GLOBAL SCALE DOCUMENTUM IMPLEMENTATION - DESIGN CONSIDERATIONS

EMC Proven Professional Knowledge Sharing 2011

Sameer Dinkar Patil
spatils@gmail.com

Table of Contents

1. INTRODUCTION

Companies which have worldwide presence face different challenges while implementing applications which cater to the requirement of all employees globally. Implementing a global scale document management system is no exception. Typical challenges faced while implementing document management system are mentioned below

- Implementing country-specific requirements for Security, Taxonomy, Object Model
- Meeting local standards for network, database, LDAP (Lightweight Directory Access Protocol), and coding
- Country-specific policies with respect to storage and retention period
- Different language requirements
- Country-specific functional requirements
- Compatibility with local application interacting with document management system
- Migration of data from existing local repositories

To meet these challenges, it becomes essential to define which part of requirement is to be met by coding and which part will be picked up from configuration. These boundaries should be defined during the design phase, taking product features and limitations into account.

This paper discusses the principles followed while designing repository, application integration, and migration to meet global requirements. Recently, I took part in a global implementation for an insurance brokerage firm. This document uses that implementation as the basis of discussion.

Each design aspect has two parts. The first part is challenges section. In this section I explain in detail the design objectives and challenges with respect to implementation and highlight the important areas with respect to certain areas of design. The second part of the section describes how these challenges are met. It gives the design approach taken to meet the requirement and the rationale behind the approach.

2. REPOSITORY CONFIGURATION

2.1. Challenges

- Deciding physical location of repository
- Meeting legal requirement of countries
- Deciding number of repository instances
- Setting up user access across multiple repositories

2.2. Design consideration

Network topology of the organization typically decides the physical location of the document management repository. Ideally, the repository should be located near the user base. If it is not possible to install separate repositories, then BOCS (Brach Office Caching Services) can be used to support access of documents in each country. BOCS supports both push and pull mode. Contents are pushed into caches using DMS (Documentum Messaging Services). Each country has its own BOCS server to serve its users. As shown in Figure 1, each region has its own repository. For Asia-Pacific region, Australia and Hong Kong has their own instances of BOCS servers.

Legal requirements of countries may not allow documents to be stored outside the country boundaries. For example, EMEA do not, by law, allow its business documents to be stored outside its physical boundaries. In such cases, a separate repository is created in each region.

Since each repository instance requires an administrator for repository and database maintenance, it is advisable to limit the number of repository instances based on the network topology and legal requirement.

There are small set of islands called Bermuda which are part of British territory but are located near east coast of USA. Legal requirements do not allow physical storage of documents outside Bermuda. Instead of installing a separate repository for Bermuda, MDO (My Documentum for Outlook) was used to physically store these documents in the same country. All documents of the Bermuda territory are stored in a separate folder. This was achieved Using Documentum →Offline-Enable→ Current folder and Subfolder option. Although one copy of document and metadata is in the US repository, it met security requirements.

Users who access documents across multiple repositories need to be imported into those repositories and added to respective groups. All Repositories project to Connection Broker of all regions.

