

Remote Document Vault

OCD

Project #3

Instructor
Dr. Jim Fawcett

Nikhil Malladihalli Chandrappa
Fall 2013

1. Executive Summary.....	3
2. Introduction	4
3. Use cases.....	5
4. Views.....	7
5. Structure	12
6. Operations	17
7. Critical Issues.....	21
8. Conclusion.....	24
9. References	25

1. Executive Summary

In this age to information it's a common place for organizations to have large amounts of information stored as text files in a central repository that requires concurrent access, whether its project documentations of a large enterprise application or research papers of a R&D department accumulated over the years. Important features for such a system is storing of the files in a structured manner that provide swift access to the information requested by multiple users.

This presents a classic case for client/server based architecture and file categorization which enhances the accessibility of files. This architectural document proposes one such tool called The Remote Document Vault.

The Remote document vault is a client/server based system which allows the users to store text files remotely in a structured manner by providing the functionality of categorization and also allows users to define a parent/child relationship between the files, this will support quick and structured access to text file contents, its properties and relationships.

Document vault will consist of following components and modules,

1. Vault Client GUI Module
2. Display and search components
3. Document Vault Module
4. Document Vault GUI Module
5. Relationship Builder Tool

Some of the Important design consideration and critical issues of Remote Document vault,

- Vault client will be designed as a component based structure with system having well defined pluggable component parts
- Vault client will have tabbed display for insertion and viewing of files
- Latest copy of Parent relationship map will be retrieved during client startup and stored in local cache.
- Relationship builder tool will be hosted as different service whose functionality will be used by Document vault module
- Functionality of adding new categories will be restricted to administrators of document vault
- Maintaining state information between client and server
 - Client/server will have per session policy instantiation
 - Client/server will have Duplex style of communication supporting asynchronous requests
- File Transfer between client and server,
 - Files will be transferred in blocks
 - To support multiple files transfer requests, Globally Unique ID (GUID) will be used for each incoming request.
 - Document vault will have a blocking queue implementation for handling file insertion/extraction request where a single thread will work on the queue to maintain the consistency of the repository.

2. Introduction

Remote Document Vault consists of vault client and a Document vault located at a remote location. Vault client will provide an easy to use GUI for file handling and GUI will be implemented using Windows presentation Foundation (WPF). Document Vault is a server application which will be developed on .Net platform to facilitate client requests. Communication between Vault Client and the Document Vault will be implemented using Windows Communication Framework (WCF).

2.1. Vault Client GUI

This is the UI layer of the tool which provides tabbed interface for the users. Client will provide the following tabs

- a) File Upload Tab (Users)
- b) Display Tab
 - Category view
 - Parent/child relationship view
 - Text/File content display view
- c) Search Tab
- d) Metadata update Tab
- e) Admin Console Tab

Vault client will be designed as component based structure for handling user requests, please refer to section “5.2” for overall architecture of the client.

2.2. Document Vault

Document vault will be the server application which will process the client requests in a Single Threaded apartment model where a single thread will work on blocking message queue. Document vault will be support insertion, extraction and searching of text and metadata files in the repository.

Document vault will have following modules,

- a) Server Module
- b) File Manager Module
- c) Query Manager Module
- d) Metadata generator Module

2.3. Relationship generator Tool

This is a standalone tool that will scan through all the metadata files in a repository to build a map of parents for each file in the vault. This tool will be hosted as service and will be used by the server every time the repository starts and when a text file is inserted or updated in the repository.

3. Use cases

Remote Document Vault can be used in places where structured storage of documents with concurrent access is essential for daily activities like Application Development teams, Project Management teams, Content providers, libraries and general public file hosting website to name a few. Some of the specific use cases of the tool are given below,

Users: Application Developers

Use Cases: Application Developers working in a parallel development environment from multiple locations will use the Vault client to store the application source code in document vault, they will use categories and parent/child relationship for segregating the source files belonging to different modules of the overall application.

Impact:

- Users will launch the vault client, they will get tabbed GUI interface and client will connect to the server.
- Users will go to File Upload tab, browse the local file system to select the files for insertion
- Users will provide category and dependencies of the file being inserted
- User will click on "Upload File", request will be sent to the server and file transmission starts
- Once the file is received at the server, server generates the metadata file and stores it in the repository.
- After the file store is completed, server will call the relationship builder to update the parent map which will be returned to the client.

Users: Librarians

Use Cases: Librarians will use the tool to perform structured search of millions of books and journals in the repository based on the category. If a student request for availability of a String Theory Journal authored by "Albert Einstein", librarian will search the repository under Journals and under physics category which considerably reducing time required to access the file

Impact:

- Users will launch the vault client, they will get tabbed GUI interface and client will connect to the server.
- Librarian will navigate to category view and navigate through the categories to determine the availability of the file
- Or Librarian can go to search tab and perform a metadata search for category = "Physics" and Text="String theory, Albert Einstein", this search will return all the journals in the category physics with text string theory and Albert Einstein in them.

User: Vault Administrator

Use Cases: Vault Administrator will have the privileges of introducing new category in the system and if user wants a new category to be created, user has to send a request to administrator

Impact:

- Admin will launch the Server GUI and will get view for category

insertion.

- Admin will review the request and makes the category based on the user request.
- Category list in the system will get updated with newly added one.

3.1. Use of Categorization functionality provided Remote Document Vault for structured storage of files:

Let's consider a simple example where IT dept of a financial services company is incorporating our Remote Document Vault to store project artifacts in below structure.

