf

Method name: SetPageNumberFormat

Documentation:

```
set the page number format for the received forwarding
 @param[in] pnf - get the page number format for the received forwarding
 @return STATUS_OK on success,
        STATUS_FAILED on failure.
         STATUS_DISK_FULL if there is not enough space in the partition.
 @NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
RFPageNumberFormat pnf

Method name: SetRFComment

Documentation:

```
 set comment for the received forwarding.
 @param[in] comment - comment to be set
@return STATUS_OK on success,
        STATUS_FAILED on failure.
        STATUS_DISK_FULL if there is not enough space in the partition.
@NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
CString comment

Method name: SetSubID

Documentation:

```
set the sub ID for the received forwarding
 @param[in] sid - get the Sub ID for the received forwarding
@return STATUS_OK on success,
         STATUS_FAILED on failure.
         STATUS_DISK_FULL if there is not enough space in the partition.
 @NOTE - It is available via GetBoxList
```

Return type:
Status

Parameters:
RFSubID sid

Method name: SetWEPDocument

Documentation:

```
set HDB Document of Workflow Execution Parameter to Box.
@param[in] node - WEP node to save
@return STATUS_OK on success,
```

Operations

```
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
DOM::NodeRef node

4.3.3. CI::BoxDocument::BoxDocument

```
BoxDocument class provides to get BOX instance
```

Operations

Method name: Acquire

Documentation:
```
@deprecated -- Please use the new Acquire . Retained only for backward compatibility.
Returns a reference to a BoxDocument
@return reference to BoxDocument interface
```

Return type:
BoxDocumentRef

Parameters:

Method name: Acquire

Documentation:
```
Returns a reference to a BoxDocument
Updates the session ID if present else creates a new sessionID and returns it to the user.
@return reference to BoxDocument interface
```

Return type:
BoxDocumentRef

Parameters:
CString& sessionId

Method name: ~BoxDocument

Method name: Cleanup

Documentation:
```
Cleanup all WAITING Documents.
The method search each box and delete WAITING Document.
@return STATUS_OK on success.
        STATUS_FAILED_INITIALIZATION when uninitialized
```

Return type:
Status

Parameters:

Operations

Method name: CreateBox

Documentation:

```
create new box and updates the properties member map in box/* object scope.
@param[out] box - instance of Box class
@param[in] boxbasepath - box type e.g. "eFilingboxes"
@param[in] boxnumber - string of 1-20 digits box number
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
        STATUS_MAX_ALLOWED_RESOURCES_REACHED if the resource limit is reached
```

Return type:

Status

Parameters:

BoxRef& box
CString boxbasepath
CString boxnumber

Method name: CreateDocument

Documentation:

```
create document instance by resource path and updates the properties member map in box/* object scope.
the name of document is least number of non-exist documents.
if actual boxes or folders does not exist, these folders will be created automatically.
@param[out] doc - instance of Document class
@param[in] boxbasepath - box type e.g. "eFilingboxes"
@param[in] boxnumber - string of 1-20 digits box number
@param[in] foldername - folder name.
@param[in/out] documentname - on input, it is document name expected by
                              users. on output, it's created document
                              name. if document has the same name
                              exists, suffix will be added.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
        STATUS_MAX_ALLOWED_RESOURCES_REACHED if the resource limit is reached
```

Return type:

Status

Parameters: *DocumentRef&*

doc *CString* boxbasepath
CString boxnumber
CString foldername
CString& documentname

Method name: DeleteBox

Documentation:

```
delete the box
if a document in the box is using, it will fail(some documents are
 deleted).
@param[in] boxbasepath - box type e.g. "eFilingboxes"
@param[in] boxnumber - string of 1-20 digits box number
```

Operations

```
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString boxbasepath
CString boxnumber

Method name: DeleteDocument

Documentation:

```
delete document
document status will be changed to DELETING. after that, document will
be deleted.
if document status is NOT READY or EDITING, it will fail.
@param[in] boxbasepath - box type e.g. "eFilingboxes"
@param[in] boxnumber - string of 1-20 digits box number
@param[in] foldername - folder name.
@param[in] documentname - serial number from "00000".
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString boxbasepath
CString boxnumber
CString foldername
CString documentname

Method name: DeleteDocument

Documentation:

```
delete document
document status will be changed to DELETING. after that, document will
be deleted.
if document status is NOT READY or EDITING, it will fail.
@param[out] progress - user can get operation progress from this.
@param[in] boxbasepath - box type e.g. "eFilingboxes"
@param[in] boxnumber - string of 1-20 digits box number
@param[in] foldername - folder name.
@param[in] documentname - serial number from "00000".
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
ProgressRef& progress
CString boxbasepath
CString boxnumber
CString foldername

Operations

CString documentname

Method name: GetBox

Documentation:

```
get Box instance by path
@param[out] box - instance of Box class
@param[in] boxbasepath - box type e.g. "eFilingboxes"
@param[in] boxnumber - string of 1-20 digits box number
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
BoxRef& box
CString boxbasepath
CString boxnumber

Method name: GetBoxList

Documentation:

```
get a list of box
@param[out] list - list of boxes. This list have snapshot of each box
                   instances.
@param[in] boxbasepath - box type e.g. "eFilingboxes"
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
BoxList& list
CString boxbasepath

Method name: GetBoxList

Documentation:

```
get a list of box
@param[out] list - list of boxes. This list have snapshot of each box
                   instances.
@param[in] boxbasepath - box type e.g. "eFilingboxes"
@param[in] from - return list from this value.
@param[in] size - list size, if "from" + "size" is bigger than the
                  number of all boxes, return list size will be smaller
                  than "size".
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
BoxList& list
CString boxbasepath

Operations

unsigned int from
unsigned int size

Method name: GetBoxList

Documentation:

```
Get a list of box. Using this box instance user can able to get only box name and the box number
@param[out] list - list of boxes. This list have snapshot of each box
                    instances.
@param[out] list - total number of boxes
@param[in] boxbasepath - box type e.g. "eFilingboxes"
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString boxbasepath
int totalBoxes
BoxList BoxList

Method name: GetDocument

Documentation:

```
get document instance by resource path
@param[out] doc - instance of Document class
@param[in] boxbasepath - box type e.g. "eFilingboxes"
@param[in] boxnumber - string of 1-20 digits box number
@param[in] foldername - folder name.
@param[in] documentname - serial number from "00000".
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
DocumentRef& doc
CString boxbasepath
CString boxnumber
CString foldername
CString documentname

Method name: GetMailBoxList

Documentation:

```
Get a list of mailbox. Using this box instance user can get box name, the box number, documentType,
userName and comment
@param[out] list - list of mailboxes. This list have snapshot of each mailbox
                   instances.
@param[out] list - total number of mailboxes
@param[in] boxbasepath - box type e.g. "ITUTBoxes"
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:

Operations

Status

Parameters:
BoxList list
int totalBoxes
CString boxbasepath

Method name: GetStatus

Documentation:
```
return the status of BoxDocument
@return INITIALIZED - BoxDocument is available.
        INITIALIZING - BoxDocument is initializing, not available.
        UNINITIALIZED - BoxDocument is uninitialized, not available.
```

Return type:
BoxDocStatus

Parameters:

Method name: Initialize

Documentation:
```
Initialize all Documents.
Deletes all the remaining undeleted ".delete_XXXX" from EFilingBoxes only on first boot-up after
software upgrade to avoid the deletion of Scanned eFiling documents during Version-up. If Document
status is CREATING or DELETING, it is changed to WAITING. WAITING Document is deleted when Cleanup()
is called. If Document status is EDITING or USING, it is changed to READY. Also the method clean up
box clipboard.
Till finishing initialization, BoxDocument::GetStatus returns INITIALIZING.
And BoxDocument returns STATUS_FAILED_INITIALIZATION on the other methods.
@return STATUS_OK on success.
        STATUS_FAILED_INITIALIZATION on INITIALIZING.
```

Return type:
Status

Parameters:

4.3.4. CI::BoxDocument::BoxList

Operations

4.3.5. CI::BoxDocument::Container

```
Container class provides common facilities about box and folder.
```

Operations

Method name: ~Container

Method name: CopyDocument

Operations

Documentation:
```
copy the documents to clipboard
if document status is NOT READY or USING or EDITING, it will fail.
@param[in] docname - source document name
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CString docname

Method name: CopyDocument

Documentation:
```
copy the documents to clipboard
if document status is NOT READY or USING or EDITING, it will fail.
@param[in] documents - source document names
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
std::vector<CString> documents

Method name: CreateArchive

Documentation:
```
create archive
@param[out] archiver - created Archiver object.
@param[in] target - archive file path / file name
@param[in] documentname - document name to be archived
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CI::OperatingEnvironment::Ref<Archiver>& archiver
CString target
CString documentname

Method name: CreateArchive

Documentation:
```
create archive
@param[out] archiver - created Archiver object.
@param[in] target - archive file path / file name
@param[in] documentlist - list of document name to be archived
@return STATUS_OK on success,
```

Operations

```
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CI::OperatingEnvironment::Ref<Archiver>& archiver
CString target
std::vector<CString> documentlist

Method name: CreateFolder

Documentation:

```
create new folder if Folder object call this function, it will fail.
@param[in] foldername - new folder name
@return STATUS_OK on success,
*STATUS_FAILED on failure,
*STATUS_DISK_FULL if there is not enough space on the disk.
STATUS_MAX_ALLOWED_RESOURCES_REACHED if total files reached Max number.
```

Return type:
Status

Parameters:
CString foldername

Method name: CutDocument

Documentation:

```
cut the documents to clipboard
if document status is NOT READY, it will fail.
@param[in] docname - source document name
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CString docname

Method name: CutDocument

Documentation:

```
cut the documents to clipboard
if document status is NOT READY, it will fail.
@param[in] documents - source document names
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:

Operations

std::vector<CString> documents

Method name: DeleteArchive

Documentation:
```
Delete Archive
@param[in] archivepath - path of the archive file that needs to be deleted
@return STATUS_OK on success,
STATUS_FAILED on failure,
```

Return type:
Status

Parameters:
CString archivepath

Method name: DeleteFolder

Documentation:
```
delete the folder
if documents in the box is using, it will fail(some documents are
 deleted).
@param[in] foldername - target folder name
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString foldername

Method name: ExtractArchive

Documentation:
```
 extract archive
 @param[out] extractor - created Extractor object
 @param[in] archiverpath - path of the archive to be extracted
 @param[in] extractorpath - path to which the archived files needs to extracted
 @return STATUS_OK on success,
 STATUS_FAILED on failure,
 STATUS_UNIDENTIFIED_FILE_FORMAT if file format cannot be recognized,
 STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CI::OperatingEnvironment::Ref<Extractor>& extractor
CString archiverpath
CString extractorpath

Method name: GetDocument

Documentation:
```
get document instance in the container.
@param[out] doc - instance of Document class
```

Operations

```
@param[in] documentname - serial number from "00000".
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
DocumentRef& doc
CString documentname

Method name: GetDocumentList

Documentation:

```
get a list of document
@param[out] list - list of documents.
*This list have snapshot of each documents instances.
@return STATUS_OK on success,
*STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
DocumentList& list

Method name: GetDocumentList

Documentation:

```
get a list of document
@param[out] list - list of documents.
This list have snapshot of each documents instances.
@param[in] from - return list from this value. (0-origin)
@param[in] size - list size, if "from" + "size" is bigger than the
number of all documents, return size will be
smaller than "size".
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
DocumentList& list
unsigned int from
unsigned int size

Method name: GetFolder

Documentation:

```
get folder instance by folder name
if Folder object call this function, it will fail.
@param[out] folder - reference to folder instance
@param[in] foldername - target folder name
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Operations

Return type:
Status

Parameters:
FolderRef& folder
CString foldername

Method name: GetFolderList

Documentation:
```
get a list of folder
if Folder object call this function, it will fail.
@param[out] list - list of folders.
this list have snapshot of each folder instances.
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
FolderList& list

Method name: GetFolderList

Documentation:
```
get a list of folder
@param[out] list - list of folders.
this list have snapshot of each folder instances.
@param[in] from - return list from this value. (0-origin)
@param[in] size - list size, if "from" + "size" is bigger than the
number of all lists, return size will be smaller
than "size".
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
FolderList& list
unsigned int from
unsigned int size

Method name: GetName

Documentation:
```
get a name of the container
@param[out] name - a name of the container
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:

Operations

CString& name

Method name: GetSize

Documentation:

```
get size of the containers
It includes the size of all files that compose Document.
@param[out] total - total size (byte) of the container
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
uint64& size

Method name: GetWebDAVProperty

Documentation:

```
get box/folder property
@param[in] key - the property name to be set
@param[out] value - the property value
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString key
CString& value

Method name: PasteDocument

Documentation:

```
paste the document from clipboard
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
STATUS_MAX_ALLOWED_RESOURCES_REACHED if the Resource Limit is reached.
```

Return type:
Status

Parameters:

Method name: PasteDocument(CString &documentname)

Documentation:

```
paste the document from clipboard
@param[out] documentname - name of the document that's been pasted
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
STATUS_MAX_ALLOWED_RESOURCES_REACHED if the Resource Limit is reached.
```

Operations

Return type:
Status

Parameters:
CString &documentname

Method name: SetName

Documentation:
```
set a name of the container
if Folder object call this function, when a folder has the name
already exists, it will fail.
@param[in] name - a name to be set
@return STATUS_OK on success,
STATUS_FAILED on failure.
STATUS_RESOURCE_WITH_SAME_NAME_EXISTS if resource with the name already exists.
```

Return type:
Status

Parameters:
CString name

Method name: SetWebDAVProperty

Documentation:
```
set box/folder property
@param[in] key - the property name to be set
@param[in] value - the property value to be set
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString key
CString value

4.3.6. CI::BoxDocument::CviewBox

Operations

Method name: ~Box

Method name: ~Container

Method name: CopyDocument

Documentation:
```
copy the documents to clipboard
if document status is NOT READY or USING or EDITING, it will fail.
@param[in] docname - source document name
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CString docname

Method name: CopyDocument

Documentation:
```
copy the documents to clipboard
if document status is NOT READY or USING or EDITING, it will fail.
@param[in] documents - source document names
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
std::vector<CString> documents