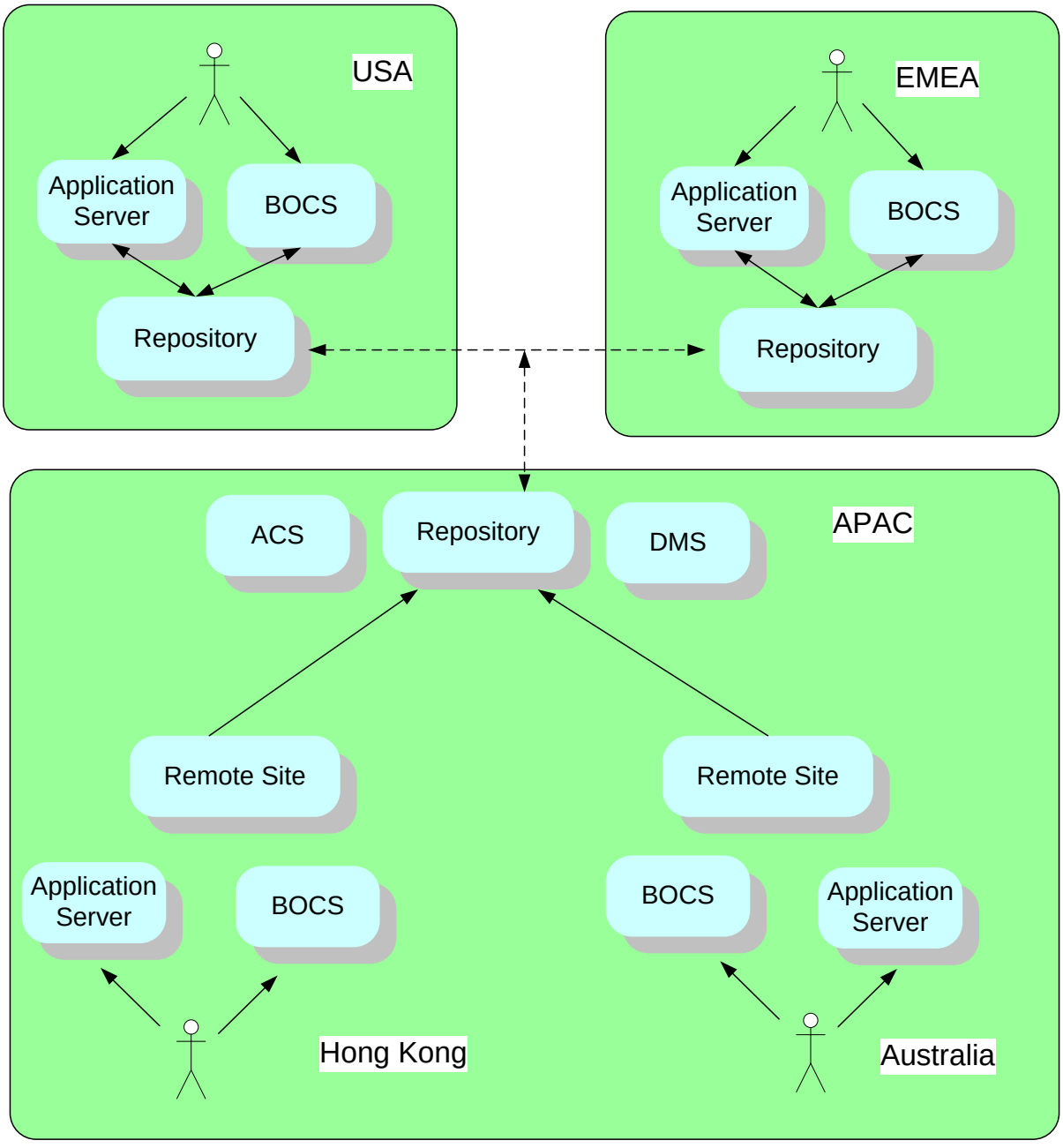

Figure 1 Architecture

Tools Used Repository Configuration– DA (Documentum Administrator)

3. OBJECT MODEL

3.1. Challenges

- Rationalization of object model that is reusable across globe
- Attributes which are mandatory in one region may be redundant in the other region
- Look up values of the attributes vary from region to region
- Language of look up values vary from region to region

3.2. Design consideration

Same object model can be used across all regions. This means choosing attributes which are very core to the operation of Document management system in different regions. These attributes are qualified to become part of object model.

Attributes or keys which are mandatory in one region are not sometimes mandatory in other regions. In such cases those attributes are part of object model and are non-mandatory. For e.g. PolicyID is mandatory for Italy but not any other country. This logic can be enforced using Composer.

- Open Documentum Project in Composer
- Import Object which needs to be modified from Repository
- Go to Artifacts → Types → select type which needs to be modified
- Click on the New button under Constraints in General tab
- In the Expression field enter docbasic expression like
  (CountryId = "IT" and PolicyId <>"") or (CountryId <> "IT")
- In Message field enter a text message relevant for users, such as “Policy ID cannot be blank”

  Type-level constraints are automatically enforced when documents are imported or metadata of existing documents is updated using Documentum applications such as WebTop or TaskSpace. Documents loaded using DFC- or DFS-based utilities can be validated after loading the object. This ensures that all object-level constraints are enforced even when documents are loaded from DFC or DFS code. Please note that the current version of MDO (6.6) does not support type-level validation.

Each region refers to the attribute in its native language. If language pack is installed, Documentum shows OOTB attributes in native language. It also provides a mechanism to display custom attributes in the native language using the following steps:

- Open a project in Composer.
- Go to Artifacts → Types → select localization type that needs to be generated.
- In Display tab, enter value for Type Label.
- Navigate to Attribute tab.
- Expand an attribute → Click on Application Interface Display.
- Enter value of the attribute in Label field in General tab.
- Right click on project and click Generate Localization Template.
- A directory called locales will be created. Inside locale directory, English is created by default.
- Make a copy of the complete en folder under the locales directory and rename the folder to the language locale you want to make available.
- For example displayed below two directories are manually created.
- Open the file generated for your type.
- Replace all text values with strings in the native language.

Figure 2 Locale Setting Using Composer

Attributes have lookup values associated with them. These values also need to be declared in the native language. In such scenarios, lookup values can be entered as a fixed list in Composer.

- Navigate to Attribute tab.
- Click on value mapping.

- Enter pair of Attribute value and display string.
- If the number of values that need to be converted are very high, it can cause YACC stack overflow error while installing DAR (Documentum Archive).
- These values can later be mapped to respective strings in native languages.
- This lookup is not only visible from import and property screen but also from advanced search screen.
- Lookup values are not applicable to all regions. In such cases, values are kept blank.

