IBM Research Labs

IBM IRL,
Block 1,IIT,
Hauz Khas, New Delhi - 110016

Summer Internship Report

IBM E-Government Project

Ashish Gupta
B.Tech 4th Year
Computer Science and Engineering
IIT Delhi

Date: 25th Jul, 2001

Executive Summary

My two-month Industrial Internship Program work term was with the IBM Research Labs, New Delhi. I was involved in the area of e-Governance during my work term, all of which will be outlined in this report. This report will cover some background information on the projects I was involved in, as well as details on how the projects were developed.

There are two major projects that I had a significant role in.

The first project involved gaining a good understanding of an IBM product called Content Manager. My task was to study its capabilities from the point of view of a data management solution in the e-Gov system. My task was to

- ❑ Understand the product,
- ❑ How to develop applications using Content Manager,
- ❑ Create a prototype eGov Application on top of it.

One of the important achievements of this project was the development of a new Programming API over the Content Manger Programming API which provides many features and supports the eGov concept of “Middleware”. The major benefits of this API are:

- ❑ It greatly simplifies the task of developing applications on top of Content Manager.
- ❑ It provides additional functionality in Content Manager, which would be a requirement for eGov project.
- ❑ It supports a layered architecture by which new capabilities can be added to the Content Manager API by implementing the required functionality as layers above the AP

An application was finally developed using the above API to demonstrate its usefulness. It was a prototype solution to a real life problem known as Inner Line Permit, which occurs in eastern states of India like Arunachal Pradesh.

The second project was on designing and implementing Audit Trails in Distributed Database Environments such as that of eGov. Audit Trail can be described as electronic evidence that can be used to trace transactions to verify their validity and accuracy. It gathers data about activity in the system and analyzes it to discover security violations.
In this project we worked on some issues regarding Audit Trails in such an environment and some solutions were proposed.

I acquired many new technical skills throughout my work term. I acquired new knowledge in the area of Distributed Databases. I also brushed up my Visual C++ skills while making the Content Manager application. Then I got introduced to the area of research and how to approach it. Most importantly, the work experience was very good which included good fellowship, cooperative teamwork and accepting responsibilities.

Although I spent a lot of time learning new things, I found that I was well trained in certain areas that helped me substantially in my projects. Many programming skills that I used in my projects, such as programming style and design, were ones that I had acquired during my studies in Computing Science.
This report concludes with my overall impressions of my work experience as well as my opinion of the Industrial Internship Program in general.

TABLE OF CONTENTS

1 Acknowledgements

During my summer internship , the staff at IBM and persons guiding me were were very helpful and extended their valuable guidance and help whenever required for the projects which I worked on.

I am very thankful to my guide Dr. P. V. Kamesam for his invaluable guidance and advice during my Summer Internship.

I am thankful to Jaijit Bhattacarya for his guidance and friendly support during my stay at IRL.

I am also very greatful to Upendra Sharma (for great cooperation and help esp. in the 2nd project and my final presentation), Ajay Gupta (for helping in my final presentation, in developing the application and the API and documenting the APIs) and Harish Chauhan (who was always ready to extend any technical support at a short notice).

I also thank Dr. Manoj Kumar who helped us in the 2nd project on Audit Trails by sharing his vast experience and giving valuable direction to the project.

Overall , the above team made my stay at IRL an enjoyable one and I am greatful to them for making it so.

2 Introduction and Overview

2.1 IBM and e-Governance – Background Information

Governments are central players in the new economy. They set the climate for wealth creation. They can act as a deadening hand on change or be the catalyst for creativity. They can cause economic stagnation...or they can set a climate for growth."

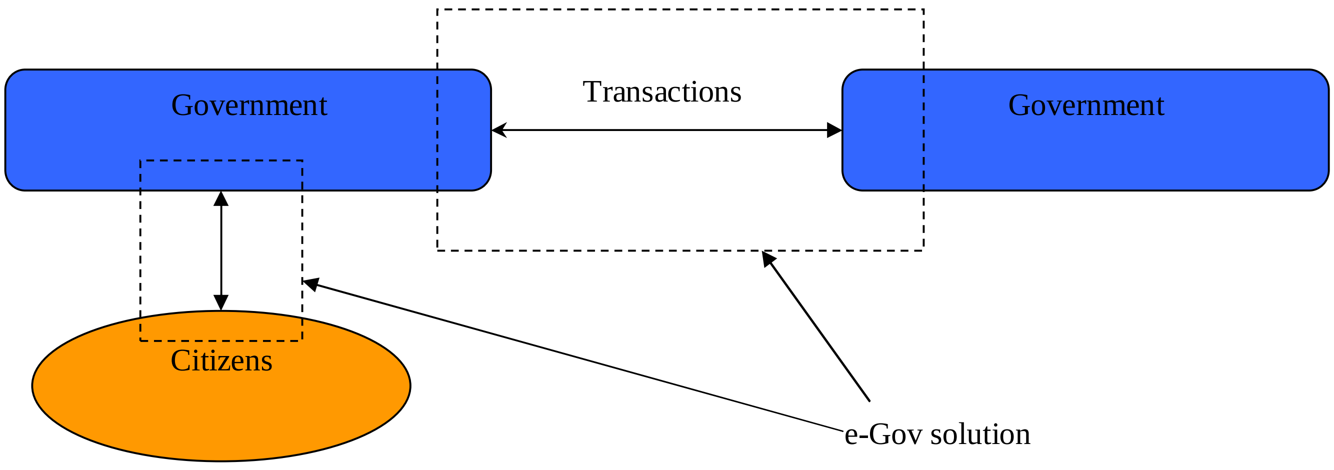
The Digital Economy

2.2 Introduction

Performance of government affects quality of life, economic growth and voter behavior. To that extent, it is essential to streamline the business of governance.
Technology has been a widely used tool to increase productivity and to remove inefficiencies in various industries in the late 20th century. However, the business of governance has been late in adopting technology solutions to increase its efficiency.
The primary reason for this anomaly is that there are numerous technology issues that need to be resolved before a true eGovernance solution can be adopted.

2.3 Need for E-governance

E-governance is a tool to provide efficient and effective governance. It increases the efficiency of governance by providing transparency and accountability and reducing the cost of governance.



2.4 Issues in implementing e-governance

Various governments, involved in the task of building a e-governance solution, are facing a three pronged problem. The first is how to select and entrust a solution provider to deliver any part of e-governance. While there are numerous solution providers in the market, it appears unlikely that any single vendor can offer the entire system/solution.

The second is to ensure that after each solution provider has created a solution, the various solutions should be integrable and should talk to each other. Since each solution provider may use its own technologies, data schemas and standards, integration will be a challenge.

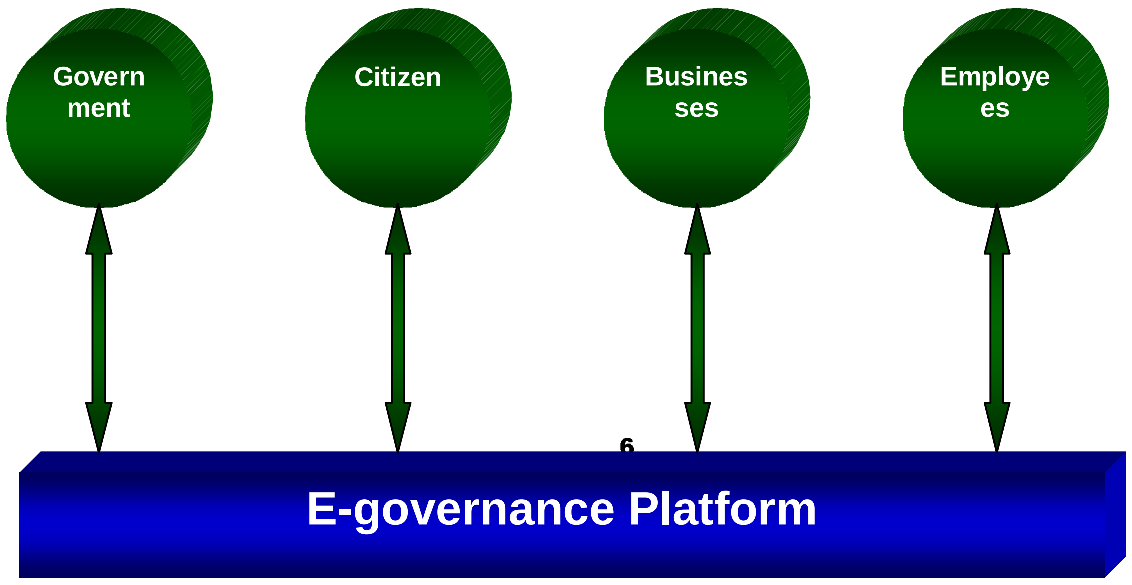
The third problem is to encourage the development of portable/ replicable solution, that can be reused in applications for other government agencies, States etc. The reasoning behind this is that, just as in businesses, around 85% of the processes are same across firms, within the same industry, it is expected that 85% of the processes should be similar across different governments. Thus, it should be possible to use the solutions developed for one government, in another government. Reusing the e-governance asset across different governments can substantially bring down the cost of governance.

2.5 Solution Approach

One possible option for addressing the above stated challenges is to adopt a model in which the solutions to be developed and deployed can adopt to any network topology, make use of a reusable middleware (which can impose some standardization). Such a middleware itself may make use of standard software components from the industry. Finally the applications themselves will be built by Total Solution Providers on top of the middleware.

2.6 Direction of Research at IRL

IRL is working on the design and prototype of the e-governance middleware, which will make it possible for governments to quickly deploy applications with the help of this middleware. These applications themselves may be to facilitate interdepartmental activities within the government, Citizen and employee interactions with the Government, or to to conduct the business of governance with businesses and enterprises.



2.7 Project Overview

2.7.1 Part-I: Content Manager

This part of the project involved gaining a good understanding of an IBM product called Content Manager. My task was to study its capabilities from the point of view of a data management solution in the e-Gov system. My task was to

- Understand the product,
- How to develop applications using Content Manager,
- Create a prototype eGov Application on top of it.