Method name: CreateArchive

Documentation:
```
create archive
@param[out] archiver - created Archiver object.
@param[in] target - archive file path / file name
@param[in] documentname - document name to be archived
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CI::OperatingEnvironment::Ref<Archiver>& archiver
CString target

Operations

CString documentname

Method name: CreateArchive

Documentation:
```
create archive
@param[out] archiver - created Archiver object.
@param[in] target - archive file path / file name
@param[in] documentlist - list of document name to be archived
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CI::OperatingEnvironment::Ref<Archiver>& archiver
CString target
std::vector<CString> documentlist

Method name: CreateFolder

Documentation:
```
create new folder if Folder object call this function, it will fail.
@param[in] foldername - new folder name
@return STATUS_OK on success,
*STATUS_FAILED on failure,
*STATUS_DISK_FULL if there is not enough space on the disk.
STATUS_MAX_ALLOWED_RESOURCES_REACHED if total files reached Max number.
```

Return type:
Status

Parameters:
CString foldername

Method name: CutDocument

Documentation:
```
cut the documents to clipboard
if document status is NOT READY, it will fail.
@param[in] docname - source document name
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CString docname

Method name: CutDocument

Documentation:

Operations

```
cut the documents to clipboard
if document status is NOT READY, it will fail.
@param[in] documents - source document names
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
std::vector<CString> documents

Method name: DeleteArchive

Documentation:

```
Delete Archive
@param[in] archivepath - path of the archive file that needs to be deleted
@return STATUS_OK on success,
STATUS_FAILED on failure,
```

Return type:
Status

Parameters:
CString archivepath

Method name: DeleteFolder

Documentation:

```
delete the folder
if documents in the box is using, it will fail(some documents are
 deleted).
@param[in] foldername - target folder name
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString foldername

Method name: ExtractArchive

Documentation:

```
 extract archive
 @param[out] extractor - created Extractor object
 @param[in] archiverpath - path of the archive to be extracted
 @param[in] extractorpath - path to which the archived files needs to extracted
 @return STATUS_OK on success,
 STATUS_FAILED on failure,
 STATUS_UNIDENTIFIED_FILE_FORMAT if file format cannot be recognized,
 STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Operations

Parameters:
CI::OperatingEnvironment::Ref<Extractor>& extractor
CString archiverpath
CString extractorpath

Method name: GetDocument

Documentation:
```
get document instance in the container.
@param[out] doc - instance of Document class
@param[in] documentname - serial number from "00000".
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
DocumentRef& doc
CString documentname

Method name: GetDocumentList

Documentation:
```
get a list of document
@param[out] list - list of documents.
*This list have snapshot of each documents instances.
@return STATUS_OK on success,
*STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
DocumentList& list

Method name: GetDocumentList

Documentation:
```
get a list of document
@param[out] list - list of documents.
This list have snapshot of each documents instances.
@param[in] from - return list from this value. (0-origin)
@param[in] size - list size, if "from" + "size" is bigger than the
number of all documents, return size will be
smaller than "size".
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
DocumentList& list
unsigned int from
unsigned int size

Operations

Method name: GetFolder

Documentation:
```
get folder instance by folder name
if Folder object call this function, it will fail.
@param[out] folder - reference to folder instance
@param[in] foldername - target folder name
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
FolderRef& folder
CString foldername

Method name: GetFolderList

Documentation:
```
get a list of folder
if Folder object call this function, it will fail.
@param[out] list - list of folders.
this list have snapshot of each folder instances.
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
FolderList& list

Method name: GetFolderList

Documentation:
```
get a list of folder
@param[out] list - list of folders.
this list have snapshot of each folder instances.
@param[in] from - return list from this value. (0-origin)
@param[in] size - list size, if "from" + "size" is bigger than the
number of all lists, return size will be smaller
than "size".
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
FolderList& list
unsigned int from
unsigned int size

Method name: GetName

Operations

Documentation:
```
get a name of the container
@param[out] name - a name of the container
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& name

Method name: GetNumber

Documentation:
```
get a number of the box
@param[out] number - a number of the box
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& number

Method name: GetSize

Documentation:

```
get size of the containers
It includes the size of all files that compose Document.
@param[out] total - total size (byte) of the container
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
uint64& size

Method name: GetWebDAVProperty

Documentation:
```
get box/folder property
@param[in] key - the property name to be set
@param[out] value - the property value
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString key
CString& value

Operations

Method name: GetWEPDocument

Documentation:

```
 get WEP document ID
  BoxDocument create HDB document and return document ID.
 @param[out] documentID - HDB document ID
 @return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
 @note user need to delete this document using document ID.
```

Return type:
Status

Parameters:
CString& documentID

Method name: PasteDocument

Documentation:

```
paste the document from clipboard
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
STATUS_MAX_ALLOWED_RESOURCES_REACHED if the Resource Limit is reached.
```

Return type:
Status

Parameters:

Method name: SetName

Documentation:

```
set a name of the container
if Folder object call this function, when a folder has the name
already exists, it will fail.
@param[in] name - a name to be set
@return STATUS_OK on success,
STATUS_FAILED on failure.
STATUS_RESOURCE_WITH_SAME_NAME_EXISTS if resource with the name already exists.
```

Return type:
Status

Parameters:
CString name

Method name: SetWebDAVProperty

Operations

Documentation:
```
set box/folder property
@param[in] key - the property name to be set
@param[in] value - the property value to be set
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString key
CString value

Method name: SetWEPDocument

Documentation:
```
set HDB Document of Workflow Execution Parameter to Box.
@param[in] node - WEP node to save
@return STATUS_OK on success,
STATUS_FAILED on failure,
STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
DOM::NodeRef node

4.3.7. CI::BoxDocument::Document

```
Document class provides facilities of document operation.
```

Operations

Method name: AppendPage

Documentation:
```
get new page instance. the new page is added as last page.
images of the page is copied when this method is called.
@param[in] page - instance of Page class
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
PageRef page

Method name: CancelEdit

Documentation:
```
cancel editing delta document
change document status from EDITING to READY. if document status is
NOT EDITING, it will fail.
when this method is called, unlock delta document. all delta
documents will be deleted.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
```

Return type:
Status

Parameters:

Method name: CopyPage

Documentation:
```
copy the page to clipboard
@param[in] pageno - source page number
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
int pageno

Method name: CopyPage

Documentation:
```
copy the pages to clipboard
@param[in] pages - vector of source page numbers
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:

Operations

Status

Parameters:
std::vector<int> pages

Method name: CopyPage

Documentation:
```
copy the page to clipboard
@param[out] progress - user can get operation progress from this.
@param[in] pageno - source page number
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
ProgressRef& progress
int pageno

Method name: CopyPage

Documentation:
```
copy the pages to clipboard
@param[out] progress - user can get operation progress from this.
@param[in] pages - vector of source page numbers
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
ProgressRef& progress
std::vector<int> pages

Method name: CreatePage

Documentation:
```
get new page instance and updates the properties member map in box/* object scope.
this page is not related with any Document. use Document::AppendPage()
after setting page properties.
@param[out] page - new instance of Page class
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
PageRef& page

Method name: CutPage

Operations

Documentation:
```
cut the page to clipboard
if document status is NOT EDITING, it will fail.
@param[in] pageno - source page number
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
int pageno

Method name: CutPage

Documentation:
```
cut the pages to clipboard
if document status is NOT EDITING, it will fail.
@param[in] pages - vector of source page numbers
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
std::vector<int> pages

Method name: CutPage

Documentation:
```
cut the page to clipboard
if document status is NOT EDITING, it will fail.
@param[out] progress - user can get operation progress from this.
@param[in] pageno - source page number
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
ProgressRef& progress
int pageno

Method name: CutPage

Documentation:
```
cut the pages to clipboard
if document status is NOT EDITING, it will fail.
@param[out] progress - user can get operation progress from this.
@param[in] pages - vector of source page numbers
@return STATUS_OK on success,
```

Operations

```
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
ProgressRef& progress
std::vector<int> pages

Method name: DeletePage

Documentation:

```
delete page
@param[in] pageno - delete page number. after deletion, the pages that
                    have greater page number are off to the front.
@param[in] size - delete page size (default = 1)
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
int pageno
int delete_size

Method name: DeletePage

Documentation:

```
delete page
@param[in] pages - vector of delete page numbers. after deletion,
                    the pages that have greater page numbers are off to
                    the front.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
std::vector<int> pages

Method name: DeletePage

Documentation:

```
delete page
@param[out] progress - user can get operation progress from this.
@param[in] pageno - delete page number. after deletion, the pages that
                    have greater page number are off to the front.
@param[in] size - delete page size (default = 1)
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Operations

Parameters:
ProgressRef& progress
int pageno
int delete_size

Method name: DeletePage

Documentation:
```
delete page
@param[out] progress - user can get operation progress from this.
@param[in] pages - vector of delete page numbers. after deletion,
                   the pages that have greater page numbers are off to
                   the front.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
ProgressRef& progress
std::vector<int> pages

Method name: ~Document

Documentation:

Method name: Edit

Documentation:
```
start to edit the document
change document status from READY to EDITING. if document status is
NOT READY, it will fail.
when this method is called, delta document is created. On saving,
delta documents will be overwritten to original documents.
@param[out] editor - instance of DocumentEditor, or NULL
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_USER_EDITING if another user is already editing.
```

Return type:
Status

Parameters:

Method name: EndCreating

Documentation:
```
change status after creating the document
change document status from CREATING to READY. if document status is
NOT CREATING, it will fail.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
```

Return type:
Status

Operations

Parameters:

Method name: EndReserving

Documentation:

```
change status after using the document
change document status from RESERVING to READY & if document status is NOT RESERVING, it will fail.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
```

Return type:
Status

Parameters:

Method name: EndUsing

Documentation:

```
change status after using the document
change document status from USING to READY when any document is NOT
using the document. some documents are still using, the document status
is unchanged. if document status is NOT USING, it will fail.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
```

Return type:
Status

Parameters:

Method name: FasterPastePage

Documentation:

```
paste the page(s) faster from clipboard
if document status is NOT EDITING, it will fail.
@param[in] pageno - page number to insert
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
     */
```

Return type:
Status

Parameters:
int pageno

Method name: FasterPastePage

Documentation:

```
paste the page(s) faster from clipboard
if document status is NOT EDITING, it will fail.
@param[out] progress - user can get operation progress from this.
@param[in] pageno - page number to insert
@return STATUS_OK on success,
         STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Operations

Return type:
Status

Parameters:
ProgressRef & prgress
int pageno

Method name: GetColorModeMap

Documentation:
```
get a map of color mode in Document
@param[out] colormap - a map of color mode in Document
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
ColorModeMap & colormap

Method name: GetHorizontalResolutionMap

Documentation:
```
get a map of horizontal resolution in Document
@param[out] hrmap - a map of horizontal resolution in Document
@return STATUS_OK on success,
       STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
ResolutionMap & hrmap

Method name: GetInputPageCount

Documentation:
```
get input page count of the document.
@param[out] count - input page count of the document. default value will be 0 if no value is set
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
int & count

Method name: GetJobTypeColorSet

Documentation:
```
get a set of jobType and colorMode combination in Document
@param[out] jcset - a set of jobType and colorMode combination
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Operations

Return type:
Status

Parameters:
JobTypeColorSet& jcset

Method name: GetJobTypeMap

Documentation:
```
get a map of job type in Document
@param[out] jobmap - a map of job type in Document
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
JobTypeMap & jobmap

Method name: GetName

Documentation:
```
get a name of the document
@param[out] name - a name of the container
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& name

Method name: GetPage

Documentation:
```
get page instance in the document
@param[in] pageno - page number
@param[out] page - instance of Page class
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters: *int*
pageno
PageRef& page

Method name: GetPageList

Documentation:
```
get an instance of PageList
@param[out] list - list of pages.
                   this list have snapshot of each page instances.
```

Operations

```
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
PageList& list

Method name: GetPageList

Documentation:

```
get an instance of PageList
@param[out] list - list of pages.
                   this list have snapshot of each page instances.
@param[in] from - return list from this value. !! 1-origin !!
@param[in] size - list size, if "from" + "size" is bigger than the
                  number of all pages, return list size will be smaller
                  than "size".
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
PageList& list
unsigned int from
unsigned int size

Method name: GetPaperSizeMap

Documentation:

```
get a map of paper size in Document
@param[out] papermap - a map of paper size in Document
@return STATUS_OK on success,
         STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
PaperSizeMap & papermap

Method name: GetPaperSizeSet

Documentation:

```
get a set of paper size in Document
@param[out] paperset - a set of paper size in Document
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:

Operations

PaperSizeSet& paperset

Method name: GetStatus

Documentation:

```
get status of the document
@param[out] st - current status of the document
@return STATUS_OK on success,
        STATUS_FAILED on failure,
```

Return type:
Status

Parameters:
DocStatus& st

Method name: GetSubsamplingSystemFile

Documentation:

```
get a path of the subsampling system file
@param[out] path - local path of the system file.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& path

Method name: GetSystemFile

Documentation:

```
get a path of the system file
@param[out] path - local path of the system file.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& path

Method name: GetThumbnailSystemFile

Documentation:

```
get a path of the thumbnail system file
@param[out] path - local path of the system file.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:

Operations

CString& path

Method name: GetTotalPage

Documentation:
```
get total page of the document
@param[out] total - total page of the document
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
int& total

Method name: GetTotalSize

Documentation:
```
get total size of the document
@param[out] total - total size of the document
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
uint64& total

Method name: GetVerticalResolutionMap

Documentation:
```
get a map of vertical resolution in Document
@param[out] vrmap - a map of vertical resolution in Document
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
ResolutionMap & vrmap

Method name: GetWebDAVProperty

Documentation:
```
get document property
@param[in] key - the property name to be set
@param[out] value - the property value
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Operations

Parameters:
CString key
CString& value

Method name: GetWEPDocument

Documentation:
```
get WEP Document node from XPath.
@param[out] node - node to be found
@param[in] xpath - the XPath to search for
                          if empty string, return root node.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
@deprecated
```

Return type:
Status

Parameters:
DOM::NodeRef& node
CString xpath

Method name: GetWEPDocument

Documentation:
```
get WEP document ID
 BoxDocument create HDB document and return document ID.
@param[out] documentID - HDB document ID
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
@note user need to delete this document using document ID.
```

Return type:
Status

Parameters:
CString& documentID

Method name: InsertBlankPage

Documentation:
```
insert blank page to the document
the page properties are copied from previous page (if inserting to
first page, the page properties are copied from next one)
@param[in] pageno - new page number of the inserted page(see InsertPage)
@param[in] size - paper size to be inserted
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
int pageno