  Tools Used for Object Model – Composer, DQL (Documentum Query Language), DA RETENTION POLICY SERVICE

3.3. Challenges

- Each country has its own retention requirement in terms of retention period and retention classification.
- Retention policy needs to be changed if underlying attributes change.
- Same code to be used across countries to avoid country-specific deployment.
- Applying retention policy in bulk during migration.

3.4. Design consideration

Each country has its own retention requirement. Sometimes in countries such as the USA—where each state has its own set of laws—retention requirements can vary from state to state. For example, the retention period for California is different than for the rest of the country (see Table 1). The first step in effective retention policy design is creating a matrix of document type to retention period mapping. As shown in Table 1, document type attribute derives the document's base date while the period is determined by the location of the document. Some of the documents are retained permanently. These documents are classified as "Do not delete" documents. The following exercise can be accomplished with the help of the legal department.

Doc Type	Classification	Period	Trigger	Criteria
Agreement	Do not delete	Permanent	None	Document Type =Agreement
Policy	Delete	8 Year	Policy Expiration Date	Country<>USA
Claim	Delete	8 Year	Last Modified date	Country=USA and State=CA
Claim	Delete	7 Year	Last Modified date	Country=USA and State<> CA

Table 1 Retention Period Mapping

Retention policy is defined on a document using RPSA (Retention Policy Services Administrator). For defining retention policy, please refer to the RPSA guide. Each policy has a base date, retention period, and authority associated with it. Custom code is written to determine base date of the document. An extra attribute called custom base date is defined at the object level. This value is populated based on document type attribute. For example, if document type is policy, then custom base date attribute is populated with policy expiration date. Otherwise, it is populated with last modified date. Retention policy can be applied at either document level or folder level. Policies can be automatically inherited from folder level or can be applied using DFC code. Folder level policy inheritance cannot be used If all documents in same folder do not have same retention policy.

Retention policy is applied to a document when the document is imported in the repository. The document also needs to be evaluated for retention policy change when document metadata is modified. During document metadata update, one of the underlying criteria might change. For example, in the retention matrix shown above, if a document's state is updated from CA to NY, the document's retention policy will change. TBO (Type Based Object) becomes the ideal place to apply retention policy due to the reasons mentioned above. But retention policy of a document cannot be updated in TBO, since TBO cannot call intercepted DFC operation on itself. A daily Job is written to identify documents which are newly created and updated recently. These documents are evaluated for the retention

policy. Based on the criteria and trigger, retention policy is applied on the document. This code performs two activities:

1. Find the eligible retention period the document.
2. Apply retention policy.

The criteria mentioned in the matrix are stored in the xml configuration file. This xml stores:

- Attributes that determine retention classification
- Attributes that determine period
- Attributes that determine base criteria
- Retention period of the document