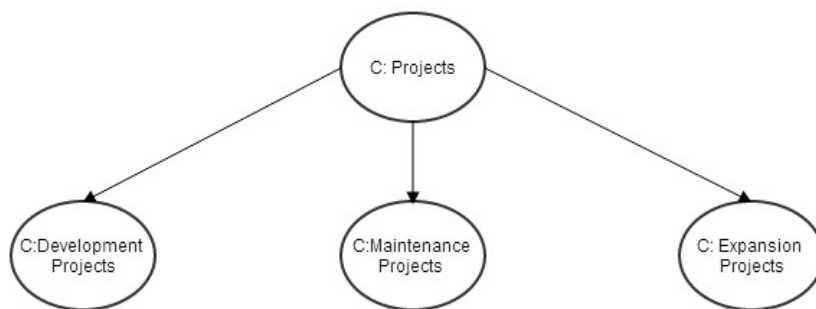


Fig 3.1 (a)

Administrator creates category hierarchy as show in the Fig 3.1(a)

Use case: Project Manager wants to upload project artifacts for a new development project, Project Initiation Request Document will be parent having dependencies on CRD, resource allocation and budget information documents.

- Project Manager will launch the Vault Client and selects multiple files for uploading
- Selects category as “Development Projects” and provides other metadata information then sends an upload request to the server.
- Document vault will service the request by storing the files in the repository.

Above use case demonstrates the structured storage supported by remote document vault.

Use case: Project Manager wants to view the budgeting details of a Development Project.

- Project manager will launch the Vault Client and navigate to Display tab.
- In category view section, navigates to Development Project section and selects on the required project Initiation Request Document
- In children view will display the dependencies of file and selects the budgeting details document.
- Contents of the document will be displayed in “Text Contents” section.

4. Views

Remote Document vault has Vault Client GUI and Admin Console GUI.

4.1. Vault Client GUI

When user launches Vault Client, User will be present with “File Upload” tab which will be the default tab presented on client startup.

4.1.1. File Upload tab

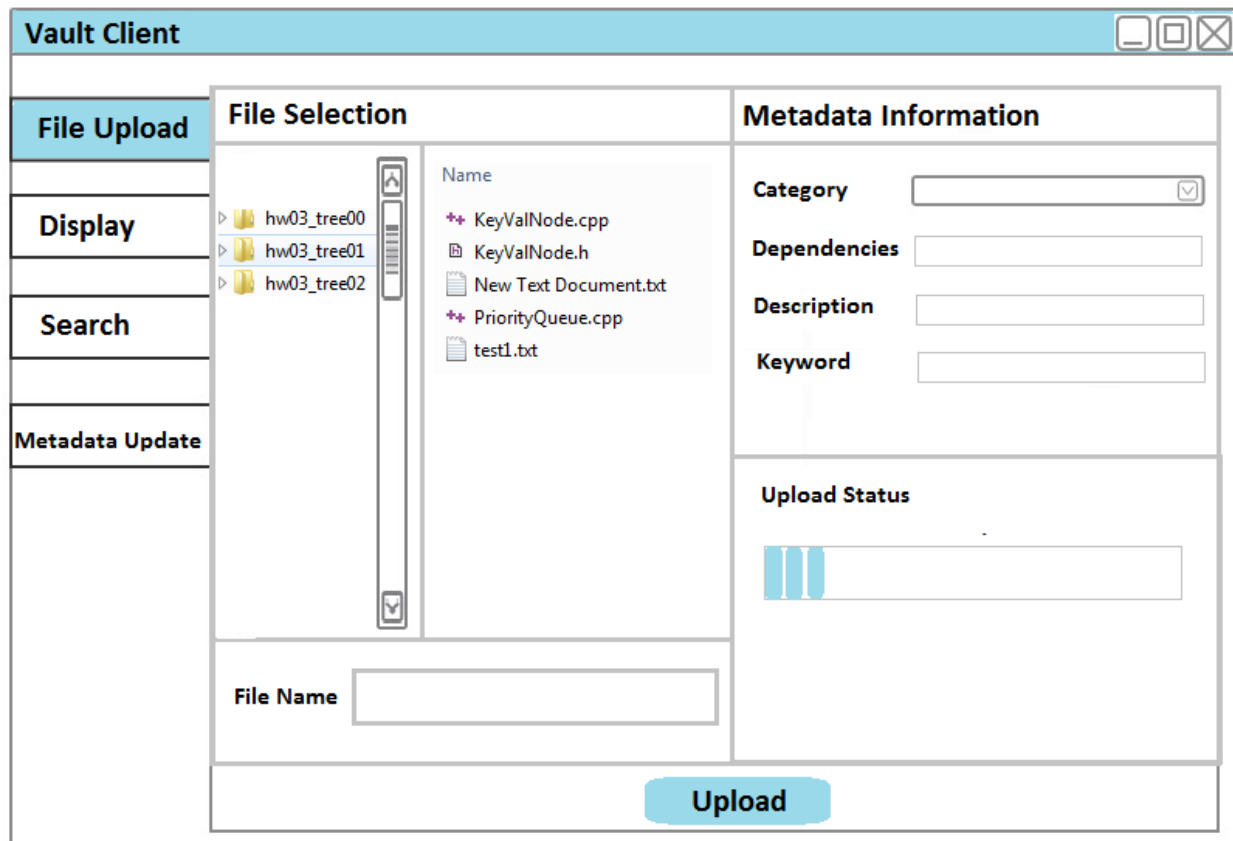


Fig. 4.1.1

File Upload tab will be designed as show in the Fig 4.1.1, it consists of

- File Selection section – It provides local browsing for files to insert into remote repository
- Metadata Information section – It accepts the input from the user for metadata creation
- Upload Status section – It shows the status bar and status messages returned by the server
- Upload Button – It will send a request to user to file upload.

In Metadata Information section, all the fields except Category will have a input text box and since users can select multiple categories it will be a multiple selection drop down box which will have all the categories in the system.