One of the important achievements of this project was the development of a new Programming API over the Content Manger Programming API, which provides many features and supports the eGov concept of "Middleware". The major benefits of this API are:

- It greatly simplifies the task of developing applications on top of Content Manager.
- It provides additional functionality in Content Manager, which would be a requirement for eGov project.
- It supports a layered architecture by which new capabilities can be added to the Content Manager by implementing them as layers above the API.

2.7.2 Part-II: Audit Trail on Distributed Database

An electronic audit trail is a form of evidence that can be used to trace transactions to verify their validity and accuracy. It gathers data about activity in the system and analyzes for the purpose of auditing the events by the application on the data. The current project focuses on the various issues involved in having an audit trail mechanishm for a distributed architecture like that of eGovernment as discussed above. It will discuss the placement of audit trail logic once the eGov architecture is in place and then other issues like its storage and security issues.

3 Project - I: Content Manager

3.1 Project Background and Scope

eGov is massive project in which data management plays a vital role. Since the number of applications running on the system will be large in number, data storage, retrieval and management will be a important issue. For this, a data management solution is required which will suit eGov requirements. An IBM product called Content Manager offers one possible solution.

Since the information generated by most of the eGov departments may not possess a structured format and thus there is an obvious need of some tool, which can mange this unstructured information in a efficient manner. Content Manager, an IBM trademark, offers a scalable solution for the same as it can manage all unstructured information in an efficient fashion.

3.2 Project's Rationale and Goals

My goals regarding Content Manager were the following:

1. To Install the Content Manager Product

2. To understand the Content Manager product and its proper configuration and settings

3. To understand its programming Interfaces and how to develop an application on top of it.

4. To develop an API on top of the Content Manager API which adds functionality to Content Manager as per the requirements of eGov.

5. To undertake an eGov application and develop it on top of Content Manager

3.3 Technical Details and Applicable Issues

3.3.1 Installing the Product

Content Manager uses IBM DB2 Universal Database as an adjunct to do its work. IBM DB2 Universal Database is an IBM product to take care of the enterprise database needs.

Prior to installing Content Manager, one needs to install Microsoft Visual C++ and IBM DB2 Universal Database and configure it properly. We faced some problems in Installing Content Manager and tried some times before giving up. We found it to be incompatible with DB2 version 7.1 so we tried DB2 Version 6.1. After that some additional configuration is required in the Windows NT regarding some system privileges. Content Manager uses the **Services** features of Windows NT and starts its modules as Services. Once you install Content Manager, you need to start these services to make Content Manager functional on the machine for that particular session.

3.3.2 Understanding Content Manager

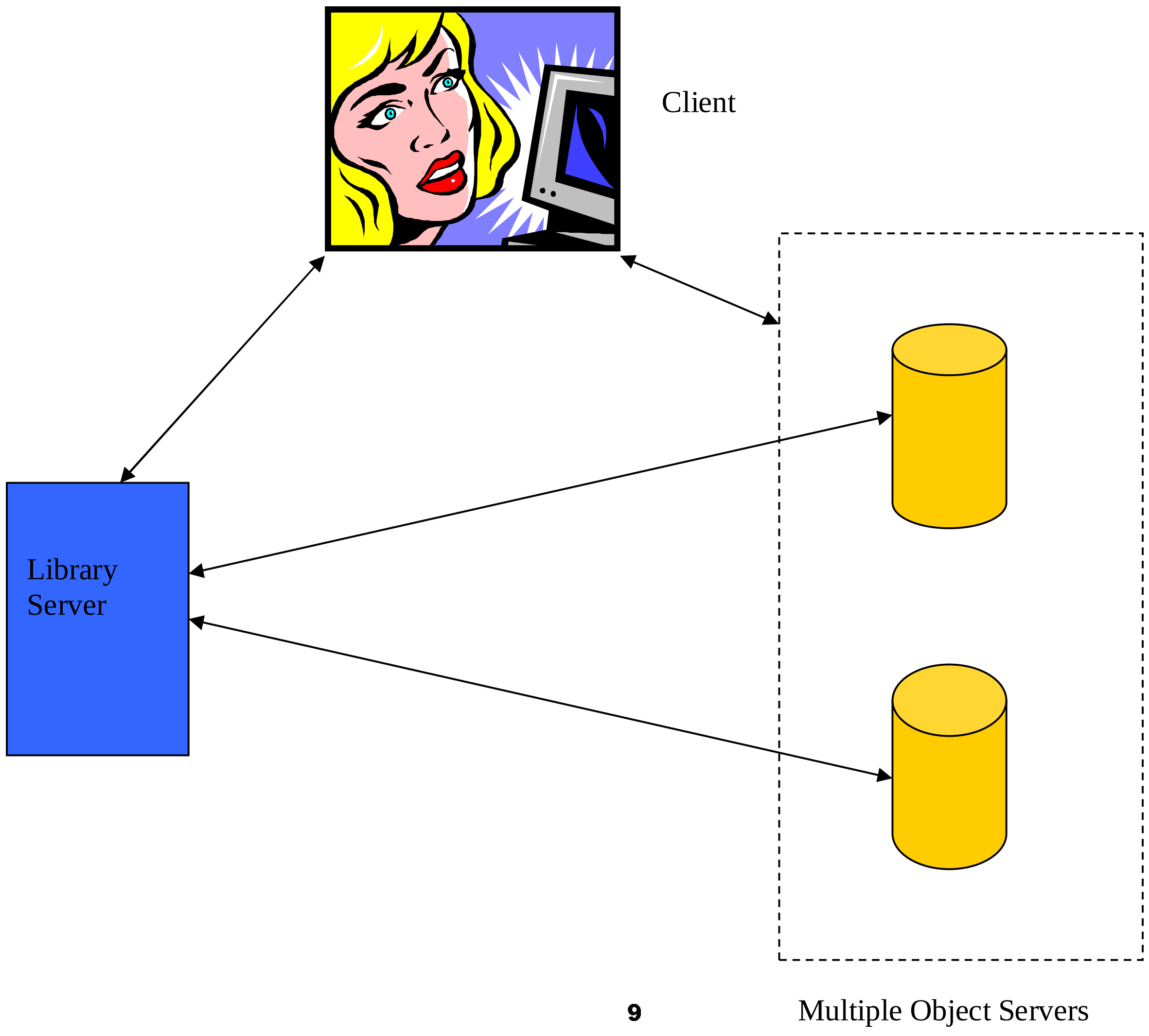
3.3.2.1 What is Content Manager ?

Simply put, Content Manager is a document management application, providing many sophisticated features and easily adaptable to a network solution for data access.

It can help you maximize the value of your information and multimedia assets. Regardless of the type of data (text documents, scanned images, audio, video, forms any binary object), it lets you store data on distributed servers and access it through a single point without knowing about the whereabouts of the data on the network.
This is ideal, where large amounts of data of heterogeneous nature is required to be accessed through a single client application where the data could actually be located across various servers.

For example a library with various branches across the country may place its books in popular formats e.g. *pdf* format and other media such as images, videos etc. across various servers which can be accessed through a central server using the Content Manager solution. Any user can then have a single view of the entire library using a web-front end.

3.3.3 Overview of the Architecture of Content Manager



The content manager consists of a client, library server and multiple object servers. The above figure shows the architecture for a typical Content Manager implementation. It consists of:

3.3.3.1 Client

This is the end user who will be accessing the data stored in the Content Manager implementation using Library Servers and Object Servers. Note that the client communicates with both Library Server as well as the Object Servers.
The connection between the client and the object servers has high bandwidth as compared to the connection between the client and the Library Server. This will become clear in the following sections.

3.3.3.2 Library Server

Library server, manages the Content Manager catalog information, locates stored objects using a variety of search technologies, provides secure access to the objects in the collection, and communicates with the object servers. A Content Manager system requires one library server, which can run on Windows NT$^{(R)}$, Windows 2000, AIX, or OS/390$^{(R)}$.
The library server uses a relational database to manage digital objects and provide data integrity by maintaining index information and controlling access to objects that are stored on the object servers. This relational database can be IBM DB2 Universal Database$^{(TM)}$ (DB2 UDB) or Oracle.

Currently, there is one Library Server per implementation. By an implementation, we mean an independent implementation of a Content Manager solution.

3.3.3.3 Object server

The object server is the repository for objects that are stored in the Content Manager system. Users store and retrieve objects from the object server, through requests that are routed by the library server. . There can be multiple Object Servers in an implementation.
The object server efficiently and automatically manages storage resources, based on the storage management entities (such as volumes) that are defined using the Content Manager system administration program A Content Manager system can have many object servers distributed across networks to provide convenient user access. Object servers run on AIX, Windows NT, Windows 2000, or OS/390.
A database on the object server contains data about the exact location of each object. The database can be either DB2$^{(R)}$ UDB or Oracle.

3.3.3.4 Advantages of the above Architecture

The above architecture provides the following advantages:

- Support for multiple, distributed object servers allows you to store digital objects close to the users who need to access them frequently. This support is especially important for delivery of large multimedia objects.

- Support for heterogeneous servers allows you to use the system for all kinds of data (including streamed data), while optimizing the processing of individual data types.

- Client communication through the library server ensures the integrity of data objects. A client can access objects only by requesting them through the library server. The library server database contains information such as object types, indexes of all stored objects, authorized users of the system, and access control lists for each object.

- Separation of client applications, indexes, and data makes applications independent of the data's location on the servers.

- The open architecture allows the intermixing of additional object servers on the same or different operating systems and supports scaling from one operating environment to another, as growth requires after implementation.

3.3.4 Features of Content Manager – Brief Summary

- Lets you store content regardless of format.
  (Unstructured data)E.g. text documents, scanned images, audio, video, forms any binary object
- Stores data on distributed servers and provides single point access.
- Provides many sophisticated features like
  - Access control
  - Storage management – Archiving , Purging , Migration
  - User management
  - Automated Workflow
  - Enterprise wide search from Internet or intranet clients.
  - Streaming audio and video.