Operations

PaperSize size

Method name: InsertPage

Documentation:

```
insert page to the document
@param[in] pageno - new page number of the inserted page
                    document[a1, a2, a3] -- InsertPage(2, b1) ->
                    document[a1, b1, a2, a3])
@param[in] page  - a page to be added. If image path as page property
                  is set, the image  will be copied to under the
                  document folder.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters: *int*
pageno
PageRef page

Method name: InsertPage

Documentation:

```
insert pages to the document
@param[in] pageno - page number to be inserted
                          (first page number of the inserted page)
@param[in] pages  - pages to be added. If an image path is set,
                    the image  will be copied to under the document
                    folder.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters: *int*
pageno *PageList*
pages

Method name: PastePage

Documentation:

```
paste the page(s) from clipboard
if document status is NOT EDITING, it will fail.
@param[in] pageno - page number to insert
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:

Operations

int pageno

Method name: PastePage

Documentation:
```
paste the page(s) from clipboard
if document status is NOT EDITING, it will fail.
@param[out] progress - user can get operation progress from this.
@param[in] pageno - page number to insert
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
ProgressRef& progress
int pageno

Method name: PutSubSamplingSystemFile

Documentation:
```
put subsampling system file to the document folder
@param[in] path - the path of system file
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CString path

Method name: PutSystemFile

Documentation:
```
put system file to the document folder
@param[in] path - the path of system file
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CString path

Method name: PutThumbnailSystemFile

Documentation:
```
put thumbnail system file to the document folder
@param[in] path - the path of system file
@return STATUS_OK on success,
```

Operations

```
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CString path

Method name: ReCreate

Documentation:

```
revert the document status back to CREATING
change status from READY to CREATING. if document status is not READY,
it will fail.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
```

Return type:
Status

Parameters:

Method name: ReplacePage

Documentation:

```
replace page in the document
@param[in] pageno - new page number of the replaced page
                     document[a1, a2, a3] -- InsertPage(2, b1) ->
                     document[a1, b1, a3])
@param[in] page  - a page to be added. If image path as page property
                    is set, the image  will be copied to under the
                    document folder.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters: *int*
pageno
PageRef page

Method name: ReplacePage

Documentation:

```
replace pages in the document
@param[in] pageno - page number to be replaced
                          (first page number of the replaced page)
@param[in] pages  - pages to be added. If an image path is set,
                     the image  will be copied to under the document
                     folder.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:

Operations

Status

Parameters: *int*
pageno *PageList*
pages

Method name: Reserve

Documentation:
```
change status for using the document
change document status from READY to RESERVING. if document status is NOT READY, it will fail.
- when Reserve() is called on RESERVING, it will fail as well.
@return STATUS_OK on success,
         STATUS_FAILED on failure,
@note: Reserve as per the requirement should not be used for EFilingBoxes.
  Behavior is undefined, as no specific checks are performed for BoxType.
```

Return type:
Status

Parameters:

Method name: Save

Documentation:
```
save the delta document.
change document status from EDITING to READY. if document status is
NOT EDITING, it will fail.
when this method is called, unlock delta document. delta documents
will be overwritten to original documents. after that all delta
documents will be deleted.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:

Method name: Save

Documentation:
```
save the delta document.
change document status from EDITING to READY. if document status is
NOT EDITING, it will fail.
when this method is called, unlock delta document. delta documents
will be overwritten to original documents. after that all delta
documents will be deleted.
@param[out] progress - user can get operation progress from this.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:

Operations

ProgressRef& progress

Method name: SaveAs

Documentation:

```
save the edited document with other name
change document status from EDITING to READY. if document status is
NOT EDITING, it will fail.
when this method is called, unlock delta document. delta documents
will be overwritten to original documents. after that all delta
documents will be deleted.
@param[in] new_name - merged document save as the name
@param[in] resourcebasepath - save to the this folder, if it is set.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
CString new_name
CString resourcebasepath

Method name: SaveAs

Documentation:

```
save the edited document with other name
change document status from EDITING to READY. if document status is
NOT EDITING, it will fail.
when this method is called, unlock delta document. delta documents
will be overwritten to original documents. after that all delta
documents will be deleted.
@param[out] progress - user can get operation progress from this.
@param[in] new_name - merged document save as the name
@param[in] resourcebasepath - save to the this folder, if it is set.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
ProgressRef& progress
CString new_name
CString resourcebasepath

Method name: SetInputPageCount

Documentation:

```
set input page count of the document
@param[in] count - input page count of the document
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:

Operations

Status

Parameters:
int count

Method name: SetName

Documentation:
```
set a name of the document
When a folder has the name already exists, it will fail.
@param[in] name - a name to be set
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString name

Method name: SetWebDAVProperty

Documentation:
```
set document property
@param[in] key - the property name to be set
@param[in] value - the property value to be set
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString key
CString value

Method name: SetWebDAVProperty

Documentation:
```
set document property
@param[in] key - the property name to be set
@param[in] value - the property value to be set
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString key
int value

Method name: SetWebDAVPropertyFromSystemFile

Documentation:

Operations

```
set properties from system file on memory
some document properties are set as WebDAV properties. Also, some page
properties are set if the main image is copied to the document folder.
@param[in] systemfile - a pointer to System File
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
*void** systemfile

Method name: SetWEPDocument

Documentation:

```
set HDB Document of Workflow Execution Parameter.
@param[in] node - WEP node to save
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there is not enough space on the disk.
```

Return type:
Status

Parameters:
DOM::NodeRef node

Method name: Use

Documentation:

```
change status for using the document
change document status from COMPLETED to USING. if document status is
NOT COMPLETED, it will fail.
@return STATUS_OK on success,
        STATUS_FAILED on failure,
```

Return type:
Status

Parameters:
Method name: GetViewPageList

Documentation:

```
get an instance of PageList. Using these objects user can access the following page properties.
jobType
paperSize
resolution
image size
size
thumbnail
colorMode
creationDate
lastModifiedDate
```

Operations

```
cutPage
@param[out] list - list of pages.
    this list have snapshot of each page instances.
 @ return STATUS_OK on success,
         STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
PageListRef PageList

Method name: GetViewPageList

Documentation:

```
get an instance of PageList. Using these objects user can access the following page properties.
jobType
paperSize
resolution
image size
size
thumbnail
colorMode
creationDate
lastModifiedDate
cutPage
@param[out] list - list of pages.
    this list have snapshot of each page instances.
@param[in] from - return list from this value. !! 1-origin !!
@param[in] size - list size, if "from" + "size" is bigger than the
    number of all pages, return list size will be smaller than "size".
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters: PageListRef PageList

Method name: PutSystemFile

Documentation:

```
put system file to the document folder
@param[in] path - the path of system file
@return STATUS_OK on success,
        STATUS_FAILED on failure,
        STATUS_DISK_FULL if there
        is not enough space on the
        disk.
```

Return type:
Status

Parameters:

Method name: GetFaxPreviewProperty

Operations

Documentation:

```
get fax preview property for document
default fax preview property is "false"
@param[out] value - the property value
@return STATUS_OK on success,
        STATUS_FAILED on failure,
```

Return type:
Status

Parameters:

Method name: SetFaxPreviewProperty
Documentation:

```
set fax preview property for document
default fax preview property is "false"
@param[in] value - the property value
@return STATUS_OK on success,
             STATUS_FAILED on failure.
```

Return type:
Status

Parameters:

Method name: MovePage
Documentation:

```
moves the page from source to destination during Scan Preview
@param[in] srcpageno    - The source page number to be moved from
@param[in] dstpageno    - The destination page number to which the
                          source page has to be moved to
 @return STATUS_OK on success,
       STATUS_FAILED on failure,
       STATUS_OUT_OF_RANGE if the user specified srcpage doesn't exist/
                           if the user specified dstpageno is out of range
Note : MovePage is not supported for XPS & DIB file formats
```

Return type:
Status

Parameters:

Method name: GetRotation
Documentation:

```
get rotation angle of the image file during scan preview
@param[in]  pageno - page number of the page that has to be rotated
@param[out] rangle  - NOROTATE / ROTATE90 / ROTATE180 / ROTATE270
@return STATUS_OK on success
       STATUS_FAILED on failure,
       STATUS_OUT_OF_RANGE if the user specified srcpage doesn't exist
Note : GetRotation return zero degrees if rotationAngle property doesn't exist
```

Return type:
Status

Operations

Parameters:
PageNo, Rangle

Method name: Rotate
Documentation:

```
otates the page by 90 degrees clockwise during Scan preview
@param[in]  pageno    - page number of the page that has to be rotated
return STATUS_OK on success,
@return STATUS_OK on success
       STATUS_FAILED on failure,
       STATUS_OUT_OF_RANGE if the user specified srcpage doesn't exist
Note : Only PDF file formats (PDF or Slim PDF or PDF/A or PDF/A-2) are supported for Rotate page
```

Return type:
Status

Parameters:
PageNo, Rangle

4.3.8. CI::BoxDocument::DocumentList

Operations

4.3.9. CI::BoxDocument::DocumentProperties

```
This class is used to get/set the set of document properties
```

Operations

Method name: GetFromProperty

Documentation:
```
get "from" property of the document
@param[out] from - "from" property of the document
@return STATUS_OK on success,
       STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString from

Method name: GetFromString

Documentation:
```
get fromString of the document
@param[out] fromString - fromString of the document
@return STATUS_OK on success,
       STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString fromString

Method name: GetFromStringFirstName

Documentation:
```
get fromStringFirstName of the document
@param[out] firstName - firstName of the document
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString firstName

Method name: GetFromStringLastName

Documentation:
```
get fromStringLastName of the document
@param[out] lastName - lastName of the document
@return STATUS_OK on success,
       STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString lastName

Operations

Method name: GetName

Documentation:

```
get a name of the document
 @param[out] name - a name of the container
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString name

Method name: GetReceptionNumber

Documentation:

```
get reception number of the document
@param[out] receptionnum - reception number of the document
@return STATUS_OK on success,
      STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString receptionnum

Method name: GetReceptionTime

Documentation:

```
get reception time of the document
@param[out] receptiontime - reception time of the document
@return STATUS_OK on success,
       STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString receptiontime

Method name: GetStatusOfHardCopy

Documentation:

```
get the status for hard copy
@param[out] hardCopy - the status for hard copy.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
@Note - It is available via GetDocumentList
```

Return type:
Status

Parameters:
CString hardCopy

Operations

Method name: GetStatusOfPollingTransmission

Documentation:
```
get the status for polling transmission
@param[out] pollingTransmission - the status for polling transmission
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString pollingTransmission

Method name: GetStatusOfReceivedData

Documentation:
```
get the status for FAX received data.
@param[out] receivedData - the status for FAX received data.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
@Note - It is available via GetDocumentList
```

Return type:
Status

Parameters:
CString receivedData

Method name: GetStatusOfRelayReport

Documentation:
```
get the status for relay report
@param[out] relayReport - the status for relay report.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
@Note - It is available via GetDocumentList
```

Return type:
Status

Parameters:
CString relayReport

Method name: GetTotalPage

Documentation:
```
get total page of the document
@param[out] total - total page of the document
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:

Operations

int total

Method name: SetStatusOfHardCopy

Documentation:

```
set the status for hard copy.
@param[in] hardCopy - the status for hard copy.
@return STATUS_OK on success,
         STATUS_FAILED on failure.
@Note - It is available via GetDocumentList
```

Return type:
Status

Parameters:
CString hardCopy

Method name: SetStatusOfPollingTransmission

Documentation:

```
set the status for polling transmission
@param[in] pollingTransmission - the status for polling transmission
return STATUS_OK on success.
      STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString pollingTransmission

Method name: SetStatusOfReceivedData

Documentation:

```
set the status for FAX received data.
@param[in] receivedData - the status for FAX received data.
@return STATUS_OK on success,
         STATUS_FAILED on failure.
@Note - It is available via GetDocumentList
```

Return type:
Status

Parameters:
CString receivedData

Method name: SetStatusOfRelayReport

Documentation:

```
set the status for relay report.
@param[in] relayReport - the status for relay report
@return STATUS_OK on success,
        STATUS_FAILED on failure.
@Note - It is available via GetDocumentList
```

Return type:
Status

Operations

Parameters:
CString relayReport

4.3.10. CI::BoxDocument::Extractor

Extractor class provides facilities to extract archive.