4.1.2.Display Tab

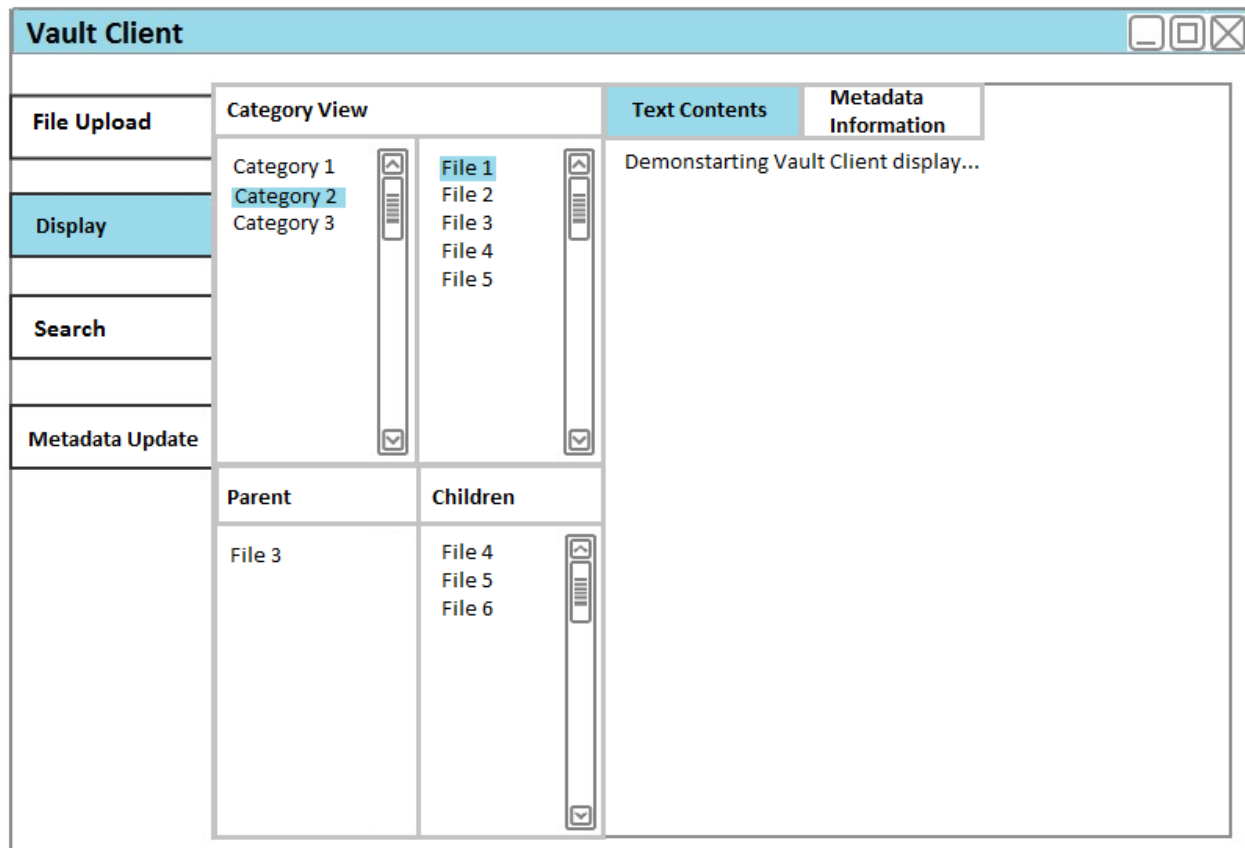


Fig 4.1.2

Display tab will be designed as show in the fig 4.1.2, it will consist of three section

- Category View – shows the categories in the repository along with associated files and when user selects on a file that will be current file. Parent/Children section, Text contents and Metadata information will be updated accordingly
- Parent/Children – Shows the parent and children for the current file
- Text Contents – will display the contents of the current file
- Metadata information – will display the metadata information of the current file
- Text Contents and Metadata information will be in a tabbed display, on clicking on tab corresponding view will be rendered.

4.1.3.Search Tab

Text Search Tab: fig 4.1.3 (a)

- It will have input box to accept query string, category list box accepting multiple inputs and search button which triggers the search operation.
- Search Progress section will shows the search operations that are currently running

- Search results section will display the “fully qualified” file names, each result will be a clickable link. When user selects a file, it will become the current file and display view will get rendered accordingly.