3.3.5 The Beginners Guide to Content Manager

After going through the various manuals of Content Manager and filtering out the essential points, I wrote a document called **Beginners Guide to Content Manager** which covers the essential concepts of Content Manager including application development and includes appendices with useful reference information regarding Content Manager. This guide serves as a useful introduction to any person striving to understand Content Manager.

3.3.6 The New Content Manager API

As the major outcome of the first project, an API was developed over the existing APIs of Content manager, keeping in mind the following benefits:

The API

- Makes it very easy to perform operations on the CM Database
  The Content Manager API is difficult to learn and use to make actual applications and hundreds of lines may be required for simple operations such as insert , delete and search etc. The new API makes it very easy to write new applications on top of CM and one can do insert , delete etc. in just one line.

- Encapsulates the complexity of CM API
  The complexity of CM remains hidden from the user. Many parameters which take default values for normal data access scenarios are handled by our API and the developer doesn't need to delve into the details of the Content Manager programming interface.

- Object Oriented Approach allows easy integration into new apps
  The new API has been implemented following an Object Oriented Approach , thus allowing easy integration into new applications. One simply needs to declare a new object for each session with Content Manger, log in and start performing the data access.
  e.g.

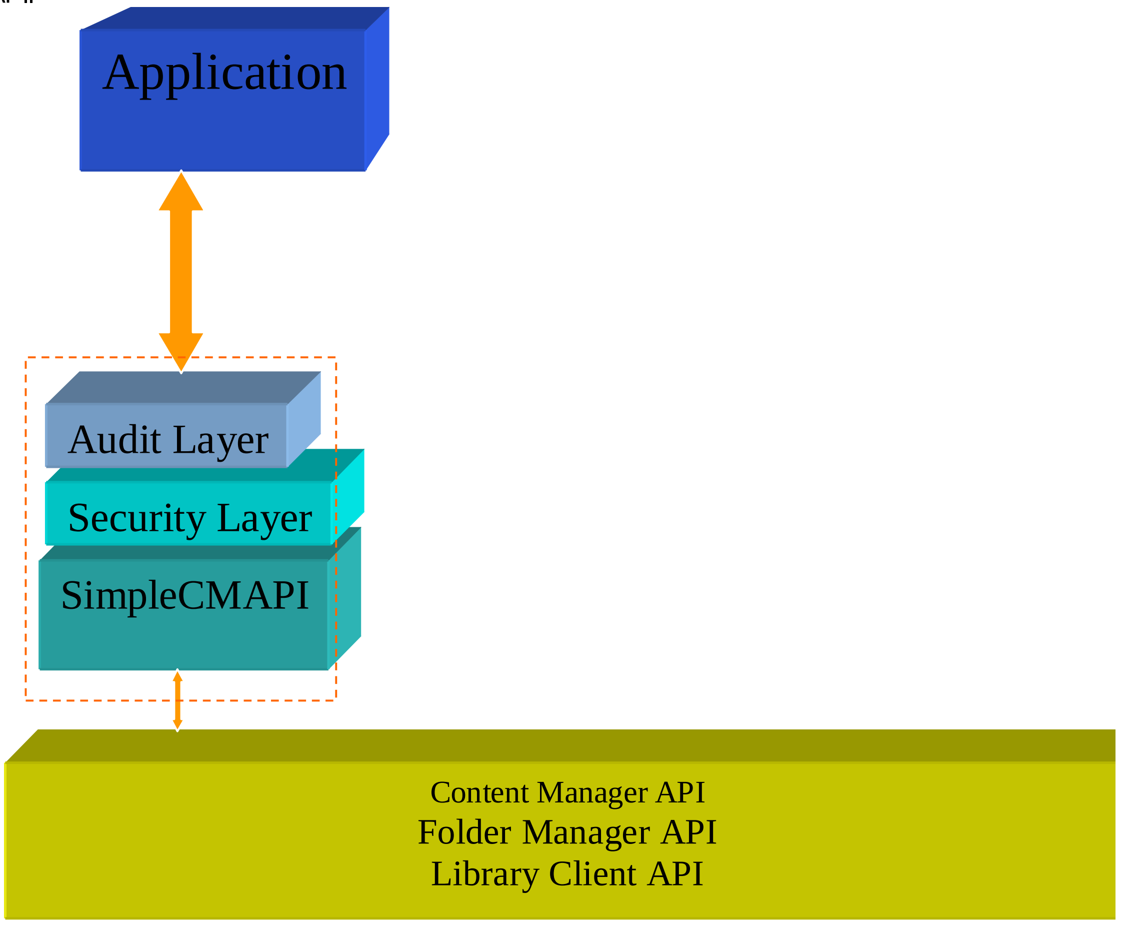
```
CSimpleCMAPI newsession;       // declare a new object
Newsession.login(login_id,login_password); // login

… start performing operations …
```

- Speeds up Application Development Time
  Since, the new API requires much less time to learn and coding , new applications can be developed very quickly.
  E.g. The first application including writing the API took us 4-5 days to finish.After writing the API , we developed another application of similar nature in just 4-5 hrs which is a significant improvement.

- Extensibility: Acts like a new layer on top of CM
  Since the new API is written on the top of Content Manager API, additional functionality can be easily added to the API to provide new features.

- New layers like Custom Access Control, Audit Layer can be added
  Some requirements of applications like those in eGov may not be completely or partially fulfilled by Content Manager. To take care of these requirements, one can simply implement the functionality required in a new layer on top of our API resulting in additional functionality. (See the figure below)
  E.g. In our prototype application to be discussed in the next section , we implemented new layers for audit trail and additional security requirements , demonstrating the above concept and how to go about implementing new functionality on top of the new API.



Layered Architecture for application development

The above figure shows the role of the new API in developing applications. The application talks directly our API and all its transcations pass through various layers like audit trail layer and security layer which can demand or implement the required functionality for the application.

3.3.7 The Inner Line Permit Application

An application was developed using the APIs described above. In order to develop an application a real life scenario of "inner line permit " was taken and a prototype solution application was developed over content manager.

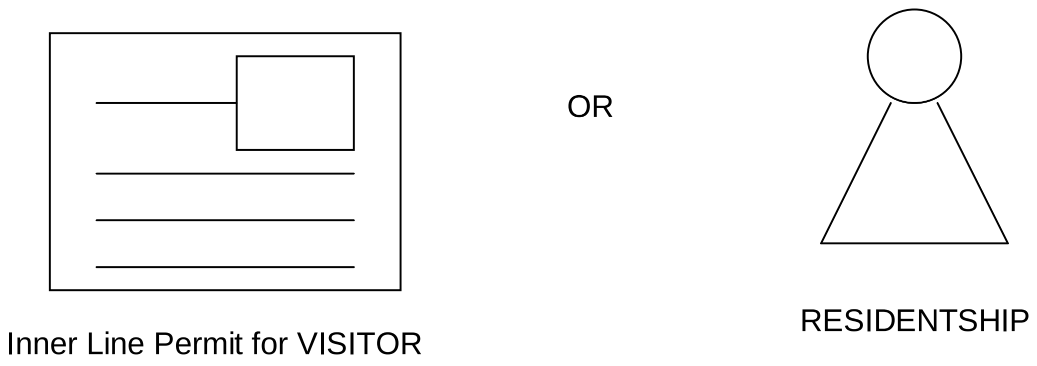
Description of inner line permit

All non-resident visitors wanting to visit Arunachal Pradesh need an Inner Line Permit to cross the border as a security requirement, which can be obtained from the Resident Commissioner and Liaison Officers, Govt. of Arunachal Pradesh. However, the citizens of Arunachal Pradesh don't need an Inner Line Permit to enter the state. This poses a problem at the entry checkpoints, as it may be difficult to make a distinction between a citizen of Arunachal Pradesh from a non-resident visitor.

- Inner line permit request is made to authorized office of the state government
- The request is entered in the system alongwith the scans of required documents
- Inner line permit is issued by authorized offices against the required documents
- At the entry check points the security personnel verifies whether a person entering the state is a resident or a non-resident using the information system
- For non-residents, validity of the inner-line permit is checked using the system
- The person entering the state (both resident and non-resident) are identified using identification marks entered in the system as well as the photographs entered in the system during the issuance of inner-line permit/ residentship.

Thus, a convenient system, which permits the state government to access the residentship database of the state with identification information would greatly assist in managing the flow of people into the sensitive state.

Arunachal Pradesh



Requirements for entering Arunachal Pradesh

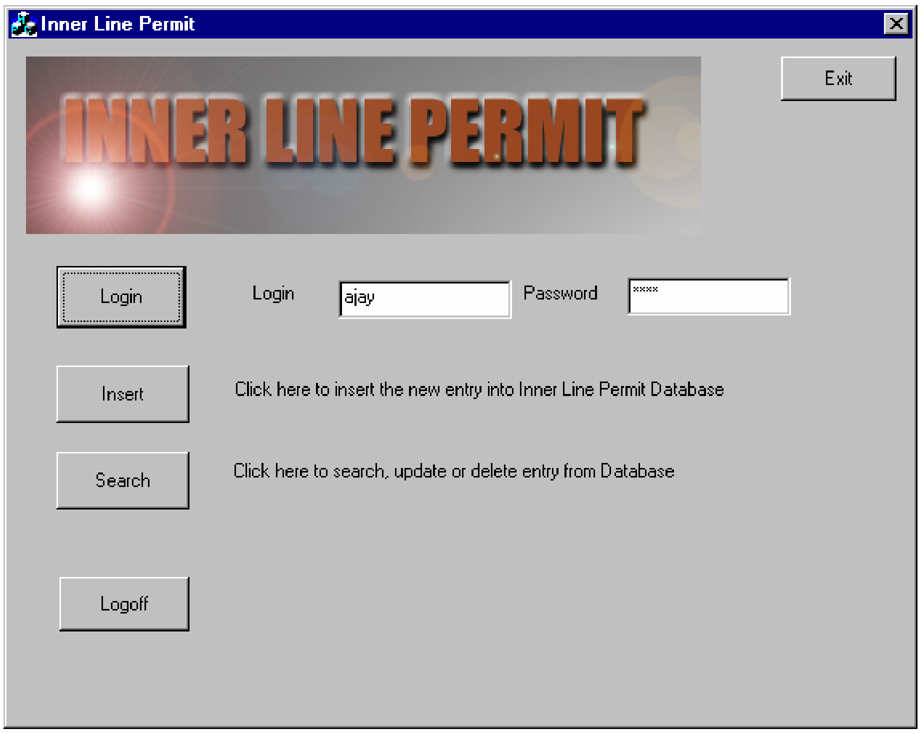
Explanation of the Solution provided by the application:

The application consists of 2 databases, which are:

- citizen databse and
- inner line permit database.