Operations

Method name: Cancel

Documentation:
```
cancel to extract
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:

Method name: ~Extractor

Method name: GetProgress

Documentation:
```
get extracting progress
@param[out] progress - extracting progress [0-100] %
@return STATUS_OK on success,
STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
int& progress

Method name: GetStatus

Documentation:
```
get extracting status
@param[out] status - extracting status
@return STATUS_OK on success,
STATUS_FAILED on failure.
STATUS_UNIDENTIFIED_FILE_FORMAT if it could not identify the file format.
```

Return type:
Status

Parameters:
ExtractStatus& status

4.3.11. CI::BoxDocument::Folder

```
Folder class provides facilities of folder operation.
```

Operations	
Method name:	~Folder

4.3.12. CI::BoxDocument::FolderList

Operations

4.3.13. CI::BoxDocument::JobTypeColorSet

Operations

4.3.14. CI::BoxDocument::Page

```
Page class provides facilities of page operation.
```

Operations

Method name: GetCopyJpegParameterFilePath

Documentation:

```
get file path of Copy-JPEG parameter
@param[out] path - file path of Copy-JPEG parameter. if fails, empty
                   string "" will return.
@return STATUS_OK on success,
        STATUS_FAILED on failure. e.g. file not found.
```

Return type:
Status

Parameters:
CString& path

Method name: GetFileSize

Documentation:

```
get file size of the page
@param[out] size - size of the page
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
uint64& size

Operations

Method name: GetImage

Documentation:
```
get an image path to the page
@param[out] path - local path of the image.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& path

Method name: GetImageSize

Documentation:
```
get size of the image
@param[out] width - width (pixels) of the page
@param[out] height - height (pixels) of the page
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
int& width
int& height

Method name: GetPaperSize

Documentation:
```
get size of the paper
@param[out] size - the string of paper size
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& size

Method name: GetResolution

Documentation:
```
get resolution of the image
@param[out] horizontal - horizontal resolution of the page
@param[out] vertical - vertical resolution of the page
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Operations

Parameters:
int& horizontal
int& vertical

Method name: GetSubsamplingImage

Documentation:
```
get a subsampling image path to the page
@param[out] path - local path of the image.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& path

Method name: GetThumbnailImage

Documentation:
```
get a thumbnail image path to the page
@param[out] path - local path of the image.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString& path

Method name: GetWebDAVProperty

Documentation:
```
get the page property
@param[in] key - the property name to be set
@param[out] value - the property value
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString key
CString& value

Method name: ~Page

Method name: PutImage

Documentation:
```
put an image to the page
actual image is NOT copied to under the document folder if this function
has NO owner document.
```

Operations

```
@param[in] path - the image file path to be put
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString path

Method name: PutSubsamplingImage

Documentation:

```
put a subsampling image to the page
actual image is NOT copied to under the document folder if this function
has NO owner document.
@param[in] path - the subsampling image file path to be put
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString path

Method name: PutThumbnailImage

Documentation:

```
put a thumbnail image to the page
actual image is NOT copied to under the document folder if this function
has NO owner document.
@param[in] path - the subsampling image file path to be put
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString path

Method name: SetCopyJpegParameter

Documentation:

```
set Copy-JPEG parameter
this parameter is also stored as file when the page is associated with
Document.
@param[in] param - Copy-JPEG parameter to be stored.
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
*void** param

Operations

Method name: SetSubSamplingSystemFile

Documentation:

```
set subsampling system file properties
 the properties are reflected to Document on addition
@param[in] systemfile - a pointer to System File
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
*void** systemfile

Method name: SetSystemFile

Documentation:

```
set system file properties
 the properties are reflected to Document on addition
@param[in] systemfile - a pointer to System File
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
*void** systemfile

Method name: SetThumbnailSystemFile

Documentation:

```
set thumbnail system file properties
 the properties are reflected to Document on addition
@param[in] systemfile - a pointer to System File
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
*void** systemfile

Method name: SetWebDAVProperty

Documentation:

```
set the page property
@param[in] key - the property name to be set
@param[in] value - the property value to be set
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Operations

Parameters:
CString key
CString value

Method name: SetWebDAVProperty

Documentation:
```
set document property
@param[in] key - the property name to be set
@param[in] value - the property value to be set
@return STATUS_OK on success,
        STATUS_FAILED on failure.
```

Return type:
Status

Parameters:
CString key
int value

4.3.15. CI::BoxDocument::PageList

Operations

4.3.16. CI::BoxDocument::PaperSizeSet

Operations

4.3.17. CI::BoxDocument::Progress

```
Progress class provides facilities to get progress.
```

Operations

Method name: GetProgress

Return type:
Status

Parameters:
int& progress

Method name: ~Progress

5. Detailed Design Description View

Scope

The detailed design description contains the internal details of each component. These details include the attribute descriptions for identification, processing, and data.

Use

This description contains the details needed by programmers prior to implementation. The detailed design description can also be used to aid in producing unit test plans.

Representation

Following subsections describe the design of BoxDocument as standard UML2 diagrams.

5.1. THEORY OF OPERATION

BoxDocument is a wrapper on DocumentStore component. It is used to store documents on the WebDAV server in a particular folder structure. The documents could be scanned/faxed/mailed etc. The folder structure comprises of Box, Folder, Document and Pages. Box is the top level which may contain folders/ documents. A folder will contain documents and a document consists of pages. Under document - Image. Subsampling and Thumbnail folders are present which contain images .The images are numbered sequentially corresponding to pages. A page should contain a main image and optionally may have Thumbnail or Subsampling image. The WEP.xml under a document, stands for Workflow Execution Parameter file which contains information regarding the workflow parameters. Each of the Image/Subsampling/Thumbnail folders contains a System Data File (SDF) numbered 00000. This contains all the information regarding the document and its pages like size, color, paperType etc. Certain properties from properties xml and properties dom are set on the each Box/ Folder/Document/Page for the managing them. A document can be in any of the following 6 states - CREATING, READY, EDITING, USING, DELETING and WAITING. It indicates the state of operation of the document. Maximum box limit for EFilingBoxes is 201 and for ITUTBoxes it is 300. Maximum folder limit inside boxes is 100. Maximum document limit for PageLogBoxes, FaxRxPreviewBoxes is 1000 and for ITUTBoxes it is 400. Max page limit for a document is 1000.

The various functionalities provided by BoxDocument are:

1) Create Box/Folder/Document/Page - Creates the resource on the server. Sets the properties for the resource and returns a reference to the instance of Box/Folder/Document/Page.

2) Get Box/Folder/Document/Page - Get the resource on the server. Get the properties set on the resource and return a reference to an instance of Box/Folder/Document/Page

3) Delete Box/Folder/Document/Page - Delete the resource on the server.

4) Scan Preview - Append/Insert/Replace/Delete Pages. Images are uploaded at the position mentioned and the page properties updated accordingly. The SDF too is updated to reflect the new pages. On Delete Page, the corresponding images are deleted .The other pages are sequenced appropriately. SDF too is updated.

5) Archive/Extract Documents - The documents in a box/folder can be archived into a zip file. And the zip file is downloaded to local machine. EFB file format is used to zip a document which is a specialized form of zip .The main zip file consists of the EFB files. The zip file can be extracted to any box/folder. The progress and state of operation of Archival and Extraction are provided to the user.

The specification of Box Archive as follows :

	Create From	Upload To					
eB3	Mash						
	BP						
	Loire						
	Alabama						
eBX	EX-Mash	EX-Mash	Weiss	Weiss2	Weiss3		
	EX-BP	EX-BP	EX-BP2	Shasta	Shasta2		
	EX-Loire	EX-Loire	Baikal	EX-Loire2	Bikal2	Reuss	Reuss2
	EX-Alabama	EX-Alabama	EX-Alabama2	Shastina	Shastina2		
	Baikal	EX-Loire	Baikal	EX-Loire2	Bikal2	Reuss	Reuss2
	Weiss	EX-Mash	Weiss	Weiss2	Weiss3		
	EX-BP2	EX-BP	EX-BP2	Shasta	Shasta2		
	EX-Loire2	EX-Loire	Baikal	EX-Loire2	Bikal2	Reuss	Reuss2
	EX-Alabama2	EX-Alabama	EX-Alabama2	Shastina	Shastina2		
	Baikal2	EX-Loire	Baikal	EX-Loire2	Bikal2	Reuss	Reuss2
	St.Helens						
	Mosel						
eBN	Weiss2	Weiss2	Weiss3				
	Reuss	Reuss	Reuss2				
	Shasta	Shasta	Shasta2				
	Shastina	Shastina	Shastina2				
	Caspian	Caspian					
	Weiss3	Weiss2	Weiss3				
	Reuss2	Reuss	Reuss2				
	Shasta2	Shasta	Shasta2				
	Shastina2	Shastina	Shastina2				

From the table, e.g. Weiss2 should support the box archive of EX-Mash, Weiss and Weiss2.
St.Helens and Mosel doesn't support TopAccess::e-Filing itself.

6) Edit document - Clipboard is in /imagedata for Weiss2 and in /work for other models, following are the Clipboard operations:
a) Cut/Copy/Paste documents b) Cut/Copy/Paste pages
c) Edit/Save/Save As/CancelEdit operations on document
To Cut/Paste pages, the document has to be in Edit state. Upon Cut/Copy/Paste of page, the document can be either saved or cancelled.

Another feature is Box Initialization, which initializes all the boxes. Any document in
1) CREATING/DELETING state is deleted
2) EDITING/USING state is brought to READY
state upon initialization and cleanup.

The following error codes are returned on special cases:
1) STATUS_DISK_FULL - There is no more space on the hard disk.
2) STATUS_USER_EDITING - A user is already editing the document.
3) STATUS_RESOURCE_WITH_SAME_NAME_EXISTS - Specified resource with the name already exists.
4) STATUS_ARCHIVE_SIZE_ERROR - Archive is not possible if the archive size is more than 2GB.
5) STATUS_UNIDENTIFIED_FILE_FORMAT - Cannot Upload an archive of file format other than zip.
6) STATUS_DEPENDENT_RESOURCE_LOCKED - Specified resource or one of its child is currently locked by another user.
7) STATUS_MAX_ALLOWED_RESOURCES_REACHED - Maximum permissible EFiling resources have been reached.

Note: All the special characters and multi-byte characters supported in MFP Panel should be checked and handled when passed as input parameters to Box Document interface calls.