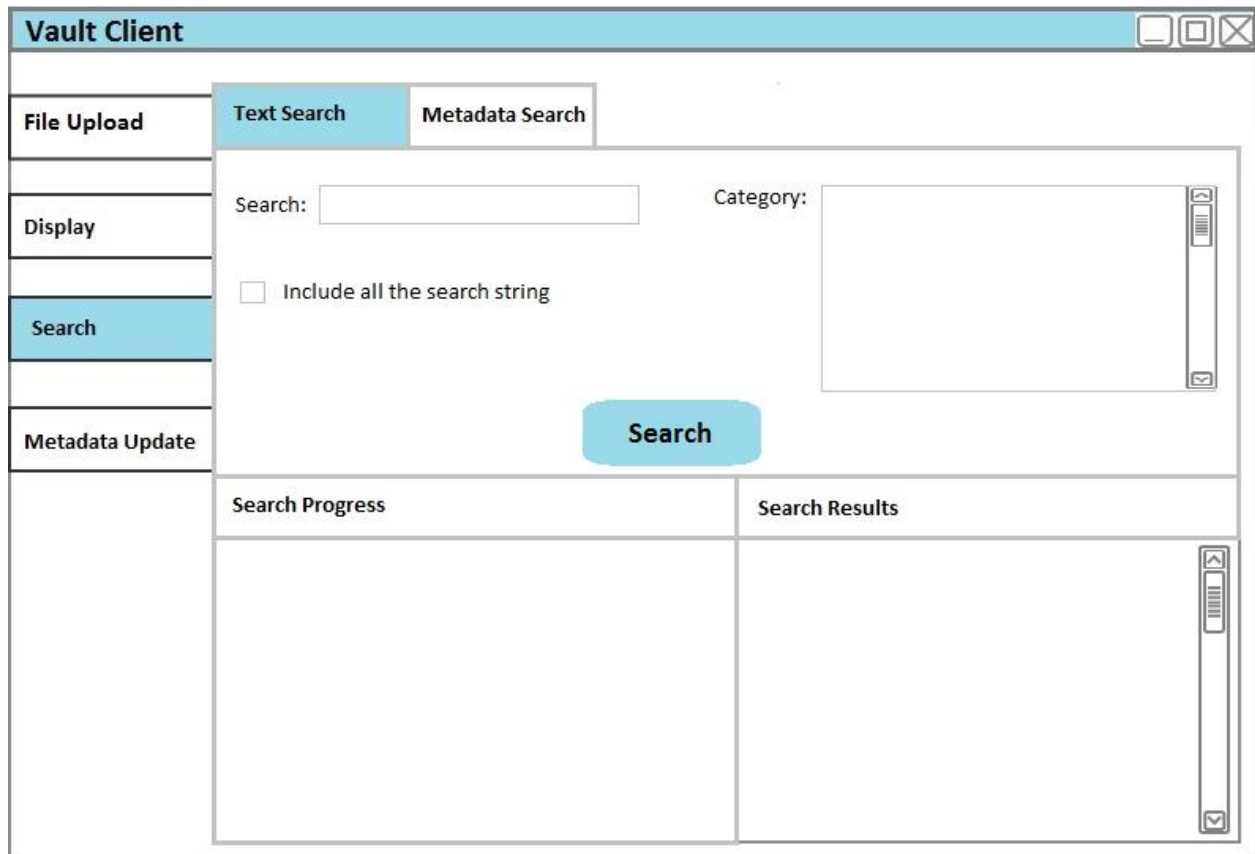


Fig 4.1.3 (a)

Metadata Search Tab: Fig 4.1.3 (b)

- It will have scrollable multiple selection list box for selecting tags and categories for metadata search.
- Similar to text search tab, metadata search tab will also have search progress and search results section.

The interface features two tabs at the top: "Text Search" and "Metadata Search". The "Metadata Search" tab is active. Below the tabs, there are two lists of metadata fields. The "Tags:" list includes Dependencies, Keywords, Last Modified Date, Description, Name, and Size. The "Category:" list includes Category 1, Category 2, and Category 3. A blue "Search" button is centered below these lists. At the bottom, there are two columns: "Search Progress" on the left and "Search Results" on the right, both of which are currently empty.

Fig 4.1.3 (b)

4.1.4. Metadata Update Tab

The "Vault Client" window shows the "Metadata Update" tab selected in the left sidebar. The main area is divided into "Current Values" and "New Values" sections. The "Current Values" section displays the following metadata fields: Categories, Dependencies, Description, Last Modified Date, Keywords, Name, and Size. The "New Values" section contains input fields for Category (with a dropdown arrow), Dependencies, Description, and Keyword. A blue "Update" button is located at the bottom right of the form.

Fig 4.1.4

Metadata Update Tab will support the updating of metadata attributes of current file selected in the Vault Client. Metadata Update Tab will be designed as shown in the fig 4.1.4

It displays current values of metadata attribute in “Current Values” section and provides functionality to update the metadata attributes in “New Values” section. Update action button will send a request to the server for updating the metadata.

4.2. Admin Console GUI

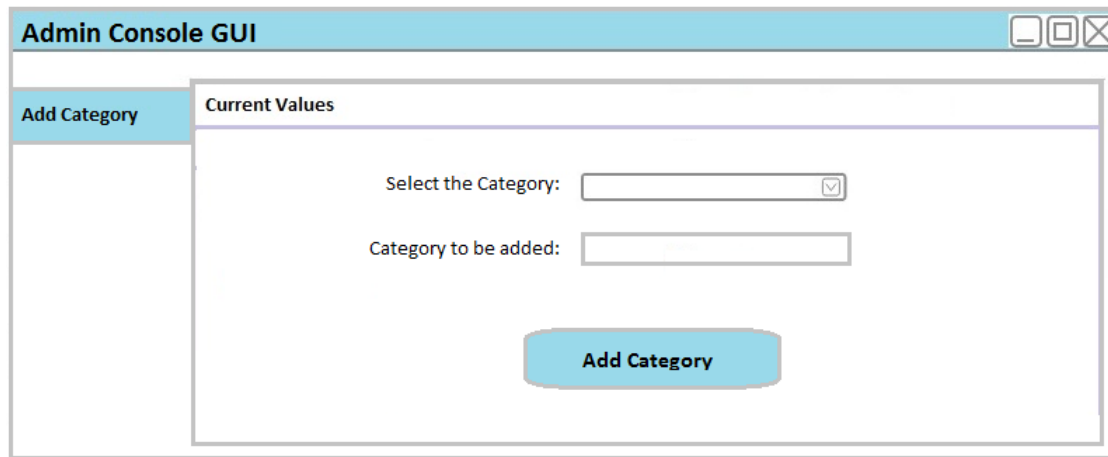


Fig 4.2

Admin Console GUI is provided at Document Vault Server which can be accessed only by the administrators of the tool. It supports the addition of new category to category list by updating the category hierarchy xml.

- Administrator can select the category hierarchy using drop down list of “select the category” field.
- Administrator inputs the new category to be added in “Category to be added” field.
- On clicking add category button, a request will be sent to server for updating the category hierarchy xml.

5. Structure

5.1. Overall Architecture

Vault client will be designed as a component based structure with system having well defined pluggable component parts and Document Vault will be designed to have separate modules performing well defined operations, overall architecture of the systems is provided below