The application, as shown in the screen shots, provides the necessary funcitonality as requested by the situation. An efficient usage of the above mentioned APIs has also been shown by providing extra security and auditing facilities in the application.

Some screen shots of the inner line permit application

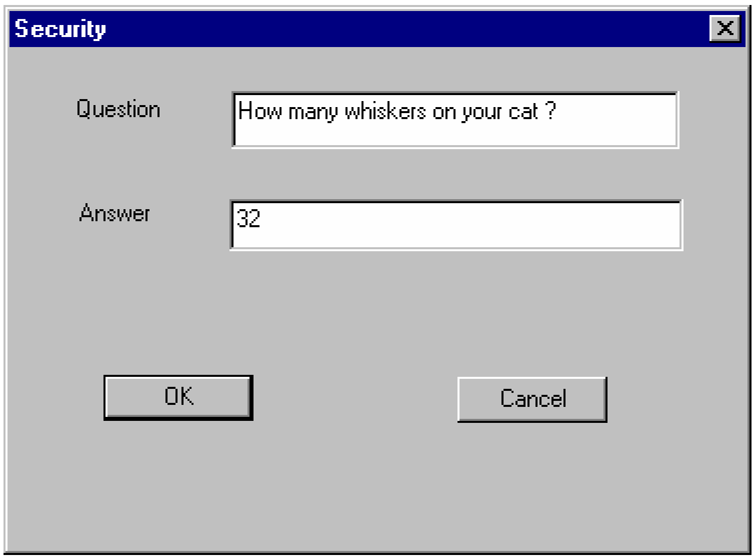


Login: ajay
Password: ****

Click here to insert the new entry into Inner Line Permit Database

Click here to search, update or delete entry from Database

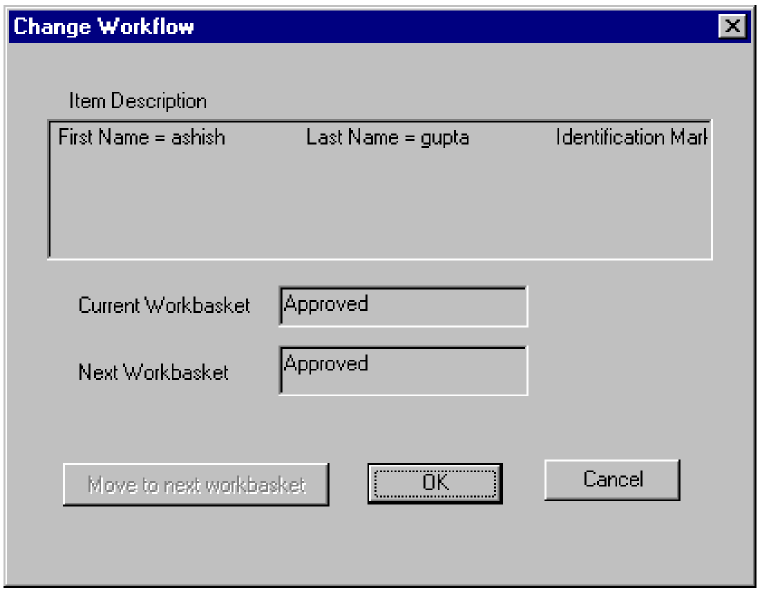
The opening screen of the Inner Line Permit Database application



Question: How many whiskers on your cat ?

Answer: 32

Demonstration of the security layer above besides the Content Manager security



Item Description

First Name = ashish Last Name = gupta Identification Marl

Current Workbasket: Approved

Next Workbasket: Approved

Demonstration of Workflow capabilities in the application

3.3.8 Future Work

Content Manager fulfills some of the requirements of eGoverment. However, it is not the complete solution for eGov Data Management needs but in can serve as the part of the solution. More extensive database applications need to be explored which can talk with databases of multiple type (federated databases). There are some other options being explored namely:

Data Joiner

A product from IBM, it provides a single point access/view to various databases (heteregenous). This can be considered as one of the solution for data management in eGOV.

EIP : Enterprise Information Portal

The IBM Enterprise Information Portal (EIP) is a comprehensive portal environment that can bring all the enterprise resources to a workstation desktop. EIP can help maximize the value of information and multimedia assets by connecting disparate content servers through a single client. The EIP client provides search capabilities that enable users to quickly and concurrently access all connected servers. EIP also allows one to customize applications specifically tailored for the enterprise. With the EIP toolkit, application programmers can quickly deploy custom-made desktop and Web-based applications.

EIP supports the concept of data connectors by which it can connect to databases from various sources including DB2, Oracle, and Content Manager and is capable of performing a federated search over the various databases.

It also provides ease of developing Web based thin clients for information retrieval in eGov. This can also be considered a solution for eGov and since it can also integrate with Content Manager , both can participate in the final Data Management solution.

3.3.9 Summary

Content Manager is a good solution for managing unstructred data arising from various applications running in eGov. It has a distributed architecture and provides many features conducive to an eGov implementation.

In the above project, a new API was developed over the Content Manager API to provide simplicity and extensibility catering to some eGov requirements which are absent in Content Manager. Using the above API, an application was also developed called Inner Line Permit which provided a prototype solution for a real life problem.

4 Project – II: Audit Trail on Distribute Databases

4.1 Project Background and Scope

As the requirements for having online access to enterprise data are escalating, so are the related security and privacy issues. Moreover, all operations performed on such information may need to be tracked and documented in the system for future reference and security reasons. E-government is one such application, which would require security and audit trails.

4.1.1 eGovernment from audit trail's perspective

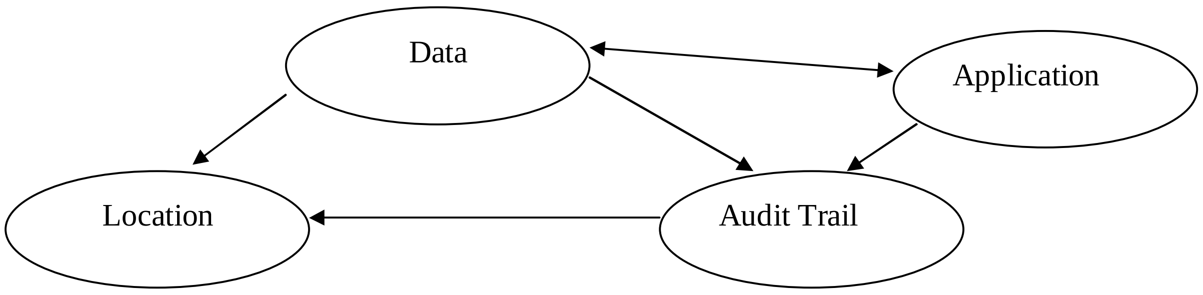
Generally, current e-government system designs aim to provide one of four service levels:

- Displaying information about agencies or aspect of the government.
- Second-level services provide simple two-way communication capabilities, usually for uncomplicated types of data collection such as registering comments.
- Third-level services facilitate complex transactions that may involve intra-governmental work-flows and legally binding procedures. Examples of such services include voter and motor vehicle registration.
- Fourth-level services seek to integrate a wide range of services across a whole government administration, as characterized by the many emerging government portals. The eCitizen portal, developed by the government of Singapore, offers a prime example of this system type.

In order to provide the above-mentioned service levels, governments are facing a three pronged problem: 1) out of many competitors in the market, whom to entrust for providing a complete solution; none of them have an experience. 2) Interoperability of various solutions for interchange of information and data. 3) Containment of cost by developing a portable/replicable solution. One of the feasible options for all the above problems is to provide a middleware and let the total solution providers build applications on top of it, which would cater to the needs of the specific departments of government. One of the important requirements, of this kind of solution, will be a robust audit trail mechanism, which will audit different events depending on the policy decided by the government or some other policy maker.

4.1.2 What is audit trail ?

An electronic audit trail is a form of evidence that can be used to trace transactions to verify their validity and accuracy. It gathers data about activity in the system and analyzes for the purpose of auditing the events by the application on the data. Hence audit trail can be defined as a series of records of the computer events relating to the operating system, application or to user. It can be used for any of the following purposes:
Individual Accountability: Users actions are recorded allowing users to be accountable to their actions.
Reconstruction Events: Audit trails can be used to reconstruct the events in case the need arises.
Problem Monitoring: Audit trails can be used to monitor and correct the problems, at the OS, system, or application level, as and when they are encountered.
Intrusion Detection: Important security issues like breaking into the system, unauthorized access, and/or any other security violation can be detected by maintaining proper audit logs of the processes which need to be monitored.