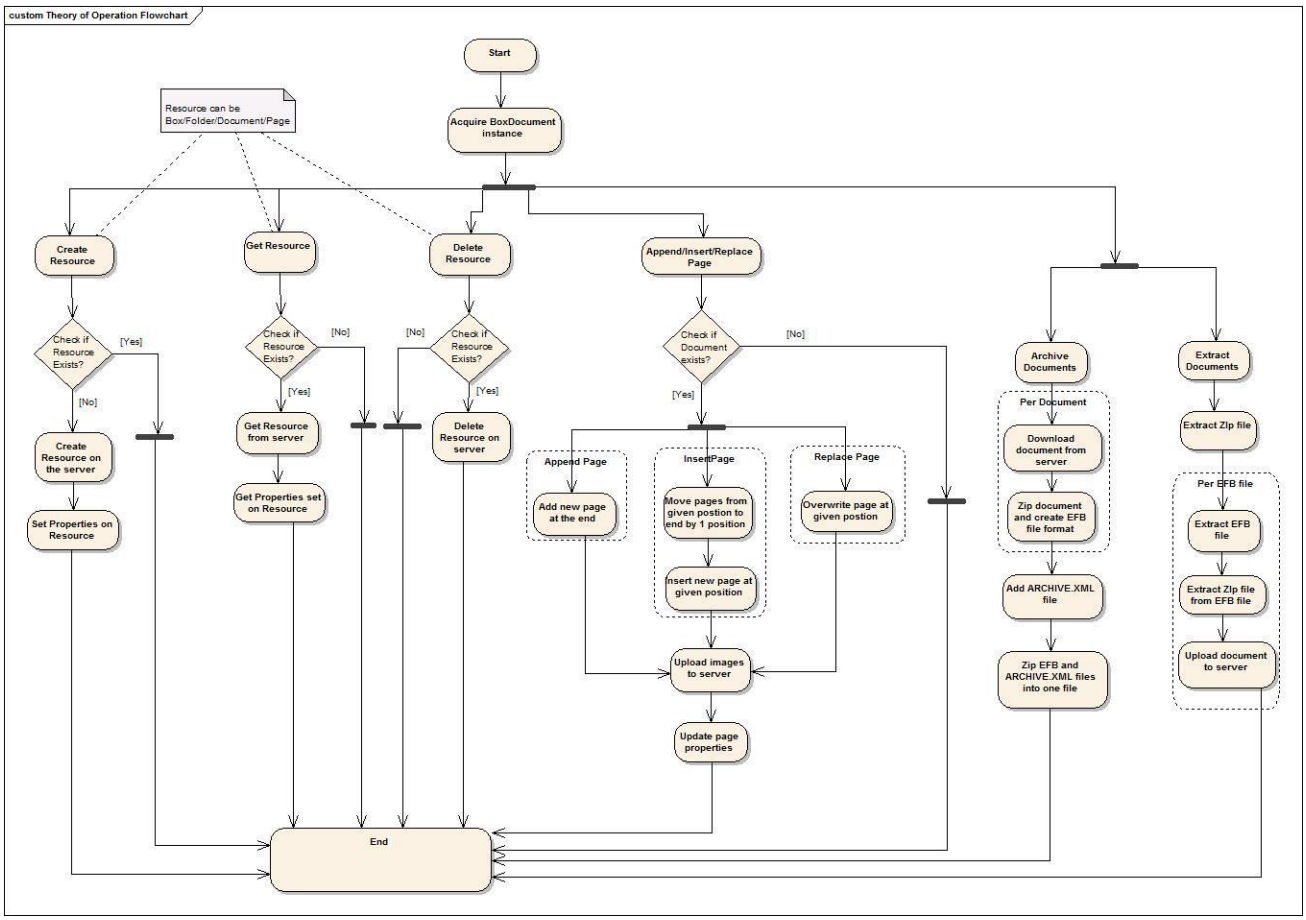


Figure 2: Theory of Operation Flowchart

5.2. CLASS MODEL

Class Diagrams

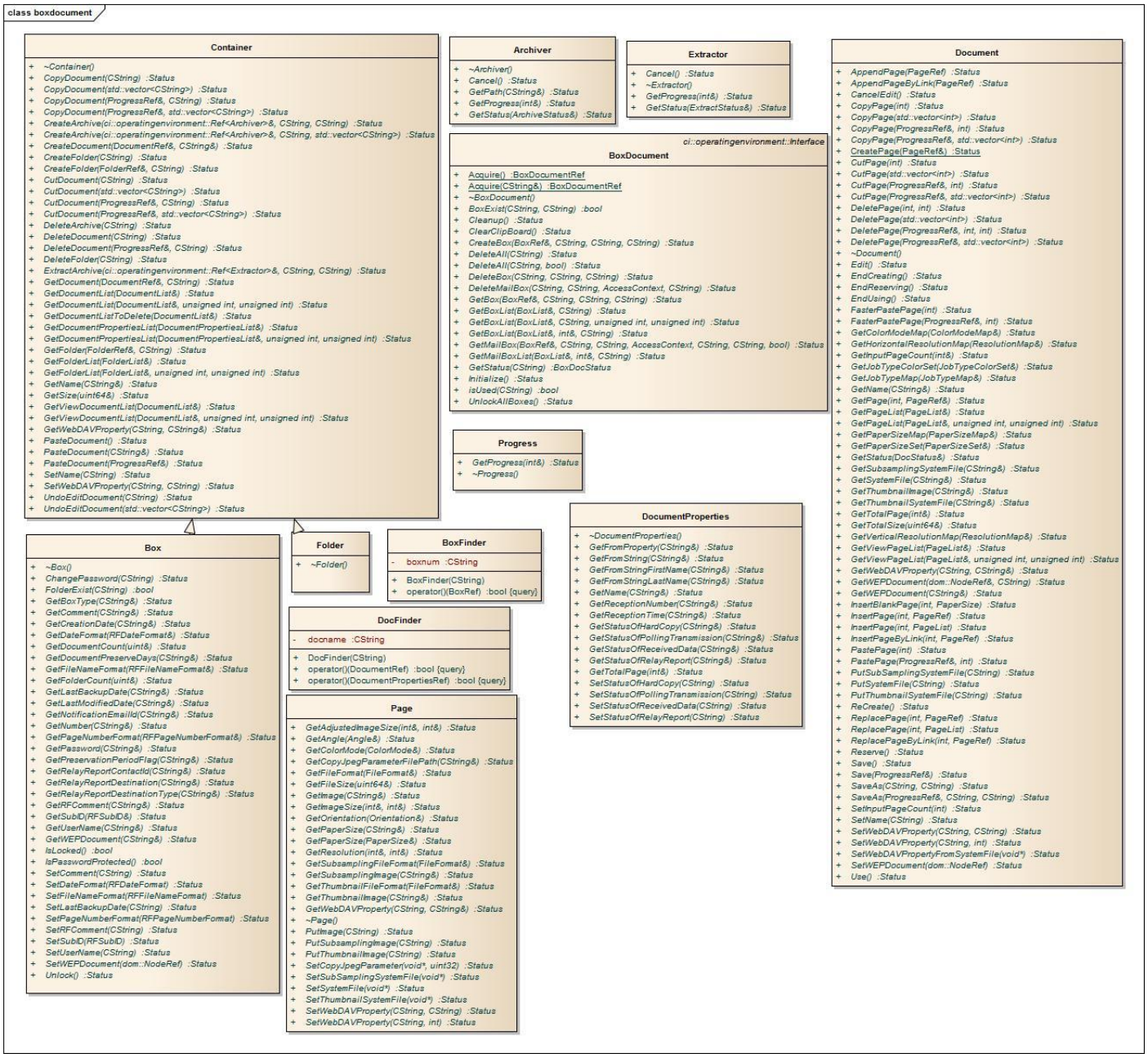


Figure 3 : BoxDocument

5.3. COLLABORATION MODEL

State chart Diagrams

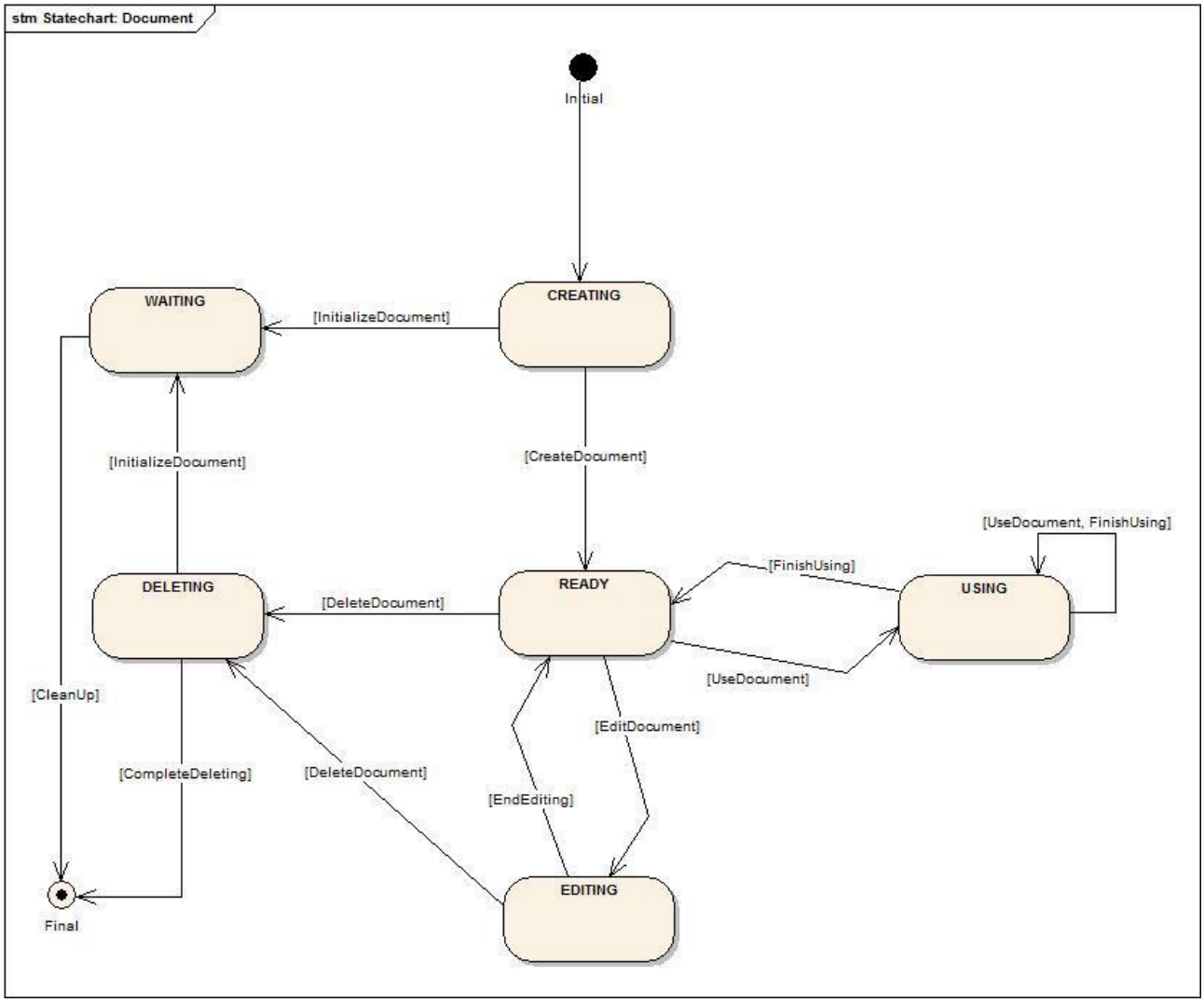


Figure 4: State Chart :Document

State transition of document

Sequence Diagrams

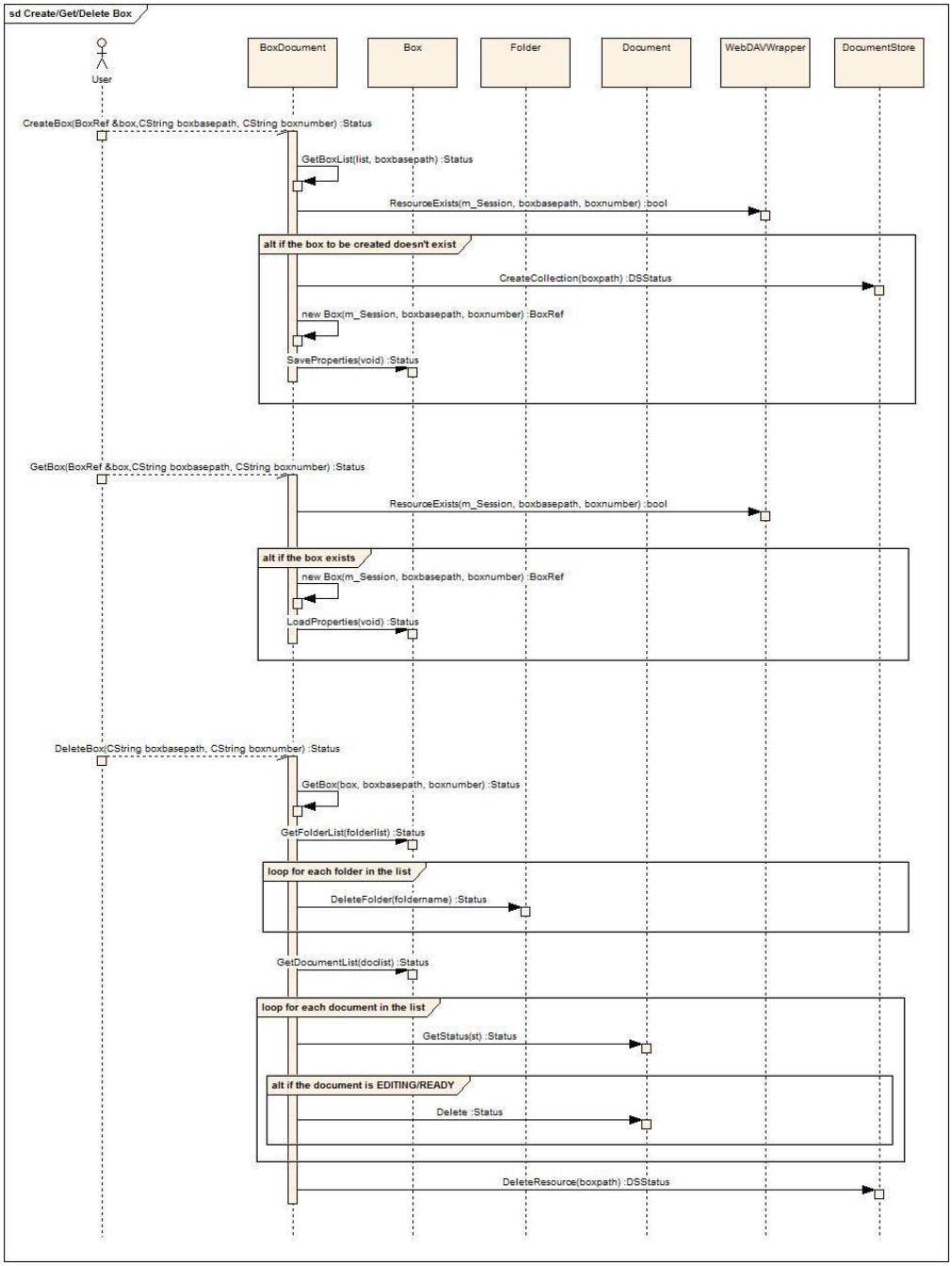


Figure 5: Create/Get/Delete Box

Sequence to Create or get or delete box. User need BoxDocument instance to perform these operations

Message documentation

CreateBox

Message documentation

User ----> BoxDocument

User requests to create new boxnumber resource.

GetBoxList

BoxDocument ----> **BoxDocument**

Get the list of existing boxes.

ResourceExists

BoxDocument ----> **WebDAVWrapper**

To check if the box to be created already exists.

CreateCollection

BoxDocument ----> **DocumentStore**

Create new box. Use DocumentStore to create collection.

new Box

BoxDocument ----> **BoxDocument**

Acquire the new box instance.

SaveProperties

BoxDocument ----> **Box**

Update and save all the WebDAV properties on to the box.

GetBox

User ----> BoxDocument

User request to get the box instance of existing box number resource.

ResourceExists

BoxDocument ----> **WebDAVWrapper**

Check if the box exists.
new Box

Message documentation

BoxDocument ----> **BoxDocument**

Create a new box instance.

LoadProperties

BoxDocument ----> **Box**

Load all the WebDAV properties to the box member variables.

DeleteBox

User ----> BoxDocument

User request BoxDocument to delete the box by box number.

GetBox

BoxDocument ----> **BoxDocument**

Get the box instance of the boxnumber resource.

GetFolderList

BoxDocument ----> **Box**

Get all the folders in the box.

DeleteFolder

BoxDocument ----> **Folder**

Delete the folder recursively. i.e., delete all the documents under it and the folder itself.

GetDocumentList

BoxDocument ----> **Box**

Get all the documents in the box (directly under the box).

GetStatus

BoxDocument ----> **Document**

Get the status of the document

Delete

Message documentation

BoxDocument ----> **Document**

Delete the documents and its contents if the document is EDITING/READY.

DeleteResource

BoxDocument ----> **DocumentStore**

Once the box contents are recursively deleted, delete the box resource itself.

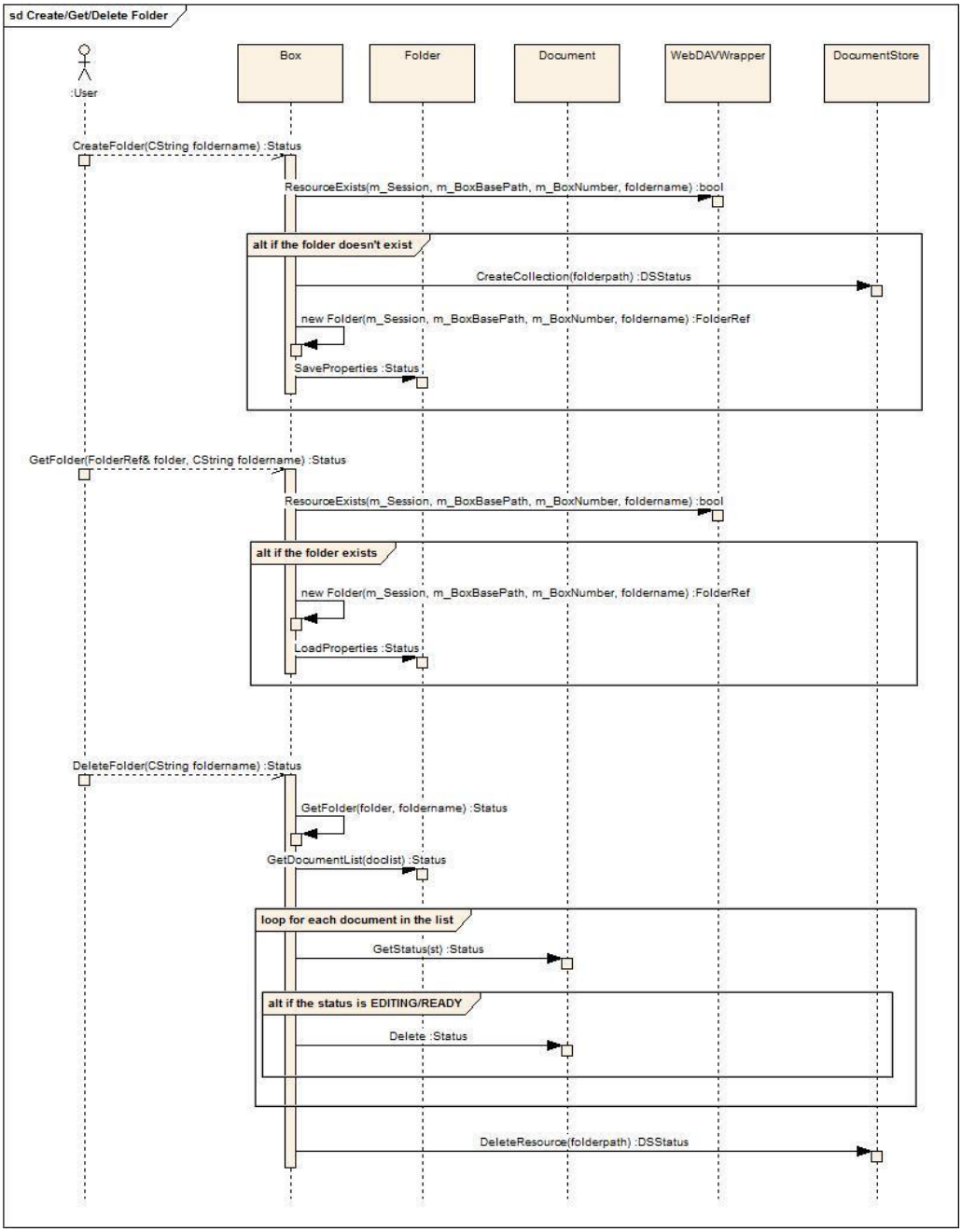


Figure 6: Create/Get/Delete Folder

Sequence to create or Get or Delete a folder. User will have Box instance to perform these operations.