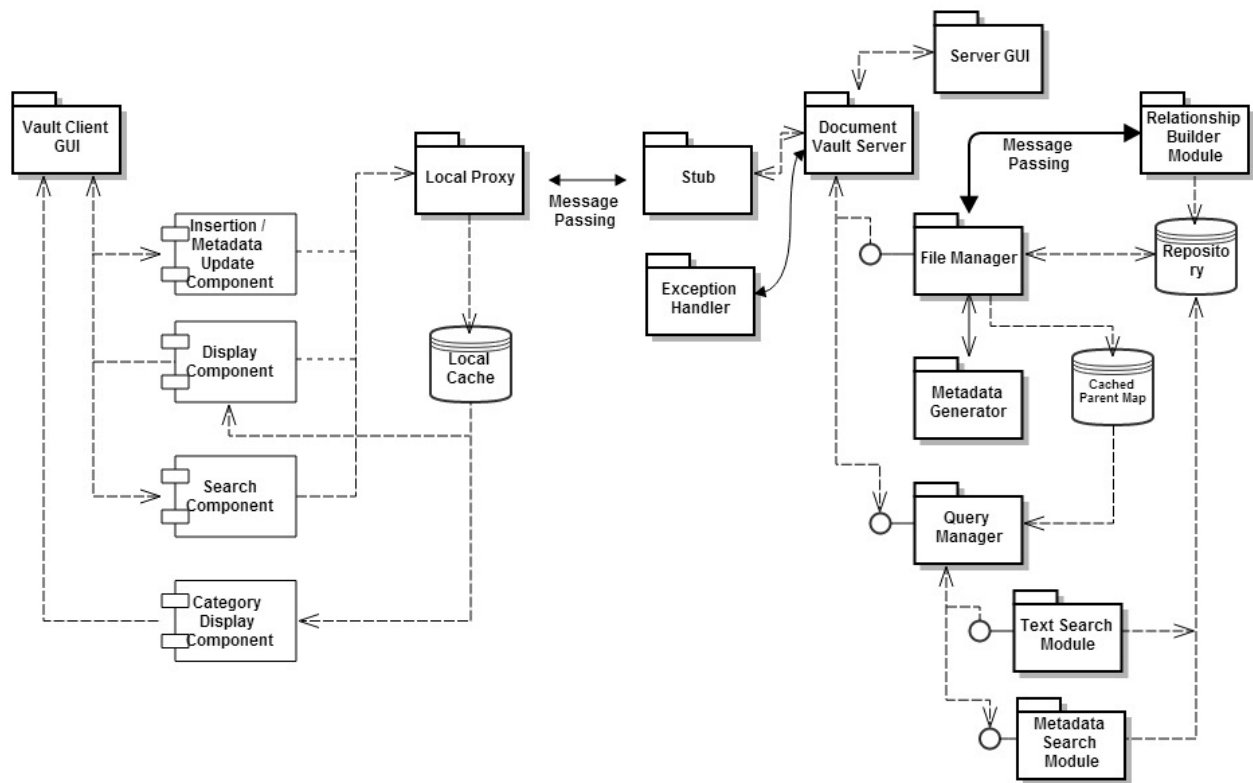


Fig 5.1 Overall Architecture

5.2. Vault Client modules and components description

5.2.1. Vault Client GUI

Vault Client GUI is the UI layer of the tool communicating with the users, Vault client provides a tabbed interface for the users to perform insertion, updating and retrieval of files into the remote repository.

Client will provide the following tabs

- File Upload Tab (Users)**
 - In this tab user will select the file to be uploaded and provide the details for category and dependencies.
 - Action Button "Upload" will be bound to component "Insertion/Metadata Update"
- Display Tab**

- This tab will have Category view, Client/parent view and content display view
 - Category view will be bound to “Category Display” component
 - Client/Parent and content display View will be bound to “Display Content” component
- c) Metadata update Tab
- Users will be able to update the metadata attributes of current file in the tab.
 - Users will be provided with a list of metadata attributes with current values and textboxes to enter new values for the tags
 - Action button “Update” will be bound to component “Insertion/Metadata Update”
- d) Search Tab
- This tab will accept the text and metadata search queries
 - This tab will be bound “search” component and also will have a content display view to display the contents of the file selected from the search.

5.2.2. Insertion/Metadata Update Component

This component will handle the insertion of the new file into the repository, it takes the local file selected by the user and user inputs for metadata creation, calls the service exposed by the server to upload the file.

5.2.3. Category Display Component

This component is responsible for rendering the Category view of the display tab, component reads the category map from the local cache for rendering category view.

5.2.4. Display Component

Display Component is the core component responsible for rendering the child/parent view and content display view of the file requested by the user.

- Display component reads the parent relationship map from the local cache and renders the child/parent view for the current file selected by the user.
- Display component checks the local cache if requested file is available in the cache to display the contents to the users, if it's not present it will make a service call to server for the required file.
- Once the file is returned by server, it reads the file provided by the proxy and renders it in display view.

5.2.5. Search Component

Search component is responsible for handling the text and metadata search queries done by the users.

- It makes a request to service exposed by server to search files. Once the results are returned, it displays to the users in a scrollable panel.

- Each file in returned result will be rendered as link and scrollable panel selection will be bound to Display Component.
- Clicking on a file in scrollable panel will make it as current file whose contents will be rendered in Display view of Search Tab.

5.3. Remote Document Vault Modules description

5.3.1. Server Module

Server module is responsible for exposing the services for the clients and handling the request from the clients, Server module will provide interfaces for: ISearch and IFileManagement and uses the IParentMap interface exposed by relationship builder tool.

- Server module will implement blocking messaging queue for handling client requests with a single thread working on request.
- Server will instantiate a factory of service objects and invokes objects corresponding to request being processed.
- When client sends a request to connect server module sends category map and parent map to the Vault client.

5.3.2. Administrative Console GUI

As a part of design consideration we have restricted the addition of new tag attributes and categories in metafiles to general users, they can be updated only by the administrators which will be supported by admin console UI.

- An admin can add new categories requested by the user by the help of Administrative Console GUI.
- Admin enters the new category and which will be updated in the list of categories maintained by Document Vault server

5.3.3. File Manager

File Manager is responsible for actual insertion, extraction and updating the text and metadata files in the repository,

- Files manager exposes the functionality for file insertion, updating and extraction
- This module is responsible for finding the references for all the files in the repositories and provides I/O capabilities
- Every time a new text file is inserted or existing metadata file is updated, file manager Calls the relationship builder to get latest parent map and stores in the cache and also returns a copy to Document Vault server module
- File Manager is also responsible for parsing the category hierarchy xml for generating a category map and storing it in the cache.
- File Manger will implement the interfaces of : File Management and Parent Map

IFileManagement.uploadFile()	This method will be responsible for calling metadata generator for creating metadata file for the new file being uploaded and storing them in the repository.
IFileManagement.updateFile()	This method is responsible updating the metadata file and handling versioning of the metadata file.
IFileManagement.extractFile()	This method scans the repository for requested file and will return the file being requested by the user.
IFileManagement.generateCategoryMap()	Creates a map of the categories and their associated files by going through all the metadata files in the repository.
IFileManagement.getParentMap()	Calls the service exposed by relationship builder tool to the latest parent map and stores in server cache.