This insures that this code is reusable even though there is variation in retention start date, retention period, and retention criteria for document types.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <docsettings xsi:noNamespaceSchemaLocation="docsettings.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <DocRules>
    - <Country Code="US">
      - <DefaultDocRules>
        - <rule checkParent="Y" attributeList="doc_type,country,state" valueList="policy,USA,CA">
            <action useDocPolicy="retention_7_Years" />
          </rule>
        - <rule checkParent="Y" attributeList="doc_type,country" valueList="policy,USA">
            <action useDocPolicy="retention_8_Years" />
          </rule>
        </DefaultDocRules>
      - <ExceptionDocRules>
```

Figure 3 Sample xml

To avoid xml parsing for each document, .ser files can be generated from xml file. This serialized file can be used to get tag values directly without parsing them. Separate object type is defined for these xml files. These xml files are made accessible from DA. Administrator can define a new policy rule or modify an existing rule by editing the xml file. Underlying .ser file is generated using TBO call.

During initial rollout, a huge number of documents are migrated in the repository. Code written to apply retention policy should take care of both bulk as well daily loads of documents, so it should be able to handle both scenarios.

Tools Used for Retention policy development – Retention Policy Administrator Guide, Composer

4. SECURITY DESIGN

4.1. Challenges

- Security requirement can vary across countries.
- There can be exception rules to the security within the same country. A business entity in a country can be more restrictive than another.
- Flipping security based on the attribute changes.
- Application integrating with repository can have different security requirements.
- User provisioning: How to map users to groups?

4.2. Design consideration

A business entity can be broadly defined as a group of related folders. For example, when a claim is created, all folders related to the claim are created. A claim number can be called a business entity.

Security rules can vary across different countries. Hence, to match the security requirement of a country, separate ACLs (Access Control List) are created. ACLs can be identified based on the prefix. For example, usa_legal_accident_doc is ACL applied to documents related to accident business segment in the USA .

ACL can vary between entities within a country. Take for example, an adjuster who has “write” access on a claim document. Another user with same role as the adjuster may not have even read access on the same claim. This is called restrictive security. In open security, users get access to all documents across entities based on his/her role. In open security, ACLs are created during initial deployment and these same ACLs are used across entities. On the other hand, in the restricted security model, a new ACL is created when an entity is created. A dummy ACL is defined at entity level. Dummy ACLs are created during initial deployment time. Dummy ACL has dummy groups with actual permission. Just before applying security permission on the object, actual groups and ACLs are created. ACL creation takes place for each entity creation. In the following example a dummy ACL, d_italy_accident_$claim, is defined. It has a dummy group Italy_$claim. When a claim (78234566) is created in the repository, actual acl Italy_accident_claim_78234566 is created along with group Italy_78234566.

Dummy ACL Name	Dummy Group Name	Permission
d_italy_ accident_$claim	Italy_$claim	Delete
d_italy_ accident_$claim	dm_word	None
d_italy_ accident_$claim	dm_owner	Read
d_italy_ accident_$claim	Administrator	Delete

Table 2 Dummy ACL

Restrictive security creates a large number of groups and ACLs in the repository. Users are part of many groups. This can potentially impact system performance. The Ideal design goal for implementing security requirements is to meet the exact security requirement without creating a large number of ACLs and groups in the system. It involves understanding business use cases from a security perspective and deciding which security rules are absolute musts and which rules are nice to have.

An attribute change of a document can lead to change in the security. For example, a claim can become a restricted claim, changing ACLs on the claim documents. During each Document is evaluated for security change during each save. This is implemented in the security TBO. In some scenarios, users may have write permission on the document but may not have change permission. In this scenario, repository level security does not allow the user to switch ACL of the document. This issue can be resolved by dynamic group. A dynamic group is like normal groups except users are not part of the group by default when a repository session is created. All users are assigned memberships of this group. This group has highest permissions on all documents. In the exceptional scenarios described above, a user can get higher privilege by declaring it as part of dynamic group. Users can be added or removed from dynamic group using addDynamicGroup(“groupName”) and removeDynamicGroup(“groupName”) method of IDfSession object.

User authorization involves assigning users to proper groups. This activity is an ongoing activity in organizations since new users can be added, roles of existing users can change, or users can become inactive. Webtop and DA has user management node under administration. Users with administrator privilege can add users from groups and roles.