Message documentation

CreateFolder

----> **Box**

User request box document to create a folder under a particular box.

Message documentation

ResourceExists

Box ----> WebDAVWrapper

Check if the folder name resource already exists under the box.

CreateCollection

Box ----> DocumentStore

Use DocumentStore to create the folder resource in the box.

new Folder

Box ----> Box

Acquire a new Folder instance.

SaveProperties

Box ----> Folder

Update the member variable to contain all the WebDAVProperties.

GetFolder

----> **Box**

User request box document to get folder instance in the box.

ResourceExists

Box ----> WebDAVWrapper

To check if the folder name resource exists in the box.

new Folder

Box ----> Box

Get a new instance of Folder

LoadProperties

Box ----> Folder

Load all the WebDAV properties to the member variables.

Message documentation

DeleteFolder

----> **Box**

User request BoxDocument to delete a folder under the box.

GetFolder

Box ----> Box

Get the folder instance of the foldername resource to be deleted.

GetDocumentList

Box ----> Folder

Get all the documents inside the folder.

GetStatus

Box ----> Document

Get the status of the document

Delete

Box ----> Document

Delete the document and its contents if the document is EDITING/READY.

DeleteResource

Box ----> DocumentStore

Once all the documents inside the folder are recursively deleted, delete the foldername resource itself.

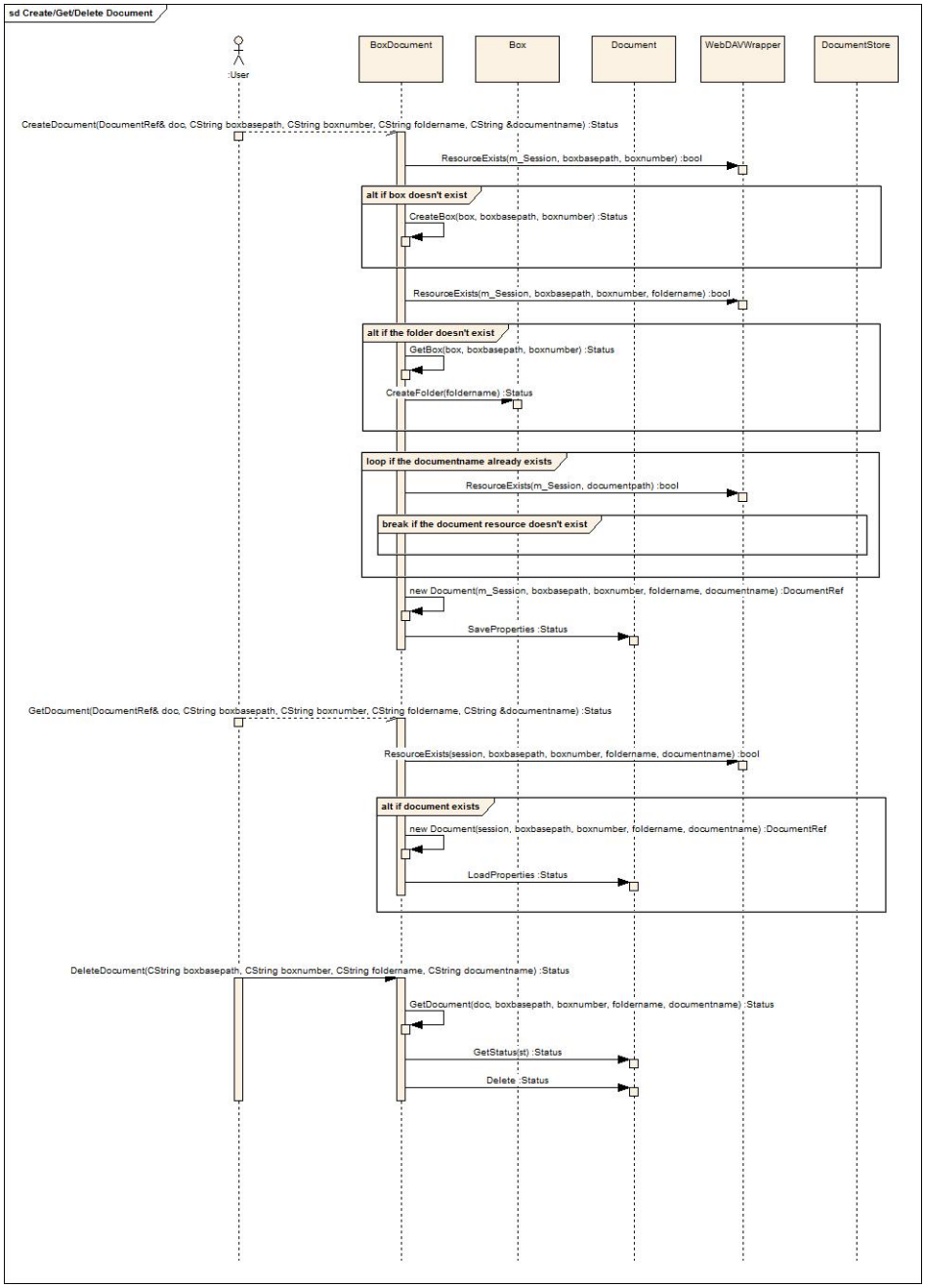


Figure 7: Create/Get/Delete Document

Sequence to Create or get or delete the documents. User can perform these operations on box/folder.

Message documentation

CreateDocument

----> **BoxDocument**

User requests BoxDocument to create the document by document name

Message documentation

ResourceExists

BoxDocument ----> **WebDAVWrapper**

Check if the boxnumber resource exists.

CreateBox

BoxDocument ----> **BoxDocument**

Create box if it doesn't exist.

ResourceExists

BoxDocument ----> **WebDAVWrapper**

Check if the folder exists

GetBox

BoxDocument ----> **BoxDocument**

Get box instance if null

CreateFolder

BoxDocument ----> **Box**

Create folder inside the box.

ResourceExists

BoxDocument ----> **WebDAVWrapper**

Check if the document to be created already exists. If it does, then create document as follows: ex: if documentname = EBX and it exists then, try to create document by name EBX-XXX, where XXX is an unsigned integer starting from 000.

new Document

BoxDocument ----> **BoxDocument**

Get new document instance of the document created.

SaveProperties

BoxDocument ----> **Document**

Message documentation

Save all the WebDAV properties on to the documentname resource.

GetDocument

----> **BoxDocument**

User requests box document to get the document instance of existing document. If the status of the document is DELETING, it fails.

ResourceExists

BoxDocument ----> **WebDAVWrapper**

Check if the document already exists. Else return failure.

---

new Document

BoxDocument ----> **BoxDocument**

Get a NEW document instance.

LoadProperties

BoxDocument ----> **Document**

Load all the WebDAV properties to the member variables.

DeleteDocument

----> **BoxDocument**

User requests box document to delete the existing document. Status of the document will be changed to DELETING before it is deleted.

GetDocument

BoxDocument ----> **BoxDocument**

Get the document instance by documentname.

GetStatus

BoxDocument ----> **Document**

Get the status of the document. Cannot delete if the document is not in either READY/EDITING state.

Delete

Message documentation

BoxDocument ----> Document

Delete the document resource and its contents.

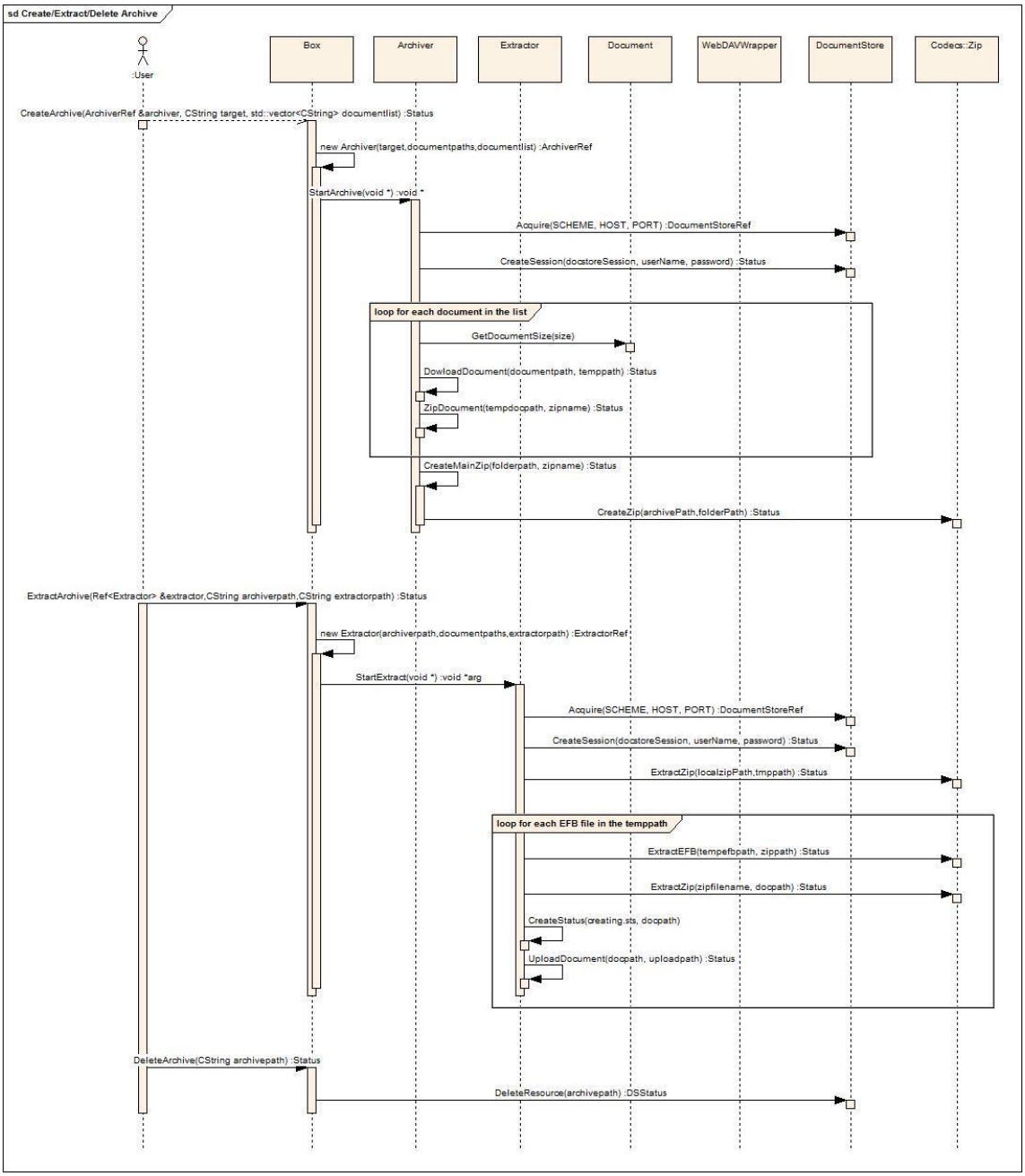


Figure 8: Create/Extract/Delete Archive

Sequence for Create/Extract/Delete Archive. These operations are performed on a box instance.

Message documentation

CreateArchive

----> **Box**

User requests BoxDocument to create archive of list of documents in the box.

new Archiver

Box ----> Box

Get new instance of archiver.

StartArchive

Box ----> Archiver

Archives the given document list used as a thread routine

Acquire

Archiver ----> DocumentStore

Acquire DocumentStore instance.

CreateSession

Archiver ----> DocumentStore

Create a new WebDAV session.

GetDocumentSize

Archiver ----> Document

Get the total size of the document including SDFs & WEP file.

DowloadDocument

Archiver ----> Archiver

Download the document to local directory.

ZipDocument

Archiver ----> Archiver

Zip the document and convert to EFB format. Use Codecs::Zip to archive the same.

Message documentation

CreateMainZip

Archiver ----> Archiver

Create a zip containing all the EFB files along with archive.xml file.

CreateZip

Archiver ----> Codecs::Zip

Create zip of all the files in the folder.

ExtractArchive

----> **Box**

User request BoxDocument to extract archive

---

new Extractor

Box ----> Box

Get new instance of extractor to perform extract operation.

StartExtract

Box ----> Extractor

Extracts the given archive used as a thread routine

Acquire

Extractor ----> **DocumentStore**

Acquire DocumentStore instance

CreateSession

Extractor ----> DocumentStore

Create document store session to upload the extracted documents.

ExtractZip

Extractor ----> Codecs::Zip

Extract the zip file to a local directory

Message documentation

ExtractEFB

Extractor ----> Codecs::Zip

Extract and convert EFB to Zip format.

ExtractZip

Extractor ----> Codecs::Zip

Extract zip file to a directory.

CreateStatus

Extractor ----> Extractor

Create creating.sts file under the docpath

UploadDocument

Extractor ----> Extractor

Upload the contents of the document and the document itself to uploadpath on the server using DocumentStore.

DeleteArchive

----> **Box**

Delete Archive from the server using document store.

DeleteResource

Box ----> DocumentStore

Delete resource on the server.