5.3.4. Query Manger

This module is responsible for interpreting the search requests to invoke either text search module or metadata search module,

- For text search it will invoke the text search module by passing search strings and options if any as parameters.
- For metadata search it will invoke the metadata search module by passing tag elements as parameters.
- Then Query processors send the results to Document Vault server module.

5.3.5. Text Search Module

Text search module is responsible for actual search operation of text files and will implement ISearch interface. It iterates through text files in the repository and searches the file contents for query strings. Fully qualified name of files which had matching search string will be returned to Query processing module.

Text search module will implement ISearch interface and it will also handle the text search switches based on the indicator say "OptionalSearchInd"

optionalSearchInd	If this indicator is set to "true", Searches for at least one of the query strings in search command.
	If this indicator is set to "false", Searches for all the query strings provided in the search command.

5.3.6. Metadata search Module

Metadata search module is responsible for handling metadata search based on categories and metadata tags, it searches through all the metadata files in the repository and returns the contents of each element of the metadata for every file in the file set which had a matching tag search string. If metadata search query contains categories, metadata will search the category map to retrieve the matching files which will considerably increase the search performance

5.3.7. Exception Handler

Error handler module will be a responsible for handling all the errors and exceptions that get generated in the Remote Document Vault tool.

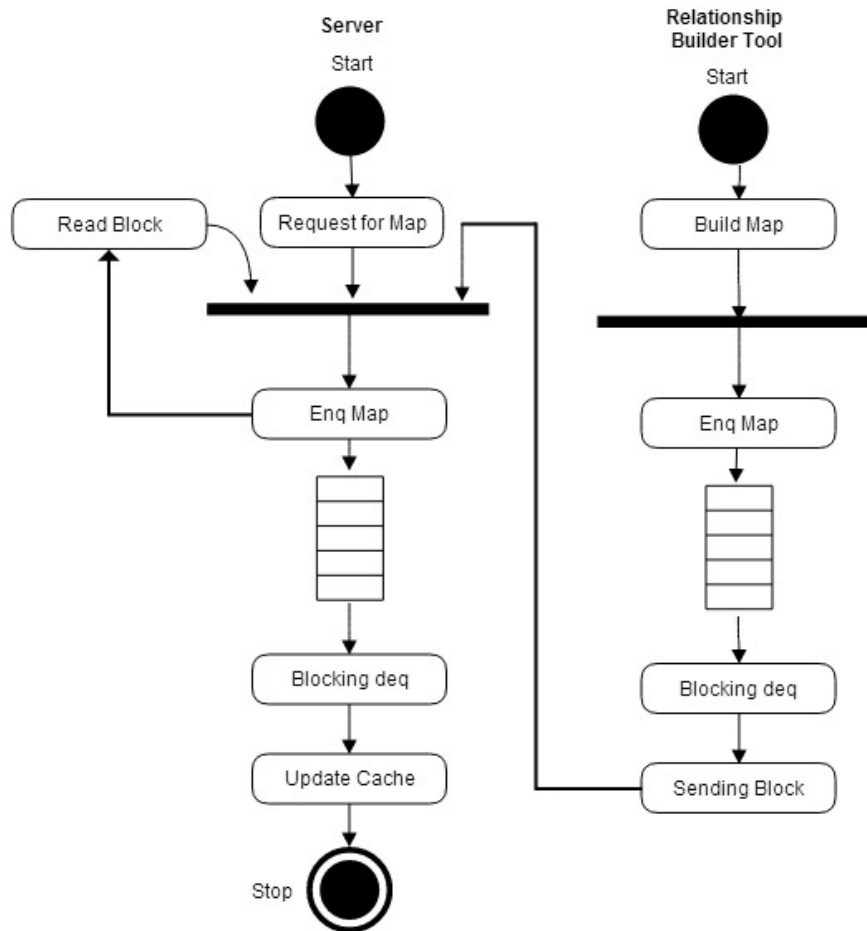
- Error handling framework will provide an interface `IServiceExceptionMessage` for exception handling which will be handled by Document vault server.
- Error handling framework will allow modules to add error numbers which will returned to error handling module for error handling.
- Error handling framework will get corresponding description of error numbers and will be displayed to the user. This error framework will support internationalization, only by changing the resource file error message can be displayed in local language.
- Error handling will also write the exception stack trace to the log file which will be helpful in analyzing the issue for Maintenance team of the tool.

5.4. Relationship Builder Module

This module searches through all the metadata files in the repository and builds a map of parents for each file in the vault and it exposes its functionality as a service in `IParentMap` which will be called upon by File Management module.