Security group within organizations can be created as administrator in repository. Using front end for user assignment is the best way of user management versus writing special code or script. Most of the time, user-to-group mapping information is stored outside of the content management system. If user provisioning information is replicated from an external application to Documentum repository using custom code, it creates dependency on the application. For example, whether a user belongs to adjuster or supervisor role is determined using user-group mapping in claims application. This code is not reusable in other regions since the user-group mapping mechanism in claims application may be not be standard across regions. Claims application may not even have a complete set of user-role mapping.

A document can inherit ACL from parent folder. This configuration can be done using DA by setting “Inherit Permission Set From” property to folder at content server level. This setting can be used if documents are classified on the basis of security. It means, all documents in a folder have the same ACL which is inherited from parent folder. If it is not possible to classify documents on the basis of security, then extra logic needs to be written. This logic will evaluate ACL of the document on the basis of metadata.

Tools Used for Development of Security –Composer, DQL,DA

5. TAXONOMY

5.1. Challenges

- Business processes vary in different countries for the same organization. This has an effect on taxonomy.
- Naming convention for same terms is different in different countries. Hence, folder names differ.
- Folders are named as per local language requirement.

5.2. Design consideration

Each country has a different way of carrying out business which affects on classification of documents and taxonomy. Folder structure for an entity can vary across different countries. For example, Coverage Information can be a folder under a claim in Italy but it may not be part of claim folder hierarchy in the USA. Country-specific differences in taxonomy are captured using folder templates. Each country has different folder templates. During entity creation, a corresponding template is chosen based on the country. For example, claim

folder templates in the USA do not have a Coverage Information folder while an Italian template for claim does. Either country name or country ID is essential information in the folder creation process.

Terms and folders names can differ between countries. For example, folder “Claim Information” in one country may be “Loss Information” in another country. While creating templates, country-specific terminologies are used. Folders in local language can be created using templates. For example, placement template in two countries can be compared in Figure 4 and 5. “0 Correspondence” folder in UK is named as “0 Corrispondenza – Placement” in Italy. Here, language and terminology both differ. Folder “05 Bind” is applicable for UK but not applicable in Italy. Such personalization by country eases folder navigation for local users.

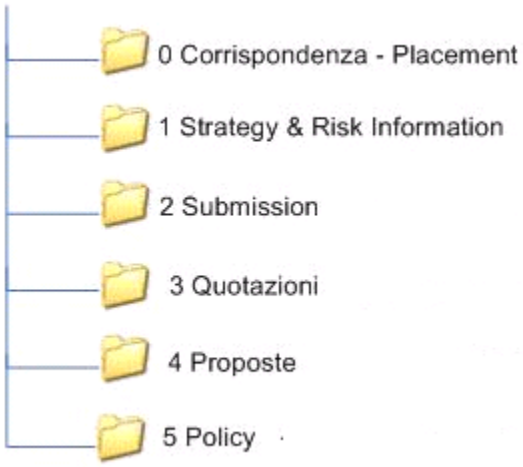

Figure 4 Italy folder structure

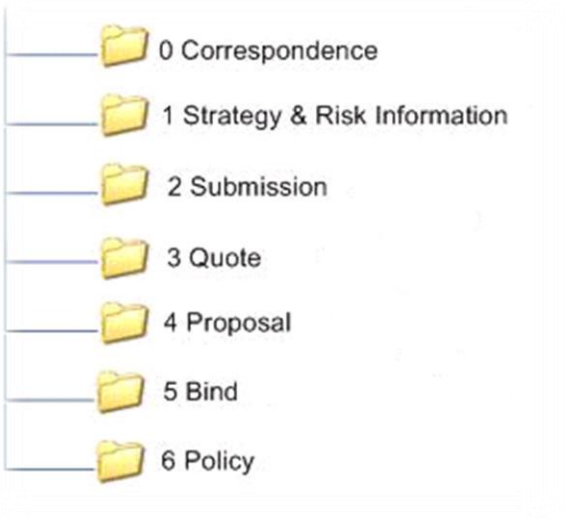

Figure 5 UK folder structure

Templates encapsulate security logic also. Dummy ACLs are associated with folders in a template. During the entity creation process, each folder is evaluated for dummy ACL. When a folder has dummy ACL, actual ACL and groups are created during folder creation.

Tools Used for Taxonomy Development– DQL scripts, Composer

6. USER INTERFACE

6.1. Challenges

- Localization of user interface on a country basis is an important part of UI (User Interface) development
- Many requirements are not met by OOTB (out of the box) WebTop. Documentum provides a framework called WDK (Web Development Kit) to custom WDK components. How to localize custom WDK components?
- Some of the functionalities are restricted to a few users. How to change user interface on the basis of user role?

6.2. Design consideration

Documentum supports localization of user interface in different languages. Release notes of corresponding client applications provides a detailed list of languages supported by Documentum, including clients such as WebTop, Taskspace, MDO, RPSA, and DA. Initial installation of Documentum has English local support. Language packs are deployed on both the Content Server and WebTop side. To deploy German and Italian language packs, the following steps are followed on Content Server:

- Uncomment entries for DE and IT in data_dictinary.ini file on Content Server
- Go to $DM_HOME/product/6.6/bin and execute following command
  Dmbasic –f dd_populate.ebs –e Entry_Point <docbase name>
  <id> <psw> data_dictionary.ini

List of available locales in repository can be checked using following DQL
Select dd_locales from dm_docbase_config

Language pack deployment steps for WebTop:

- Download and install DAR for particular locale

- Download the language pack zip file. Deploy these files to Webtop war.
- Edit app.xml and add require locale

Date fields are displayed in the local format in different countries. For example, USA date format is mm/dd/yyyy while the widely accepted date format in the UK is dd/mm/yyyy. Special handing is required while updating WDK date control in component class. While updating its values, the convertToClientTime function is used to ensure local date format conversion.

Parts of the user interface cannot be localized. These should be communicated clearly to the user. Table 3 describes different components of WebTop screen and their possible localization.

Sr No.	User Interface	Is localized?
1	Out of box Documentum menu bar	Yes
2	Custom menu bar	Yes
3	Out of box Documentum object Labels	Yes
4	Custom Object Labels	Yes
5	Out of box Documentum attribute label	Yes
6	Custom Attribute Label	Yes
7	Values assistance for an attribute	Yes
8	Value assistance while search	Yes
9	Folder Names Created using templates	Yes
10	Folder Name Created Manually	No
11	Document name	No
12	Document contents	No

Table 3 Language Translation Support