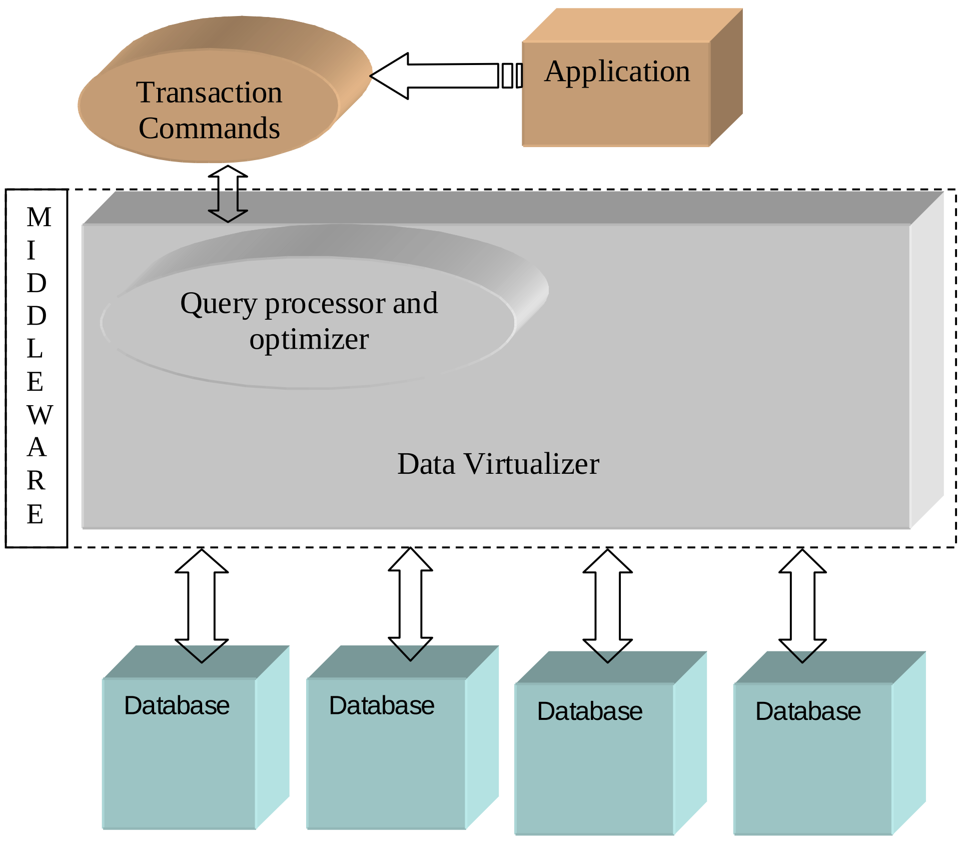


Any auditing system constitutes of two parts 1) *Audit Data Collector* and 2) *Audit Data* Analyzer. The operations of Audit Data Collector can be classified into any of the following three classes, as shown in the figure above, namely 1) storage of audit trail 2) audit of data access and changes 3) audit of system, application, or user events.

In this rough draft we will be addressing the issues involved in Audit Data Collection, which is composed of three above-mentioned sub-issues.

4.2 Project's Rationale and Goals

A small block diagramatic view of the eGov meiddleware is as shown below.



Data virtualizer is an important element in the eGov middleware. I does the task of abstracting the physical location of data and provides a single logical view of the backend databases.

Since security and auditing is a mandatory feature of any government operation and thus while providing a solution to government one has to give serious consideration to both auditing and security. In this part of my project I've concentrated on the auditing system which could possibly be a part of the eGov middleware. The auditing system which was studied during this summer training keeps in mind the distributed nature of the eGov solution.

In the current work we have studied the problem of auditing the entire eGov solution distributed across the network and to maintaing, efficiently and effectively, a complete audit of all the operations carried out by the middleware and providing role based access control features to the audit database as well.

4.3 Technical Details and Applicable Issues

Problem Description

Where should we place the Audit Trail Component in the e-government architecture (as shown above) is an important question because audit trail will be an important requirement in e-government and will be used for various services including security services etc.

4.3.1 Architectural choices

Possible options are

1. *Application*

   One can place the audit trail component in the application itself. This means the application developer will be responsible for figuring out the audit trail logic as per the requirements and implementing it in their applications. Or he may use Audit Trail modules to implement the Audit Trail functionality in his application.

2. *In the Middleware, above Data Abstraction Unit (DAU)*

   In this scenario, the audit trail component will intercept all the transactions taking place between the applications and the DAU. It will then take care of the audit trail requirements independently of the application and store them according to the

3. *In the Component Databases*

   We can also place the audit trail components in each of the databases in the system. Thus each local database will be responsible for keeping the audit trails of all the operations performed on the data. This is a popular practice and can be implemented using database triggers.

Before designing the above mentioned e-government system, one needs to know how the audit trail architecture will look like to take care of required issues in the middleware.
Hence we need to evaluate these options with reference to above-mentioned distributed system and choose the best possible architecture.

Let us look at the issues, which help us to evaluate these options.

4.4 Requirements for Audit Trail

In a massive project as e-government, the requirements of Audit Trails are important to be laid out. The final audit trail implementation must follow these requirements.

Some important issues are:

- Security Issues
  - Risking security at the hands of applications

    In the first option, if we place the audit trail component in the application itself, then audit trail becomes application dependent. This involves risk to the e-government system, as the application developers may not adhere to the requirements of audit trail.
  - Tamper proofing of Audit Trail

    Audit Trail represents sensitive data, and the auditor must be ensured that the audit trail data has not been tampered.
  - Access Control to Audit Trail

    We may need to provide access to audit trail in hierarchical manner.
- Implementation Issues
  - Application Complexity

    With regards to implementation of Audit Trail component, we also want simplicity in application development.
  - Database design Complexity

    In a distributed database environment, simplicity in database design will be preferred.
- Audit Trail Transparency to the application developers.

  It is preferred that the audit trail mechanism is kept transparent for the application developers, as it will greatly simplify the task of application development.
- Consistency of Audit Trail across the entire system

  This is a very important requirement, as the e-government system will be monolithic in nature, hence audit trails must be stored in a uniform fashion in terms of format, architecture, access etc.

- Ease of Audit Policy Management.

  Concept of Audit Policy management is novel in e-government and will provide lot of simplicity in managing the Audit Trail functionality of e-government. With the help of Audit Policy Manager, non-technical staff will be able to edit the settings of the audit trail component and hence modify its functionality according to the requirements.
- Modularity of Audit Trail Component

This is desirable because it permits us to easily change and manage audit trail mechanism in the system. If a change is required in the Audit Trail mechanism at a later stage, one simply has to modify the Audit Trail Module w/o changing other components in the system.

The following table evaluates the issue of placement of audit trail.

Issues	ATC in Application	ATC above DAU	ATC in databases
Security issues			
Risking security at the hands of applications	The application provider may not fulfill the requirements of Audit Trail as required by the e-government system.	Such a problem will not be encountered because it is centralized.	Such a problem will not exist, however, there can be risk of implementation of the audit trail at the

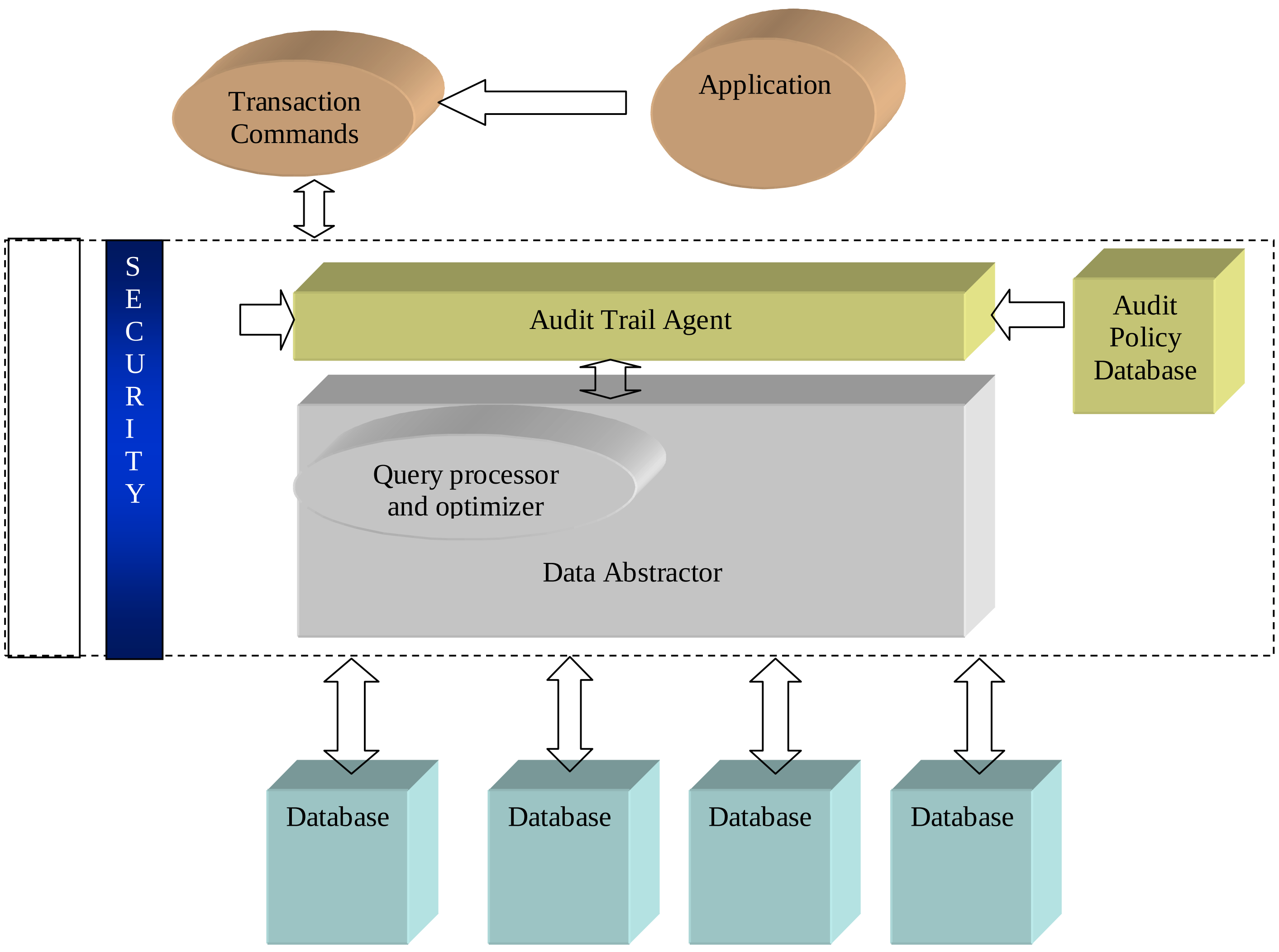
			database level for this again is decentralized.
Tamper proofing of Audit Trail	If left at the hands of application developer then again the situation is similar because the nature of generation of audit data and its maintenance is decentralized.	Because this is centralized, hence the generation and maintenance of the audit data is not an issue as these are the bodies, which maintain all the policies.	Tamper proofing is risked at the hands of the database designers and maintainers.
Access Control to Audit Trail	Since this is left at the hands application developers this is also at the risk of not getting implementing to the level of expected requirements.	No such issue is there, as this will be implemented n shipped by a responsible organization.	Similar to that in the case of application developer.
Application Complexity	Increases and thus not preferable.	Is decreased, as they are relieved of it.	Is decreased, as they are relieved of it.
Database design complexity	Unaffected	Database design is simplified, as the designer is relieved of the responsibility of it.	Is increased as they have to take care of the necessity of meeting the requirements laid down by the government.
Audit Trail Transparency to the application developers	Not present	Present	Present
Consistency of Audit Trail across the entire system	Difficult to maintain and thus not preferable	One of the simple features of the middleware design.	Difficult to maintain and thus not preferable
Ease of Audit Policy Management	Difficult	Easy	Difficult