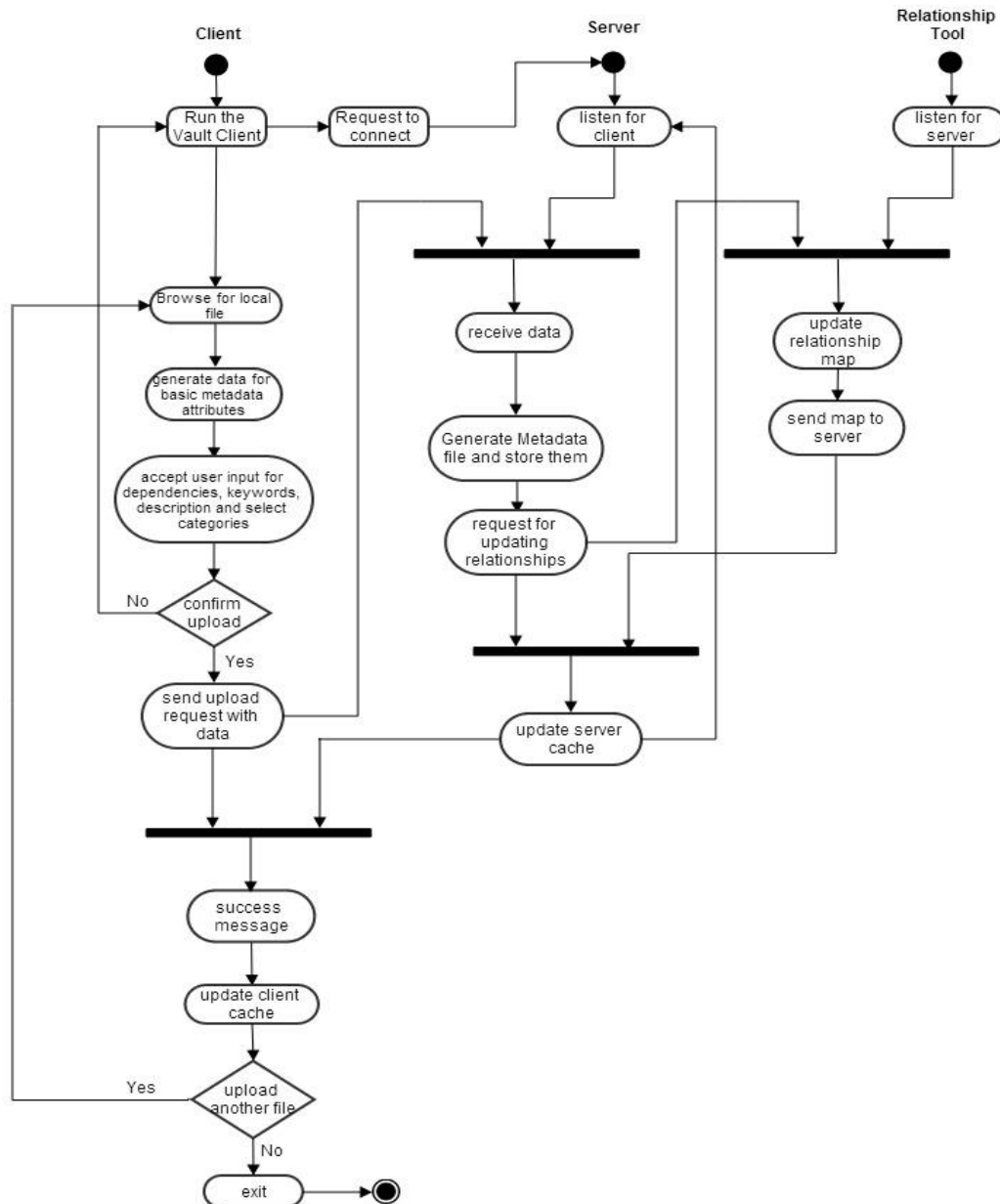
6. Operations

6.1. Server Startup activity diagram



Above diagram describes the sequence of activities that takes place on server startup, on server startup it builds category map and it requests for parent map from the relationship tool builder which will be stored in the server cache for future use during search operations. When the server gets a request to connect from the clients, it sends a copy of category map and parent map to be cached at client side.

6.2. File Insertion activity diagram

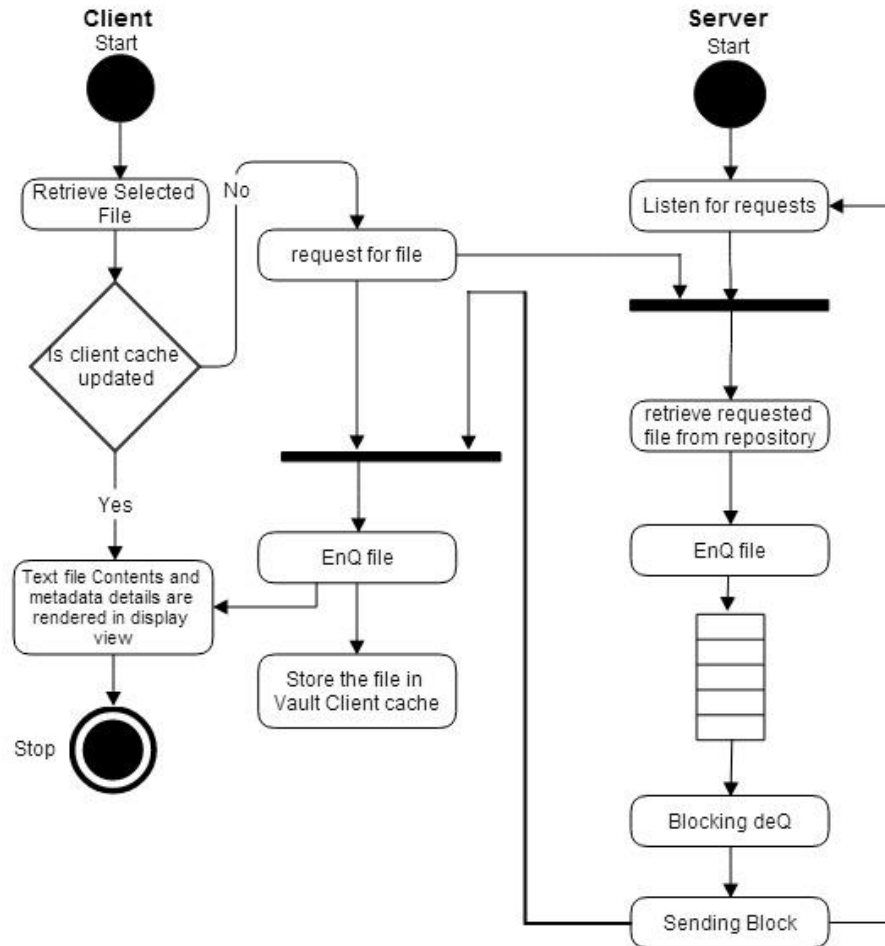


During Vault client startup it will send a request to connect to the server, user will be shown the insertion tab.