Some of the items mentioned above are supported by out of box Documentum installation, (such as 1, 3, and 5). For some items, configuration is required through composer (such as 4, 7, and 8). Features such as creation of folder involve creation using this template.

WDK framework supports localizations. When custom screens are developed, labels of controls can be externalized. WDK framework provides a mechanism to label those

templates in the native language by creating a property file for each locale. It follows the format below:

[bundle name] + ”Prop” + ”_”+ [local country] + ”_” [local variant]

For example

createClaimProp_us_en.properties (USA English)

createClaimProp_fr_fr.properties (France French)

Apart from control labels, error messages and warning messages need to be displayed in native language. It is important to take this fact into account while designing and coding of component class. These strings are being picked up from property files.

Some of these actions are restricted to a few users. For out of the box actions, presets are used to limit access to certain functionality using preset editor. For example, delete is restricted to admin users only. Users can be mapped to roles. Preset enables actions based on the role of a user to be hidden or displayed. This involves a configuration change only. Users who are part of dmc_wdk_presets_coordinator can create new presets. Preset editor is available under Administrator node in WebTop. Access to custom actions can be limited using the scope feature of the WDK framework.

Each region has one instance of WebTop, though WebTop code used across regions is the same. Each region supports multiple countries. There can be front-end variation based on the region. For example, the list of mandatory attributes for Policy creation may be different across regions. In such scenarios, a different jsp screen can be developed for each region. During run time, the corresponding jsp screen can be displayed. This can be accomplished using region variables. There are two ways to support this change feature:

1. Region variable can be read from the application server environment variable. Each application server type has its own way of storing environment variable.
2. Extending IApplicationListner class file provided by wdk framework. This code, executed during application startup, stores any variables read in the memory. Variables can be used to display a corresponding screen using Java script.

Tools Used for User Interface development – IDE like Eclipse, WDK

7. APPLICATION INTEGRATION

7.1. Challenges

- Which application interface to use (Native Application or SSO) into WebTop
- Deciding mode of communication (synchronous or asynchronous)
- User profiling: How to establish unique identity of users logging in from multiple applications
- Autofile: How to locate exact folder path of the document based on the metadata during upload
- How to identify documents uniquely across applications
- Which metadata need to be stored in Documentum system vs. application

7.2. Design consideration

Application integration layer—a combination of synchronous and asynchronous services that can be consumed at the enterprise level—is defined in order to integrate applications to document management functionality.

Before building application integration layer, it is essential to evaluate the requirements of each application. Ideally, application users should log in to a Documentum application such as WebTop to perform document management functionality. This results in less code development on the application integration layer. For ease of operation, users are provided with SSO into WebTop. WebTop supports SSO using standard tools such as Siteminder, Kerberos, RSA, Tivoli Access Manager, and so on. Users can be directly navigated to an exact folder based on their position in calling application. For example, users wanting to access correspondence related to claim they are working on will click on a link that takes the user to WebTop. This lands the user in the claim folder. This can be accomplished using custom WebTop component. Please find the sample URL below. A new component, ViewFolder, is created, taking the claim number as its i/p parameter.
http://test.webtop/webtop/component/ViewFolder?claimNo=6782224667

There are cases where directly logging into WebTop cannot be an option. This is especially true for applications which are exposed to the external user. Changing the user interface and provisioning a user inside Documentum is not an option.

Some operations are resource intensive while others can be less so. For example, setting a folder hierarchy for a new claim involves a lot of sub-tasks such as creation of groups, ACLs, and folder objects. Such resource-intensive services are potential candidates for asynchronous communication. Apart from this, asynchronous services are used where immediate response or no response is required. For example, bulk load of documents through nightly batch jobs. Asynchronous communication errors are tracked in the tables. Using this information, reason of failure is communicated to the application owner through the batch job.

Some operations are used synchronously as well asynchronously. It is essential to avoid duplicate code development. Core logic is implemented in POJO (Plain Old Java Object) classes. This logic is called from both synchronous and asynchronous application layer code. The same set of parameters are passed to synchronous and asynchronous communication except that the protocol-related parameters differ. For example, createFolder.java has a function named createFolder (pCredential credential, pFolder folder).

In this scenario, pCredential is a structure with elements such as user ID, password, and application code. pFolder is another structure which has elements such object type and country code.