4.5 Solution

In the previous subsection, we discussed the various pros and cons of different options of keeping the audit trail component. From the discussion, we can conclude that placing the audit trail component in the Middleware, as shown in the figure below, is the best option.

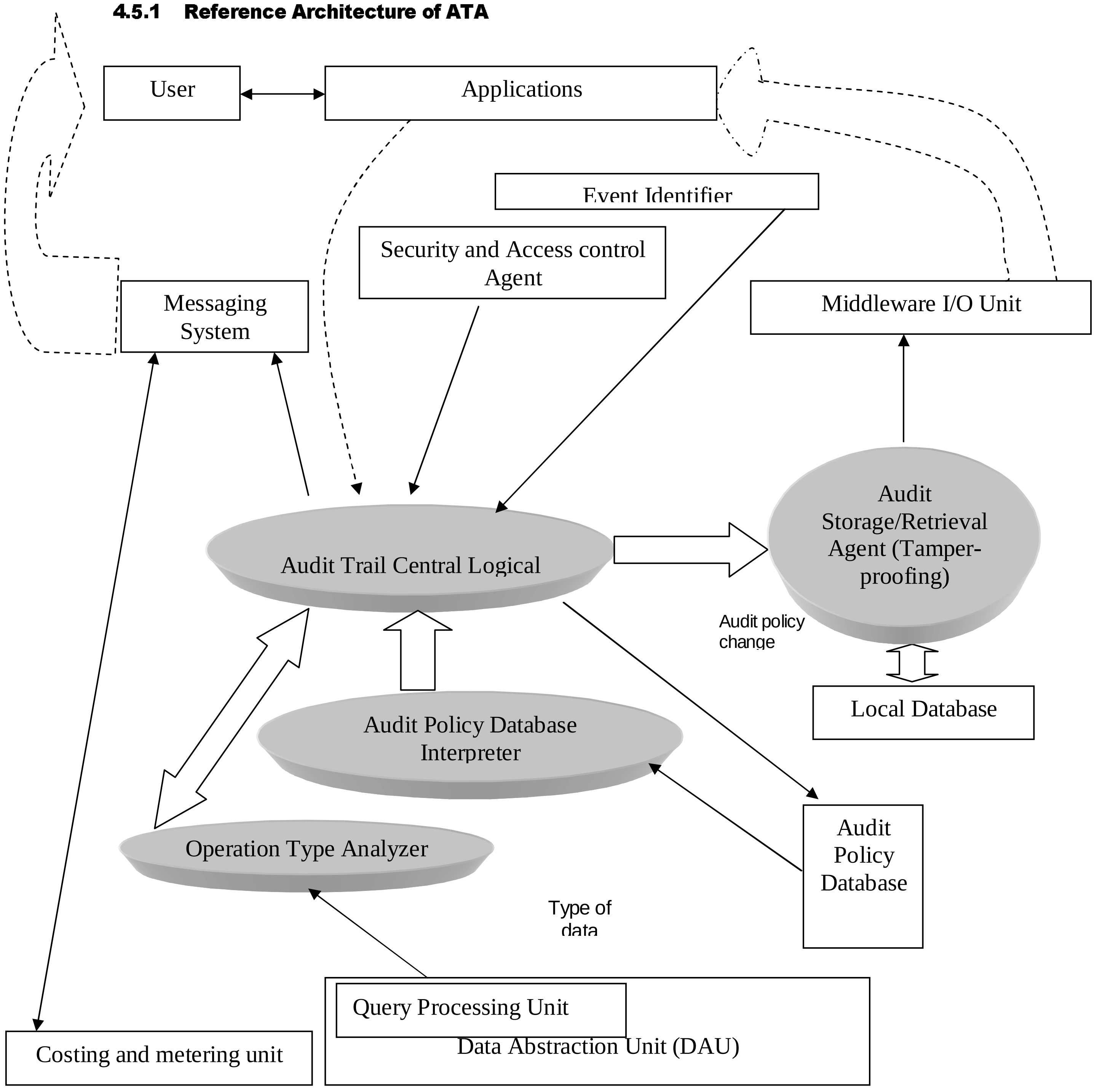
As shown, the audit trail agent (ATA) has been placed in the middleware above the DAU. The idea is to operate the ATA at the middleware level and more independent of the other components of the architecture namely the applications and the databases.

This approach offers many new perspectives of looking at the design of Audit Trail.
It makes sense to pull out the audit trail component and convert it into an independent module called the Audit Trail Agent, which will be responsible for the Audit Trail Mechanism in the e-government system. In the architecture shown below, Audit Trail Agent lies between the applications and the DAU and intercepts all the transactions and other communications, which the application is issuing to the Middleware. Thus Audit Trail agent is actually interested in observing the "actions" of the application and records them for later analysis. It is important to note here that in the above architecture, the audit trail mechanism is not bound to the data storage/retrieval mechanism but exists independent from it. This is important, as the Audit Trail is a separate application with not much reason to bind it to the databases.



In the next section we will discuss the architecture of ATA and the communication of its components with the other components of Middleware.

4.5.1 Reference Architecture of ATA


User
Applications
Event Identifier
Security and Access control Agent
Messaging System
Middleware I/O Unit
Audit Trail Central Logical
Audit Storage/Retrieval Agent (Tamper-proofing)
Audit policy change
Local Database
Audit Policy Database Interpreter
Operation Type Analyzer
Audit Policy Database
Type of data
Query Processing Unit
Data Abstraction Unit (DAU)
Costing and metering unit

4.6 Future Work

We plan to implement the complete ATA in a working, distributed system, which could be EIP or Content Manager and sort out further technical issues involved with it which include:

- Storage of Audit Trails
- Security Mechanisms

4.7 Summary

A detailed review of the existing state of art has been carried out for audit trail systems in various enterprise and non-enterprise solutions. A intutive architecture and logical placement of the ATA has been proposed which optimizes across maximum number of enchmark issues.

5 Learning From the Internship Program

The Internship program was quite beneficial for me. It helped me in improving my various technical skills and enhanced my knowledge in new areas.

- I gained new knowledge in the area of Databases and Distributed Databases, the various issues involved and mechanisms in these systems etc.
- By studying Content Manager, I also learnt that how database products function and what are the various issues one need to be aware of while looking for Data Management solutions.
- E-Gov is an emerging field and I got some insight into a how a new field looks like when in the initial stage and what are the various things which need to be done initially like requirement analysis, survey of existing solutions etc.
- I was working on a research topic for the first time and I got introduced to this important area of research and methods employed in conducting research.
- I brushed up my knowledge of Visual C++ and MFC (Microsoft Foundation Classes), as it was required to develop applications on top of Content Manager

Work Experience
My internship was quite satisfactory in terms of work environment. The team I worked with was very friendly and helped me a lot in all my problems. New experiences include
- Teamwork

In these projects 4-5 people worked together thus providing enough opportunity for proper teamwork and coordination. This was a good experience for me as the team was very cooperative and understanding.

- Responsibility and keeping commitments

The importance of honoring commitments and time of others was an important thing, which I learnt as a summer Intern. Especially, while working as a team it is very important to keep these points in mind.

6 Applying My University Skills

My education at IIT was very helpful in my Internship. The programming skills which I developed in IIT were very helpful in developing the API and the Inner Line Permit application. The courses CS120 (Computer Programming) and CS130 (Data structures and Algorithms) were especially helpful in this regard.

Course on File Systems (CS311) was also very helpful as it covered some database concepts which were required in understanding Content Manager and the database concepts in the second project as well.

HU320 (Technical communications) proved to be very helpful in my final presentation and the various documents which I had to write during my Internship. I applied the principles learnt in the course in my final presentation which was well appreciated by colleagues.