- User will select the file to be uploaded from the local system and inputs the metadata information like file category, dependencies and descriptions.
- User will click on the "Upload File" action button, which sends a request to the server of file upload along with data.
- Server will process the request by generating the metadata file with user provide inputs and stores the new file and metadata in the repository

- Server then sends a request to relationship builder for updated parent map and stores in the server cache.
- Server then returns the parent map to client, it will be stored in the client cache thus maintaining coherency between client and server.

6.3. Current file content and metadata display activity diagram



User will be in the Display tab of Vault Client GUI, user selects on a file from category view or parent/child view which sends a request to Display component,

- Display component checks whether client cache is updated and has the files being request.
- If cache has the latest copy then renders the display view with contents from the cache.
- If not, client will send a request to server for the file being request.
- Server will process the request and retrieve the file from repository.
- Server then sends the file and its metadata as blocks over to client which will then render them in display view.

6.4. Metadata update activity diagram



User will be in the metadata update tab of Vault Client GUI,

- User provides the new values for the tag attributes and clicks on action button “update”.
- Insertion/metadata update component will send a request to server for updating the metadata information.
- File Manager searches for the previous metadata version and extracts its contents. Builds a newer version of the metadata file with contents provided by the client and previous tag values.
- Server calls relationship builder service to update the parent map which will then be stored in the server cache.

- Server sends the map along with status message to client, indicating the successful update of the metadata file.

7. Critical Issues

The Remote Document Vault has two main performance parameters, response time and structure storage/retrieval of the files from the repository. At any given point of time server can have multiple Vault clients sending request, server should be able to service all the requests in a reasonable amount time without any conflicts, lets discuss some of the critical issues that are impacting the main performance parameters

- **Service Object Lifetimes**

Maintaining the state information between client and server is an important criterion impacting the usability and performance of remote document vault, if we make service objects per call or singleton every time a request comes in from the same client a new server object is spawned for it which will considerably decreases the server performance.

To avoid creation of large number of server objects at given time, we will maintain per session design for maintain the state information

- **Request Handling**

When working with a file repository it essential to maintain the transactional consistency in the repository. There can be situations where request might come in to updated the same file or try to access a file which is already opened for another client, such situation will result in I/O exception.

Server:

To avoid conflicts with concurrent file access we will implement server with a blocking message queue where a single thread will be working on the queue.

Client:

If a file being requested is already opened by another client, exception will be handled and in catch block, client repeatedly tries to access the file. Maximum retries will be restricted to 50 times.

- **File Transfer in blocks**

We made a design decision to transfer the files over the networks in blocks which will reduced the load on the network but it present a situation to server where keeping track of multiple file transfer requests will be complex.

We can reduce the complexity by assigning a Globally Unique ID to file transfers. When file transfer request comes in client has to send a GUID assigned to request message is attribute say "fileTransferId" which will be used to distinguish between the file transfers.

- **Insertion of the file with incorrect child dependencies**

When user provides metadata information during file upload, if files provided as part of the dependencies are not present in the repository and building parent map with incorrect relationship will result in malfunctioning of the tool.

- a) To avoid the incorrect child dependencies, server has to validate the dependencies provided by the user by searching the repository for the metadata files. If files are not present failure message should be sent to vault client.
- b) If users are uploading multiple files in a top down approach i.e. first uploading the parent then its children, client has to indicate the multiple file upload with help of an indicator in the request message which will make the server to hold a lock until all the files are received and then validate the child dependencies.
- c) Another solution would be show a advisory message to user perform multiple file insertions in bottom up approach i.e. first uploading the children then the parent.

- **File Ownership**

Access right for the file uploaded to the repository for users other than the up loader.

- Only Author of the file or person who uploaded the file will have edit right to the files
- If a user wants to change the contents of a file, he needs to update the cached copy in vault client and perform a new upload thus maintaining the consistency of the files.
- This might well require an authenticated login to the Vault client to identify the user, for this OCD it's out of scope and can be considered for future releases of the tool.

- **Insertion of text files with the same name**

Repository will not allow the insertion of the files with the same name. Repository supported by this tool will not have sub directories hence tool will not support insertion of files with same name.

- a) When server receives a request to upload a file with name already present in the repository it sends an error message to the Vault client which will be displayed to the user.
- b) When two file upload request has the same file name, service request which enters the blocking queue first will have successfully file upload and error message will be sent to the other file upload request.

- **Addition of New Categories**

There can be multiple users wanting categories of their own which may not be of any use to the overall system, thus providing this privilege to users will affect the structured storage and retrieval of file supported by the Remote Document Vault.

- New categories can only be added by an administrator of the system and functionality will be supported only by Document Vault server GUI.

- Users who require a new category should sent a request to architect describing the need for a new category
- Architect reviews the request and sends the request to administrator for adding the category.

- **Archiving the old files**

Document Vault supports versioning of files, any updated to the existing file will result in a new version to be stored in the repository. Soon the repository will have hundreds of older version of files which are not being used hence document vault has to support archiving of the older version of the files.

Archiving of the files will be done programmatically. Server will archive files which are not accessed over six months into the archive repository.

8. Conclusion

The Remote Document Vault provides a robust file storage repository with following features,

- Tools support structured storage of files with categorization and parent/child relationship
- Tools provides concurrent access to remote repository with Vault Client
- Tool supports well designed tabbed GUI in Vault Client which enhances the usability of the tool
- Vault Client provides Display view which supports interactive navigation of file repository
- Accessing of files are faster with structured search provided by metadata search functionality
- Tool can be easily incorporated in a real world environment like IT organizations, libraries etc. with minimum rollout time
- Critical issues are reasonably resolved

9. References

- [1] Test Harness OCD, by Gang Cheng, provided by Dr. Fawcett
- [2] Modular Quality Assurance toolset OCD, by John Walthour, provided by Dr. Fawcett
- [3] <http://www.lcs3.syr.edu/faculty/fawcett/handouts/CSE681/Lectures/ProjectHelpers.htm>