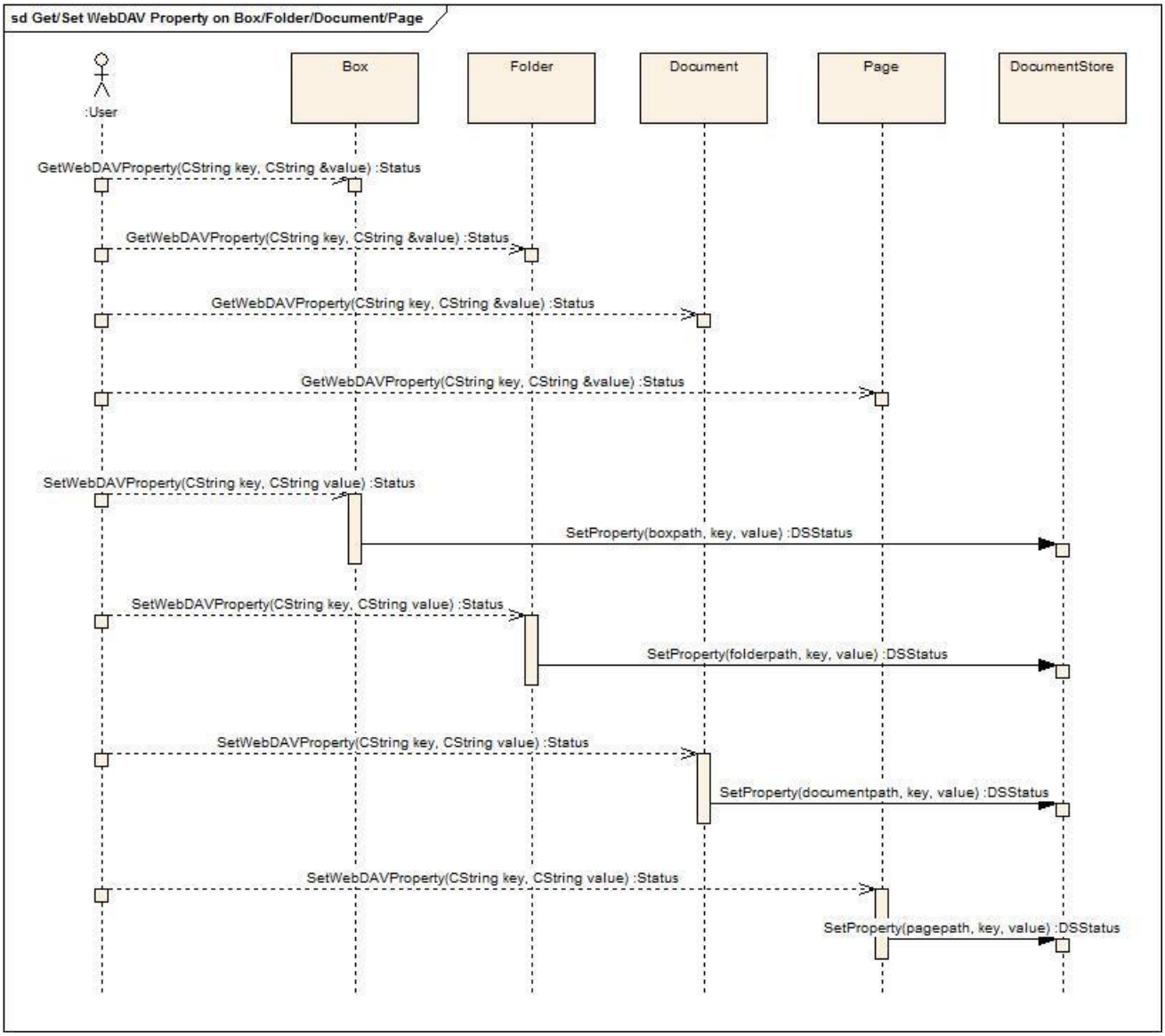


Figure 9: Get/Set WebDAV Property on Box/Folder/Document/Page

Sequence to get/set WebDAV property on the resource. Resource can be a box/folder/ document/page, user needs to have instance of corresponding resource to perform get/set operation.

Message documentation

GetWebDAVProperty

----> **Box**

User uses BoxDocument to get the WebDAV property on the box.

GetWebDAVProperty

----> **Folder**

Message documentation

User uses BoxDocument to get the WebDAV property of the folder.

GetWebDAVProperty

----> **Document**

User uses BoxDocument to get the WebDAV property of the document.

GetWebDAVProperty

----> **Page**

User uses BoxDocument to get the WebDAV property of a page in the document.

SetWebDAVProperty

----> **Box**

User uses BoxDocument to set the WebDAV property on the box.

SetProperty

Box ----> DocumentStore

SetProperty in DocumentStore is called to set a WebDAV property on the resource.

SetWebDAVProperty

----> **Folder**

User uses BoxDocument to set a WebDAV property on the folder resource.

SetProperty

Folder ----> DocumentStore

SetProperty is called to set a WebDAV property on the resource.

SetWebDAVProperty

----> **Document**

User uses BoxDocument to set a WebDAV property on the document resource.

SetProperty

Document ----> DocumentStor

Message documentation

SetProperty is called to set the WebDAV property on the resource.

SetWebDAVProperty

----> **Page**

User uses BoxDocument to set a WebDAV property on a page inside a document.

SetProperty

Page ----> DocumentStore

SetProperty is called to set a property on the WebDAV resource.

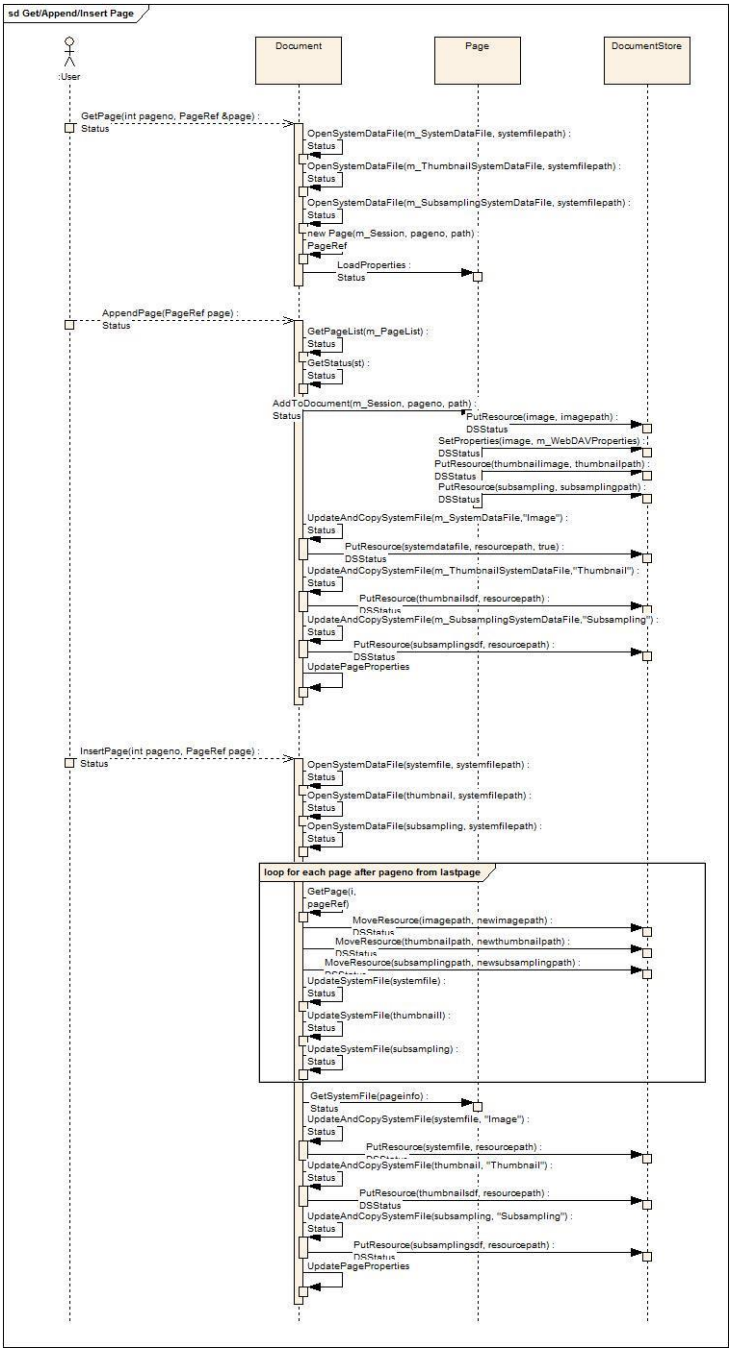


Figure 10: Get/Append/Insert Page

Sequence for Get/Append/Insert Page. These operations can only be performed on a document.

Message documentation

GetPage

----> **Document**
User requests BoxDocument to get a page in the document by page number.

Message documentation

OpenSystemDataFile

Document ----> Document

Open Image/00000 SystemDataFile. This file contains all the page related information. Page number, pagename, page size, pagetype etc properties of each page in the document exist in this.

OpenSystemDataFile

Document ----> Document

Open Thumbnail/00000 SystemDataFile if exists

OpenSystemDataFile

Document ----> Document

Open Subsampling/00000 system data file if it exists.

new Page

Document ----> Document

Get new page instance.

LoadProperties

Document ----> Page

Load all the WebDAV related properties of the page to member variables.

AppendPage

----> **Document**

Get new page instance. The new page is added as last page. Images of the page is copied when this method is called. AppendPage works only if the document is either in EDITING/CREATING status.

GetPageList

Document ----> Document

Get the instances of the existing pages into list.

GetStatus

Document ----> **Document**

Message documentation

Check the status of the document. If it is not EDITING/CREATING, append page fails.

AddToDocument

Document ----> Page

Put the resource to the document under Image/Thumbnail/Subsampling.

PutResource

Page ----> DocumentStore

Put the resource to the imagepath on the server.

SetProperties

Page ----> DocumentStore

set all the WebDAV properties of the image.

PutResource

Page ----> DocumentStore

put the thumbnail image to the server if required.

PutResource

Page ----> DocumentStore

Put the subsampling image on to the server if required.

UpdateAndCopySystemFile

Document ----> Document

Update the systemfile with the pagenumbers and other page properties of the page appended and copy the same to the document on the server.

PutResource

Document ----> DocumentStore

Overwrite the SDF with the updated one.

UpdateAndCopySystemFile

Message documentation

Document ----> **Document**

~~Update the systemfile with the pagenumbers and other page properties of the page appended and copy the~~ same to the document on the server.

PutResource

Document ----> DocumentStore

Overwrite Thumbnail SDF with updated one.

UpdateAndCopySystemFile

Document ----> Document

~~Update the systemfile with the pagenumbers and other page properties of the page appended and copy the~~ same to the document on the server.

PutResource

Document ----> DocumentStore

Overwrite Subsampling SDF with updated one.

UpdatePageProperties

Document ----> Document

~~Update all the page related properties and required WebDAVProperties of the appended page and the current~~ document.

InsertPage

----> **Document**

~~User request BoxDocument to insert a page at some position in the document. New page number of the~~ inserted page: document [a1, a2, a3] -- InsertPage (2, b1) ->document [a1, b1, a2, a3])

OpenSystemDataFile

Document ----> Document

~~Open Image/00000 SystemDataFile. This file contains all the page related information. Page number,~~ pagename, page size, pagetype etc properties of each page in the document exist in this.

OpenSystemDataFile

Document ----> Document

Message documentation

Open Thumbnail SDF if it exists.

OpenSystemDataFile

Document ----> Document

Open subsampling SDF if it exits.

GetPage

Document ----> Document

Get the page and its details. Slide the page to next. i.e., i +1 position.

MoveResource

Document ----> DocumentStore

Move the resource from current name to new name. i.e., if the image name is 003, new image name will be 004. This is done to sequence the pages after insert page.

MoveResource

Document ----> DocumentStore

Move the thumbnail resources if exists, like images for sequencing.

MoveResource

Document ----> DocumentStore

Move the subsampling images like Images/Thumbnails.

UpdateSystemFile

Document ----> Document

Update the systemfile with the pagenumbers and other page properties each page moved.

UpdateSystemFile

Document ----> Document

Update the systemfile with the pagenumbers and other page properties of the page moved.

UpdateSystemFile

Message documentation

Document ----> Document

Update the systemfile with the pagenumbers and other page properties of the page moved.

GetSystemFile

Document ----> Page

Get the system file information of the page to be inserted. It will contain all the page related information like type, size etc.

UpdateAndCopySystemFile

Document ----> Document

Update the systemfile with the pagenumber and other page properties of the page inserted and copy the same to the document on the server.

PutResource

Document ----> DocumentStore

Overwrite Image SDF with the updated one.

UpdateAndCopySystemFile

Document ----> Document

Update the systemfile with the pagenumber and other page properties of the page inserted and copy the same to the document on the server.

PutResource

Document ----> DocumentStore

Overwrite the Thumbnail SDF with the updated one.

UpdateAndCopySystemFile

Document ----> Document

Update the systemfile with the pagenumbers and other page properties of the page inserted and copy the same to the document on the server.

PutResource

Document ----> DocumentStore

Message documentation

Update Subsampling SDF with the latest one.

UpdatePageProperties

Document ----> Document

Update all the page related properties and required WebDAVProperties of the inserted page and the current document.

Figure 11: Replace/Delete Page

Message documentation

ReplacePage

----> **Document**

Replace page in the document. New page number of the replaced page: document [a1, a2, a3] -- InsertPage (2, b1) -> document [a1, b1, a3])

GetPageList

Document ----> Document

Get the list of pages in the document.

OpenSystemDataFile

Document ----> Document

Open Image/00000 SystemDataFile. This file contains all the page related information. Page number, pagename, page size, pagetype etc properties of each page in the document exist in this.

OpenSystemDataFile Document

----> Document Open Thumbnail

SystemDataFile.

OpenSystemDataFile

Document ----> Document

Open Subsampling SystemDataFile.

PutResource

Document ----> DocumentStore

Put resource is used to just overwrite the image existing in the server currently to accomplish replacing of the page in the document.

PutResource

Document ----> DocumentStore

Overwrite the thumbnail image with the new thumbnail image.

PutResource

Message documentation

Document ----> DocumentStore

Overwrite the subsampling image if it exists.

UpdateAndCopySystemDataFile

Document ----> Document

Update the systemfile with the pagenumbers and other page properties of the page replaced and copy the same to the document on the server.

PutResource

Document ----> DocumentStore

Overwrite Image SDF with the updated one.

UpdateAndCopySystemDataFile

Document ----> Document

Update the systemfile with the pagenumbers and other page properties of the page replaced and copy the same to the document on the server.

PutResource

Document ----> DocumentStore

Overwrite the thumbnail SDF with the updated one.

UpdateAndCopySystemDataFile

Document ----> Document

Update the systemfile with the pagenumbers and other page properties of the page replaced and copy the same to the document on the server.

PutResource

Document ----> DocumentStore

Overwrite Subsampling SDF with the updated one.

UpdatePageProperties

Document ----> Document

Message documentation

Update all the page properties (WebDAV) of the new page added. Also update Document properties (WebDAV) To reflect total size, total pages etc.