7 Appendices

7.1 Appendix A : Requirements for eGov data management

Sr No.	Requirement	Content Manager Solution
1.	Audit Trail: Audit trail requirements should be parameterizable by application. This should include list of columns for which audit trails needs to be maintained Time of change Who made the change Reason of change Short description of change number of modifications that need to be tracked duration for which the audit trail needs to be maintained track changes by employee track details of records viewed by user Audit trail should be maintained for both Object Server and the metadata stored in the library server	CM does provide a feature called history. Checking in progress.
2.	Access control by application Access to all data owned by an application needs to be limited to the owner application itself and to any other application to which the administrator of the owner application has given the permission. This control is to be given at a table level, column level and row level. Even application developers should not be able to access data of tables for which t does not have explicit rights	CM has well defined security mechanisms. Access can be restricted for each item. Users can also be restricted to do only certain operations on the data.
3.	Security : Authentication & Access control including protection against external attacks on the system Non-repudiation Integration with certification authorities Integration payment gateways	
4.	Scalability : Large amounts of data needs to be handled for storage Query support Response time as size of database increases Support for distributed storage servers will be required Distributed Indexing servers for query needs may also be needed	CM allows for one Library Server in one implementation, which catalogs all the data available on multiple object servers. This may be a big limitation. See below. (Needs to be clarified from CM developers)
5.	Record Management features :	CM has well defined mechanisms for migration

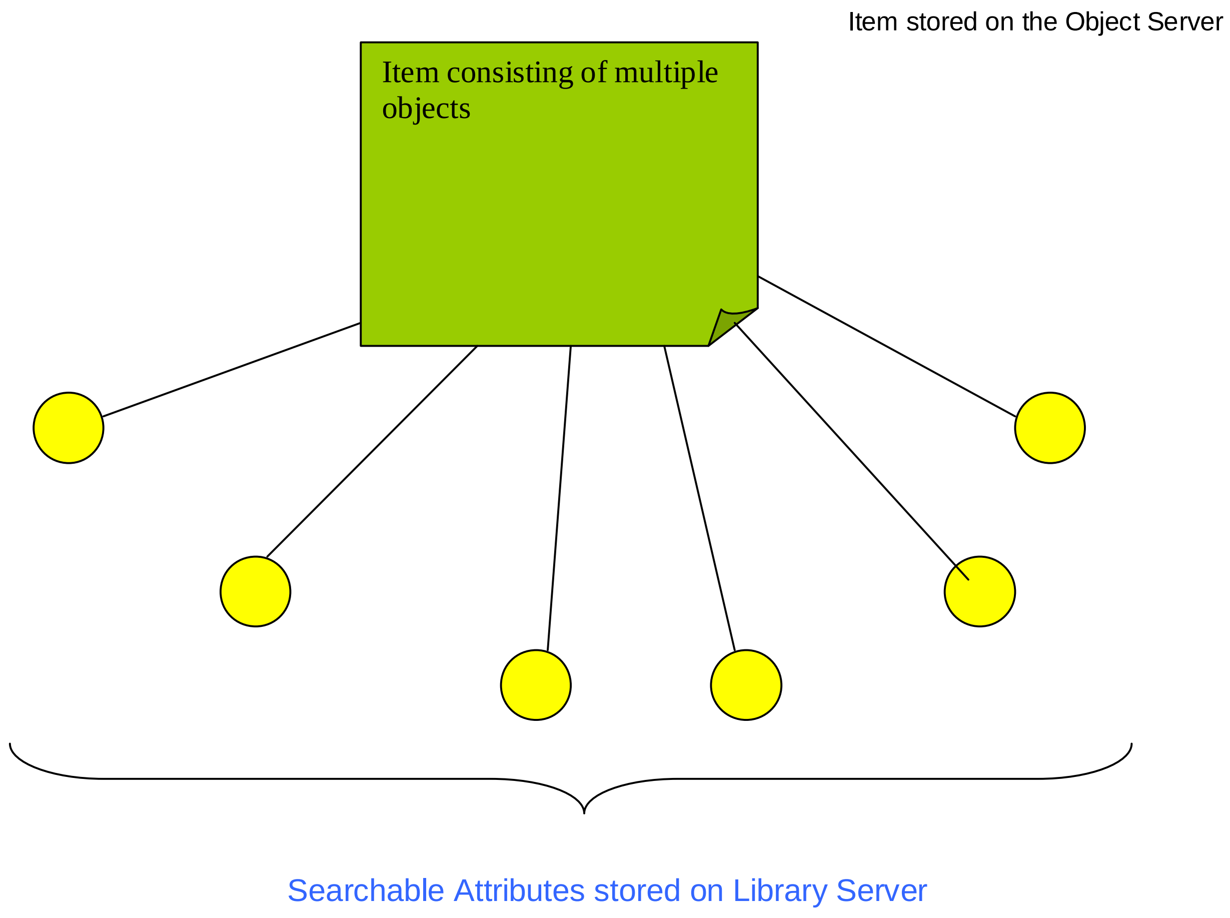
	Purging Backup/Archiving Protection against data corruption – keeping multiple version	and backup on other storage systems.
6.	Providing a Web based front-end to the citizens. This will act as an interface between the citizens and the government and will provide government services over the web.	EIP can be used along with Content Manager product to provide web clients.
7.	Record management policy setting by class of records where each class is owned by a single application and may span over multiple distributed databases	
8.	Data privacy support – dynamic access rights by column and tuples	
9.	Legacy system support? Integration of existing data	
10.	Integrated Workflow solution	

A Major Issue : Indexing and storing data

A major issue regarding e-Gov applications is the way the huge amount of data will be stored and available for query.

The way Content Manager implements it is through a Library Server and multiple Object Servers.

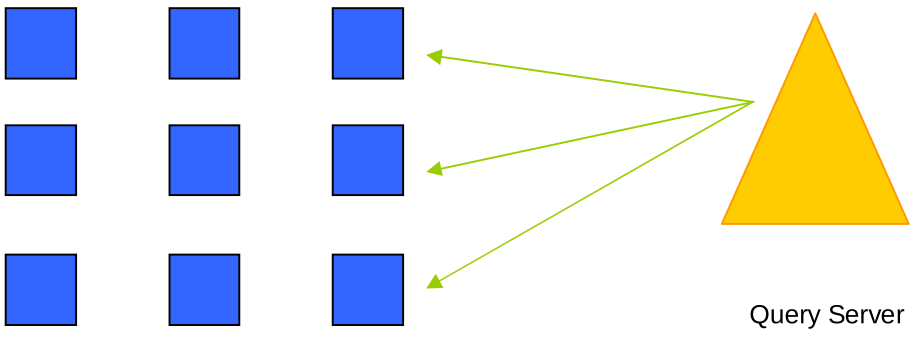
The documents (or items) are stored in the Object servers and their searchable attributes (key field values) are stored on the Library Server which acts as a catalog to all the items which are stored no various object servers connected to it. Hence what it means is that all the fields on which search may need to be carried out need to be stored on the Library Server so that it can be searched upon.


Item stored on the Object Server
Item consisting of multiple objects
Searchable Attributes stored on Library Server

Essentialy, the item acts as an anchor for the attributes which may be a photograph , scanned image , or and other object. A major concern which springs up from this is that if an application requires all or most its fields to be searchable then all that data will be stored on the Library Server itself. Since we have a central Library Server , the data may become very large and unmanageable for a single library server. And in that case nothing or very little will be stored on the Object Server which is meant for storing the actual data. This issue holds independent of implementation of the middleware and it needs to be decided what kind of data model we need to pursue to accomplish our data query needs in a heavy application as e-Gov.

We may need to look into other applications, which face similar problems for example, Google Search Engine and the EIP which connects to various repositories of data and gathers query results from them.

Google also needs to index a vast amount of information and provides quick results. It may have a distributed library server solution in which the query is sent to multiple search servers, which all search in parallel and report to the query server whatever they find.



Multiple Library (Search) servers

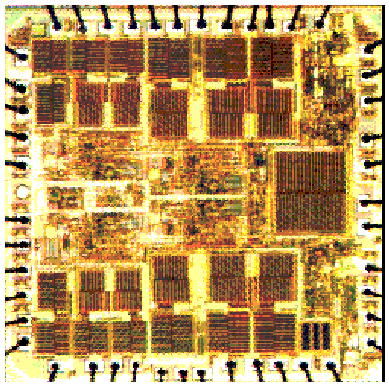
7.2 Appendix B : Some questions regarding Content Manager

Sr No.	Questions	Resolutions
1.	**What is maximum number of rows allowed in a Content Manager table for an index class?**	
2.	Can a Library Server be replicated? Distributed Library Servers? Defined 2 library servers L1 and L2, 2 Object servers O1 & O2. O1 & O2 was linked to both L1 & L2. The client could not log to L2 – error was thrown Can 2 library servers share an Object server	CM supports multiple LS but a single client cannot access multiple LS BUT does it support distributed LS in a single implementation?
3.	How can we span similar class of data across multiple object servers (the owner application, access control, policies and security of the class of data is same) (Using the concept of Collections) Ref: LDV example in Sys Admin manual	
4.	Can we restrict the view of index classes depending upon the user / application?	This can be done using the concept of access lists.
5.	Can we have queries over joins on metadata (Library server data)	
6.	Why do we need to write a custom application for say a license application? In other words, why won't the client provided with CM serve the purpose?	
7.	To what detail is the audit trail maintained (history) – can the audit trail be automatically archived/purged on the basis of policies	
8.	Data privacy support – dynamic access rights by column and tuples	
9.	Record management support – archiving, purging support etc.	
10.	Digital certificate management and authentication	
11.	Legacy system support? Integration of existing data	
12.	How are record management properties setup ?	
13.	How well is MQ Workflow integrated with CM	

7.3 Appendix C : Security using Smart Cards

Smart Card Technology

Similar to a credit card, a smart card stores information on an integrated microprocessor chip located within it.



Smart cards are secure, compact and intelligent data carriers. Though they lack screens and keyboards, smart cards should be regarded as specialized computers capable of processing, storing and safeguarding thousands of bytes of data. Similar in size and shape to plastic credit cards, smart cards with electrical contacts have a thin metallic plate just above center line on one side of the card. Beneath this dime-sized plate is an integrated circuit (IC) chip containing a central processing unit (CPU), random access memory (RAM) and non-volatile data storage. **Data stored in the smart card's microchip can be accessed only through the chip operating system (COS), providing a high level of data security.** This security takes the form of passwords that allow a user to access parts of the IC chip's memory or encryption/decryption measures, which translate the bytes stored in memory into information.