```
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  - <xs:complexType name="CreateFolder">
    - <xs:sequence>
        <xs:element name="folder" type="pFolder" maxOccurs="unbounded" />
        <xs:element name="credential" type="pCredential" maxOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
```

Figure 6 Create Folder

This schema is used by both WebServices and MQ listener. For MQ listener, xml parser is used to convert incoming messages into Java structure before calling POJO methods. Applications can use the view component of WebTop for document view and download. Whereas, the application integration layer can be used for other transactions. In this scenario, inherent Documentum capability for content transfer is utilized.

Synchronous services such as search requires special handling. The maximum limit of rows that can be returned need to be determined during requirement gathering. Absence of such restriction can cause overload to both calling application and application integration layer.

Each application has its method of user management. Legacy and homegrown applications rarely use enterprise level identity management tools. User login name can be different for different applications. A common key such as email ID is used to identify a user uniquely across applications. Creating duplicate user ID for the same user must be avoided. If the same user logs in from a different application, he/she should be logged in as the same user in Documentum.

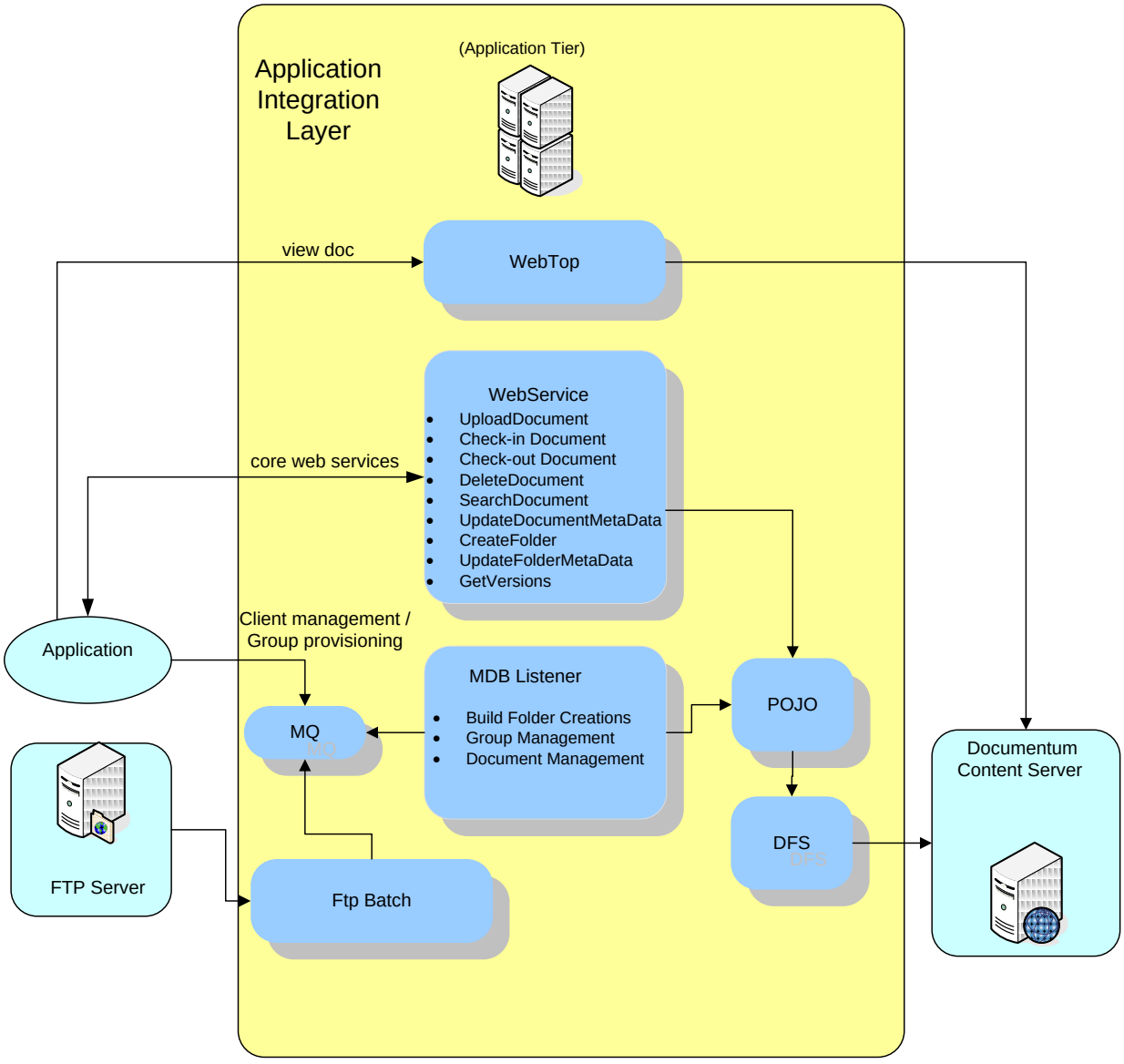

Figure 7 Application Integration Layer

The following points should be considered when designing i/p parameters for application integration services:

- A document's unique path can be determined using metadata fields. The application need not know the taxonomy maintained in the Documentum system. Application integration should resolve the correct path of the document based on past values. If the taxonomy changes, calling applications do not have to change their logic.
- A document can be uniquely indentified across all applications. Documents moved from one folder to another can be easily retrieved by application based on their unique metadata fields. Even if non-unique metadata of the document is updated, documents can be identified.
- A minimum of metadata fields should be stored in Documentum. Business process-related fields should not be part of the Documentum system. If there is change in the source system document fields, there is no need to propagate those changes in Documentum.

On top of the normal security model, some applications may want to restrict access to certain documents from other applications. When the same user is logged in to Documentum from different applications, they should get a different view. For example, if a user is logged in from the "Legal" application, he/she gets access to documents such as agreements. But when the same user logs in from the "Correspondence" application, he/she cannot update the agreement. Implementing only repository level security will not solve this problem; this is accomplished using application level security. Application level security acts on top of repository level security. Each application has its code.

During document upload, a_application_type is set to application code. When a session is created for a service call, the corresponding application code is set in the session. For example, a_application_type is set to application_code_legal for documents of type agreement. When a user logs in to repository through application integration code, a_application_type is set on the basis of calling application. If application integration code is called from Legal application, then a_application_type is set to application_code_legal using code sessionConfig.appendString("application_code","application_code_legal"). Hence, while accessing the repository from Legal application, users can access documents of type Agreement, When the value of a_application_code is not set, security of the document is

decided only by ACL of the document. If a_application_code is set, it evaluates the a_application_type first and then ACL on the Object.