DeletePage

----> **Document**

Delete pages in the list. After deletion, the pages that have greater page number are off to the front. The pages can be deleted only if document is in CREATING/EDITING status.

DeleteSinglePage

Document ----> Document

Delete each page in the list and sequence pages after deletion.

OpenSystemDataFile

Document ----> Document

Open Image SDF.

OpenSystemDataFile

Document ----> Document

Open Thumbnail SDF.

OpenSystemDataFile

Document ----> Document

Open Subsampling SDF.

GetPage

Document ----> Document

Get the page instance of the page that should be deleted.

DeleteResource

Document ----> DocumentStore

Delete the Image/<pageno> resource from the document on the server.

DeleteResource

Message documentation

Document ----> DocumentStore

Delete the Thumbnail/<pageno><ext> resource from the document on the server. Ext may be optional.

DeleteResource

Document ----> DocumentStore

Delete the Subsampling/<pageno> resource from the document on the server.

MoveResource

Document ----> DocumentStore

To sequence the pages after deletion, move the current page to current page -1 position. i.e., if the current page is 004, then new page name will be 003.

MoveResource

Document ----> DocumentStore

Move the thumbnail data to accomplish sequencing of pages.

MoveResource

Document ----> DocumentStore

Move the subsampling data to accomplish sequencing.

UpdateSystemFile

Document ----> Document

Update the systemfile with the pagenumbers and other page properties each page moved.

UpdateSystemFile

Document ----> Document

Update the systemfile with the pagenumbers and other page properties each page moved.

UpdateSystemFile

Document ----> Document

Update the systemfile with the pagenumbers and other page properties each page moved.

Message documentation

UpdateAndCopySystemDataFile

Document ----> Document

Update the Image SDF data after sequencing and copy the same to document on the server.

PutResource

Document ----> DocumentStore

Overwrite the Image SDF with the updated one.

UpdateAndCopySystemDataFile

Document ----> Document

Update the SDF after sequencing and copy the same to the document on the server.

PutResource

Document ----> DocumentStore

Overwrite the Thumbnail SDF with the updated one.

UpdateAndCopySystemDataFile

Document ----> Document

Update the SDF after sequencing and copy the same to the document on the server.

PutResource

Document ----> DocumentStore

Overwrite the Subsampling SDF with the updated one.

UpdatePageProperties

Document ----> Document

Update Document properties (WebDAV) to reflect total size, total pages etc.

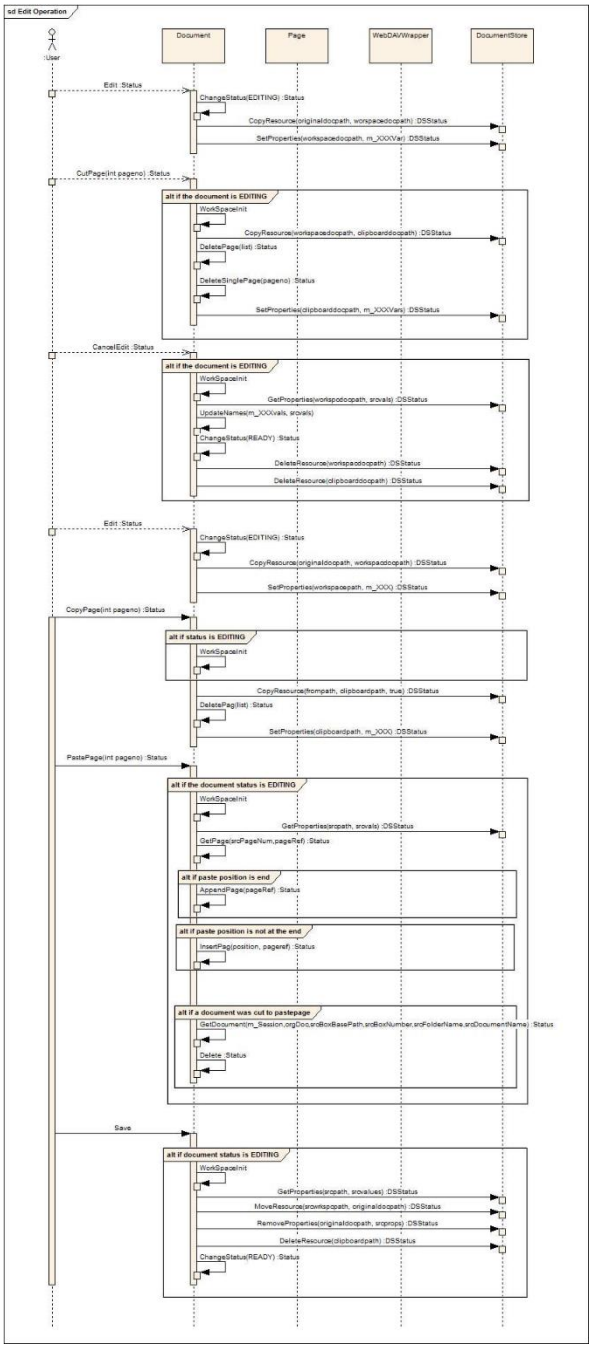


Figure 12: Edit Operation

Sequence for Edit Operation. Edit operations can only be performed on the documents. Cut/ Copy/Paste can be performed on the pages and Edit/Save/CancelEdit on document.

Message documentation
Edit
-→ **Document**

Message documentation

Start to edit the document by changing document status from READY to EDITING. If document status is not READY, it will fail. When this method is called, delta document is created. On saving, delta documents will be overwritten to original documents.

ChangeStatus

Document ----> Document

Change the status of the document to EDITING.

CopyResource

Document ----> DocumentStore

Copy the Original resource to the WorkSpace (i.e. in /imagedata for Weiss2 and /work for other models). Any changes to the document (Cut/Paste etc) will be performed on this copy and will only be saved to original document if the user explicitly calls save.

SetProperties

Document ----> DocumentStore

Set the Original BoxBasePath, BoxNumber, FolderName and DocumentNames as WebDAV properties to the document in Workspace (i.e. in /imagedata for Weiss2 and /work for other models) for reference.

CutPage

----> **Document**

User selects the page to be cut and calls CutPage with the pageno.

WorkSpaceInit

Document ----> Document

Update the m_XXX variables, so that any further changes happen on the delta document and not the original document.

CopyResource

Document ----> DocumentStore

Copy the document from workspace docpath (i.e. in /imagedata for Weiss2 and /work for other models) to clipboard Docpatb.

DeletePage

Message documentation

Document ----> **Document**

Delete all the pages from the clipboard document except the ones that are cut by the user.

DeleteSinglePage

Document ----> Document

Delete the page from the workspace path (i.e. in /imagedata for Weiss2 and /work for other models), the page that needs to be cut.

SetProperties

Document ----> DocumentStore

Set the WorkSpace (i.e. in /imagedata for Weiss2 and /work for other models) BoxBasePath, BoxNumber, FolderName and DocumentNames as WebDAV properties to the document in ClipBoard for reference.

CancelEdit

----> **Document**

User request to cancel editing delta document. Change document status from EDITING to READY. If document status is not EDITING, it will fail. When this method is called, unlock delta document. All delta documents will be deleted.

WorkSpaceInit

Document ----> Document

Update the m_XXX variables, so that any further changes happen on the delta document and not the original document.

GetProperties

Document ----> DocumentStore

Get the properties like m_XXX values set previously. i.e., srcBoxBasePath, srcBoxNumber, srcFolderName, srcDocumentName

---

UpdateNames

Document ----> Document

Update the m_BoxBasePath, m_BoxNumber, m_FolderName, m_DocumentName members with

srcBoxBasePath, srcBoxNumber, srcFolderName, and srcDocumentName respectively**.**

Message documentation

ChangeStatus

Document ----> Document

Change the status of the document to READY.

DeleteResource

Document ----> DocumentStore

Delete the WorkSpace (i.e. in /imagedata for Weiss2 and /work for other models) document completely. This is part of clean-up activity.

DeleteResource

Document ----> DocumentStore

Delete any clipboard document created for this document and the session. This is part of clean-up activity.

Edit

----> **Document**

Start to edit the document by changing document status from READY to EDITING. If document status is not READY, it will fail. When this method is called, delta document is created. On saving, delta documents will be overwritten to original documents.

ChangeStatus

Document ----> Document

Document must be changed to EDITING status.

CopyResource

Document ----> DocumentStore

Copy the original document to WorkSpace (i.e. in /imagedata for Weiss2 and /work for other models).

SetProperties

Document ----> DocumentStore

Set the Original BoxBasePath, BoxNumber, FolderName and DocumentNames as WebDAV properties to the document in Workspace (i.e. in /imagedata for Weiss2 and /work for other models) for reference**.**

CopyPage

Message documentation

----> **Document**

User requests BoxDocument to copy a page from the document. Copied page will be in clipboard until paste is called.

WorkSpaceInit

Document ----> Document

Update the m_XXX variables, so that any further changes happen on the delta document and not the original document.

CopyResource

Document ----> DocumentStore

Copy the document from Workspace path (i.e. in /imagedata for Weiss2 and /work for other models) or the original path to clipboard.

DeletePag

Document ----> Document

Delete all the pages from the clipboard document except the ones that are copied by the user.

SetProperties

Document ----> DocumentStore

Set the WorkSpace (i.e. in /imagedata for Weiss2 and /work for other models) BoxBasePath, BoxNumber, FolderName and DocumentNames as WebDAV properties to the document in ClipBoard for reference.

PastePage

----> **Document**

User requests BoxDocument to Paste a page previously cut or copied. The document in which user is trying to paste should be in EDITING status.

WorkSpaceInit

Document ----> Document

Update the m_XXX variables, so that any further changes happen on the delta document and not the original Document.

GetProperties

Message documentation

Document ----> DocumentStore

Get the properties like m_XXX values set previously. i.e., srcBoxBasePath, srcBoxNumber, srcFolderName, srcDocumentName

---

GetPage

Document ----> Document

Get the page from the Clipboard path.

AppendPage

Document ----> Document

If the position to paste is at the end of the document, then call AppendPage.

InsertPag

Document ----> Document

If paste position is not at the end of the document, then call InsertPage.

GetDocument

Document ----> Document

Obtain the reference to the original Document which was cut.

Delete

Document ----> Document

Delete the document which was cut before.

Save

----> **Document**

WorkSpaceInit

Document ----> Document

Update the m_XXX variables, so that any further changes happen on the delta document and not the original Document.

GetProperties

Message documentation

Document ----> DocumentStore

Get the properties like m_XXX values set previously. i.e., srcBoxBasePath, srcBoxNumber, srcFolderName, srcDocumentName

MoveResource

Document ----> DocumentStore

Move the updated delta document to the original path.

RemoveProperties

Document ----> DocumentStore

Remove all the temporarily set properties like srcDocName, etc from the document.

DeleteResource

Document ----> DocumentStore

Delete the document if it was copied to clipboard.

ChangeStatus

Document ----> Document

In the end, change the status of the document to READY.

Activity Diagrams

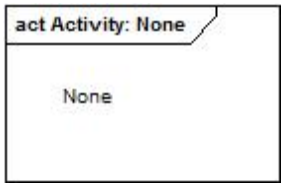


Figure 13 : Activity : None

5.4. EXECUTION MODEL

Deployment diagrams

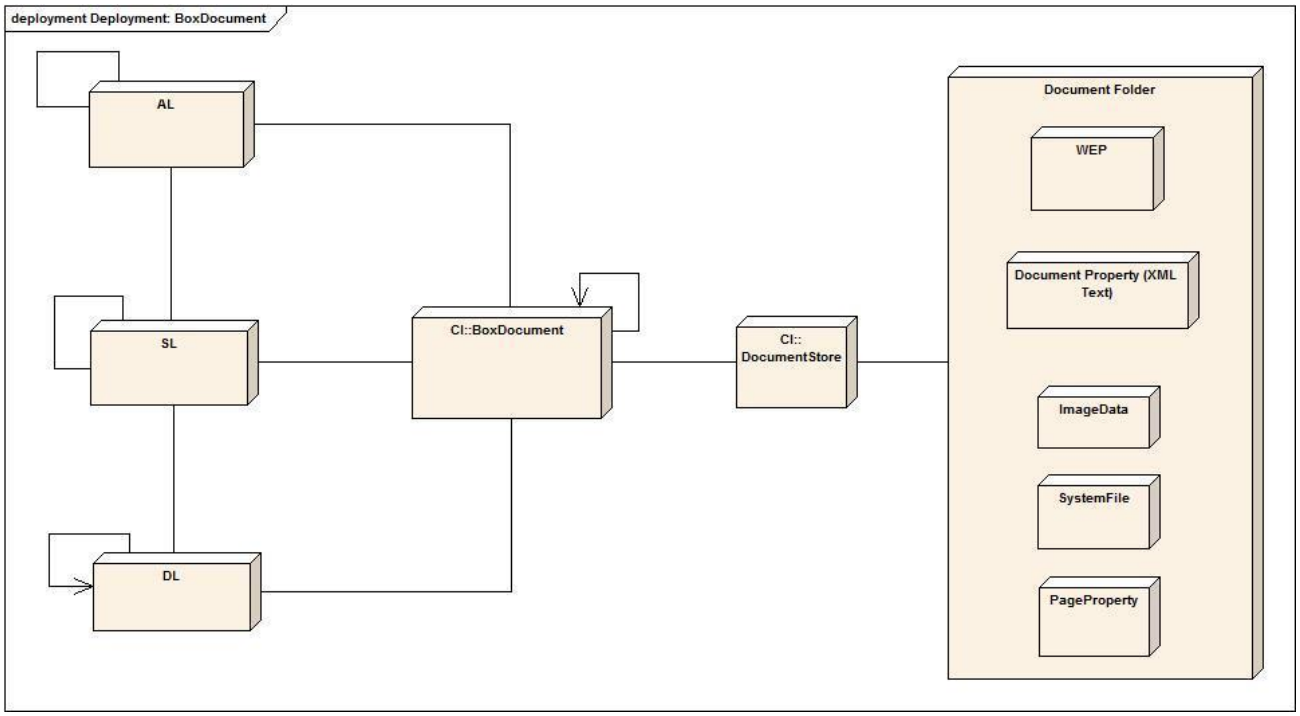
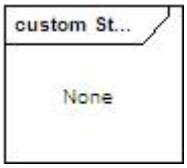


Figure 14: Deployment: BoxDocument

5.5. UI FLOWS



Appendix