Tools Used for Application Integration Development – IDE like Eclipse, JAXB , DFC

8. MIGRATION

8.1. Challenges

- Migration from different repositories
- Code reuse for different migration
- Completion migration before production rollout
- Completing migration without disrupting existing systems
- Ability to perform delta migration

8.2. Design consideration

An organization can have its data stored in different repositories. These repositories can vary in terms of technology, standards, and structure. The repository can be any system that stores documents and can vary from homegrown content management applications to third-party products. Instead of maintaining their own repository, some organizations store documents with external vendors. When all these factors are taken into consideration, the best option is to divide the migration process into different stages. This leads to better code reuse and easy code handling.

Migration is logically divided in three stages. These stages are based on standard E-T-L methodology.

- Stage I: Involves pure extraction of data from different sources. Each source has its own extraction process. The source system has its own folder structure, security, metadata, technology, and architecture. These factors decide if extraction code can be reused. Files are extracted to the staging area and metadata is temporarily loaded in the staging database.

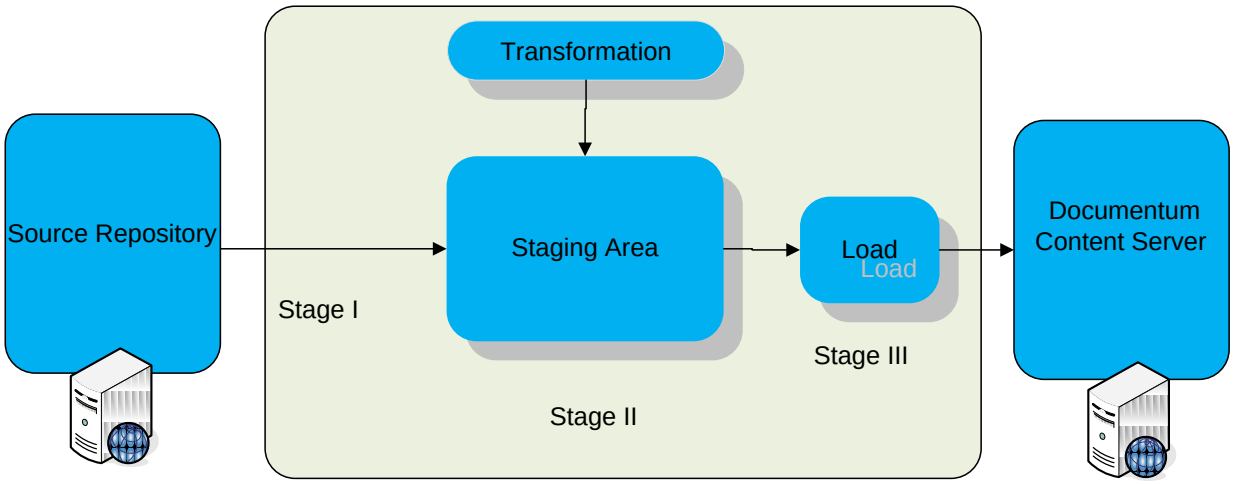

Figure 8 Migration Stages

- Stage II: In this phase, cleaning of metadata and translation of metadata is achieved. Sometimes, file conversion is required. After this stage, data is ready to be loaded in the repository.
- Stage III: Files are loaded in the repository in this phase. Apart from that, necessary folder creation, applying ACL to documents, and applying metadata is also taken care of in the phase.

All stages run in parallel. Daily load of document is decided by the extraction process and the import process throughput. If both source and destination systems are live, then there is a limit on number of threads that can run for source and destination. This also impacts migration throughput.

All the fields from different repositories should be rationalized before starting migration. This involves mapping of fields from different repositories in the document system. This exercise can be accomplished with the help of business users. In the sample mapping sheet (Table 4), ‘claim number’ is called ‘loss number’ and ‘claim number’ in different system.

Field Name	Repository 1	Repository 2
Claim number	Loss Number	Claim Number
Claimant Name	Name	First Party Name
Adjuster Name	Colleague Name	User Name

Table 4 Field Mapping

Unique identifier of a document is determined during design time. This identifier is used for

- Reconciliation of data once migration is done
- Error resolution
- Delta migration: Changes in source system are migrated to destination system once the initial migration is completed.

Migration of files is a time-consuming process. It is important to start migration before production rollout. Duration may be months depending on file size and number of files loaded. Migration of documents can be started much before mapping of metadata field is available. This process involves extracting files in the staging area. This saves a lot of time since this is most time-consuming part. Once mapping of metadata fields from source to destination is avialable, documents that are loaded in the repository can be upated with proper meatadata fields.

Tools Used for Migration Code development – IDE such as Eclipse, PLQL Developer/Toad

9. CONCLUSION

Global implementation of document management poses unique challenges in terms of repository design, application integration, and migration.

A lot of coding and rework can be avoided by careful planning, design consideration, and effective engagement of users. A repository design implemented for one country and region can be extended to multiple regions and countries with configuration changes. This leads to less turnaround time for implementation and extra savings for clients.