There are two basic kinds of smart cards. An "intelligent" smart card contains a central processing unit -- a CPU-- that actually has the ability to store and secure information, and "make decisions," as required by the card issuer’s specific applications needs. Because intelligent cards offer a "read/write" capability, new information can be added and processed. For example, monetary value can be added and decremented as a particular application might require.
The second type of card is often called a memory card. Memory cards are primarily information storage cards that contain stored value, which the user can "spend" in a pay phone, retail, and vending or related transaction.

The intelligence of the integrated circuit chip in both types of cards allows them to protect the information being stored from damage or theft. For this reason, smart cards are much more secure than magnetic stripe cards, which carry information on the outside of the card and can be easily copied. **Smart cards are an effective way of ensuring secure access to open interactive systems, such as encryption key mobility, secure single sign-ons and electronic digital signatures.**

The network computing and cellular telephone industries use smart cards to authenticate users in new systems that demand the utmost in security.

How is a chip card different from the magnetic stripe card that I carry in my wallet?
Existing magnetic stripe cards have limited capacities to carry information. A smart card carries more information than can be accommodated on a magnetic stripe card. It can make a decision, as it has relatively powerful processing capabilities that allow it to do more than a magnetic stripe card (e.g., data encryption).

What is the cost of an average chip card?
Trying to respond to this question is like asking the cost of a car without defining whether it is a used VW or a news Rolls Royce. Chip cards range from $.80 to $15 depending upon their capacity and quantities.

How secure and confidential are smart cards?
Smart cards actually offer more security and confidentiality than other financial information or transaction storage vehicles. A smart card is a safe place to store valuable information such as private keys, account numbers, passwords, or valuable personal information. **It's also a secure place to perform processes that one doesn't want exposed to the world, for example, performing a public key or private key encryption**.

Chip cards have computational power to provide greater security, allowing verification of the cardholder. Entering a PIN is one method of verification. The benefit of the smart card is that you can verify the PIN securely, off-line.

A possible application for smart cards

Suppose your driver license were carried on a smart card. The front and back would look as they do now—photo and demographic information on one side, notations and codes in excruciatingly tiny print on the other. Outwardly, no difference. Inside? Another matter entirely.

The microchip inside your smart card would carry name, address and physical description; it could hold your photographic likeness—compressed and digitized—and, perhaps, a fingerprint or other biometric measurement that is uniquely yours. Moreover, an encryption algorithm and secret key built into the microchip would protect all that data.

But don't stop there. Think of the paper-laden process of selling the family car or the complex documentation commercial vehicles require. Now, imagine all that paper transformed into electronic bytes stored in a vehicle smart card.

Some questions

1. Is it possible to store a photograph of the person inside the smart card, which may be required in the Inner Line Permit Application?

Links

- http://home.hkstar.com/~alanchan/papers/smartCardSecurity/

  A good article regarding security issues in SmartCard technology. Also looks at asymmetric cryptographic algorithms for personal authentication by government authorities.

- http://ntrg.cs.tcd.ie/mepeirce/Project/Chaum/cardcom.html

  Looks at various types of cards and their comparisons.

7.4 Appendix D : Reference Guide for the new API

InsertItem (Insert)

Format

InsertItem (*IndexClass, NoOfAttrib, AttribList, AttribValueList, NoOfParts, PartList, ItemID)*

Purpose

The **InsertItem** API creates an item in the index class you specify. You must specify any required attributes for that index class. You must also specify the parts(objects) of the item, which you want to create.

Parameters

IndexClass

char* - input
Indexclass name in which you want to create an item.

NoOfAttrib

int – input
The number of attributes in the *AttribList*.

AttribList

char** – input
The name of the attributes of the indexclass. When you create an item you must specify the attributes of indexclass in *AttribList*.

AttribValueList

char** - input
The value of the attributes. When you create an item you must specify the value of the attributes in *AttribValueList*.

NoOfParts

int – input
The number of parts in the *PartList.*

PartList

char** - input
This list contains the various parts of the items. Each entry in the *PartList* is a filename.

Return Values

On successful completion, this function returns zero and value of created item id is returned in *ItemID.*

SearchItems (Search)

Format

SearchItems (*IndexClass, NoOfAttrib, AttribList, AttribValueList, ItemIDList, NoOfHits*)

Purpose

Use the **SearchItem** API to locate an item in the index class you specify. You must specify the attributes and their values to define the search criteria.

Parameters

IndexClass

char* - input
Indexclass name in which you want to locate an item.

NoOfAttrib

int – input
The number of attributes in the *AttribList*.

AttribList

char** – input
When you search for an item you must specify the name of the attributes in *AttribList* on which you want to make a search.

AttribValueList

char** - input
The value of the attributes. When you search for an item you must specify the value of the attributes in *AttribValueList*.

ItemIDList

PITEMID& - output
The pointer to the item ID list. This list contains the documents item IDs that matches the search criteria.

NoOfHits

int& – output
Contains the number of items that match the criteria.

Return Values

On successful completion, this function returns zero. The *NoOfHits* contains the number of items and *ItemIDList* contains the list of item IDs.

GetItem (Get the information of an item)

Format

GetItem(*ItemID, NoOfAttrib, AttribList, AttribValueList, NoOfParts, PartList, PartHandleList, readwrite*)

Purpose

Use the **GetItem** API to get the attributes and parts information of an item.

Parameters

ItemID

ITEMID - input
The identifier of an item for which you want to get the information. This identifier is the item ID.

NoOfAttrib

int& – output
The number of attributes associated with the specified item.

AttribList

char**& – output
This list contains the name of the attributes associated with specified item.

AttribValueList

char** & - output
This list contains the value of the attributes.

NoOfParts

int& – output
The number of parts in the *PartList.*

PartList

char**& - output
This list contains the various parts(objects) of the specified item. Each entry in the *PartList* is a filename.

PartHandleList

HOBJACC*& - output

This list contains the handles of various parts (objects) of the specified item. When the parts information for an item is returned the various objects are opened to get the information.After using the parts(objects) information they need to be closed so *PartHandleList* contains the handles for those parts.

readwrite

BOOL – input
This variable specifies whether the item to be opened in share read mode or read write mode.

Return Values

On successful completion, this function returns zero.The *NoOfAttrib* contains number of attributes, *AttribList* contains name of the attributes, *AttribValueList* contains value associated with the attributes, *NoOfParts* contains number of parts in the item, *PartList* contains list of parts and *PartHandleList* contains the list of handles of parts of the item.

GetItemsInfo (Get the description of a list of items)

Format

GetItemsInfo(*NoOfItems, ItemIDList, ItemDescription*)

Purpose

Use the **GetItemsInfo** API to get the description of a list of items.

Parameters

NoOfItems

int - input
The number of items in the *ItemIDList*.

ItemIDList

PITEMID – output
This is the list of item IDs.

ItemDescription

char** & - output

This contains the description of list of items. Desription means the information like whether item is a document, folder, workbasket or workflow and associated attributes of the item.

Return Values

On successful completion, this function returns zero. The *ItemDescription* contains description of list of items.

DeleteItem (Delete)

Format

DeleteItem(*ItemID)*

Purpose

Use the **DeleteItem** API to delete a folder or document.

Parameters

ItemID

ITEMID - input
The identifier of an item you want to delete. This identifier is the item ID.

Return Values

On successful completion, this function returns zero.

CloseObject(close)

Format

CloseObject(*ObjectAccHandle)*

Purpose

Use the **CloseObject** API to close an object.

Parameters

ObjectAccHandle

HOBJACC - input
The object access handle.

Return Values

On successful completion, this function returns zero.

ReplaceObject(Replace)

Format

ReplaceObject(*ObjectAccHandle)*

Purpose

Use the **ReplaceObject** API to replace an object. The object is stored into the server by replacing the existing object.

Parameters

*ObjecAcctHandle*t

HOBJACC - input
The object access handle. The object is stored into the server by replacing the existing object.

Return Values

On successful completion, this function returns zero.

UpdateItem (Update)

Format

UpdateItem (*IndexClass ,ItemID, NoOfAttrib, AttribList, AttribValueList, NoOfParts, PartList)*

Purpose

The **UpdateItem** API update an item in the index class you specify. You must specify the attributes, of the item, which you want to change. You must also specify the parts(objects) of the item, which you want to change.

Parameters

IndexClass

char* - input
Indexclass name in which the specified item belongs.

ItemID

ITEMID& – input/output
When an item is updated, the new item ID is issued for the updated item. The value of new item ID is returned in *ItemID.*

NoOfAttrib

int – input

The number of attributes in the *AttribList*.

AttribList

char** – input
The name of the attributes which need to be updated.

AttribValueList

char** - input
The changed value of the attributes. The name of the attributes are in the *AttribList.*

NoOfParts

int – input
The number of parts in the *PartList.*

PartList

char** - input
This list contains the various parts of the items. Each entry in the *PartList* is a filename.

Return Values

On successful completion, this function returns zero and value of new item id is returned in *ItemID.*

Logon(Log on)

Format

Logon(*LoginName, Password)*

Purpose

Use the **Logon** API to do the login.

Parameters

LoginName

const char* - input
The name of the user who want to do login.

Password

const char* - input
The password of the user who want to do login.

Return Values

On successful completion, this function returns zero.

Logoff(Log off)

Format

Logoff(*void)*

Purpose

Use the **Logoff** API to do log off from the system.

Parameters

None

Return Values

On successful completion, this function returns zero.

FreeMemory(Free the memory)

Format

FreeMemory(*void)*

Purpose

Use the **FreeMemory** API to free the memory.

Parameters

None.

Return Values

None.

StartWorkFlow(starts the workflow)

Format

StartWorkFlow(*WorkFlowName, ItemID)*

Purpose

Use the **StartWorkFlow** API to start the workflow means put the given item in the first workbasket of the workflow.

Parameters

WorkFlowName

char* - input
The name of the workflow which you want to start.

ItemID

ITEMID& - input
The identifier of an item which you want to put in the workflow. This identifier is the item ID.

Return Values

On successful completion, this function returns zero.

RouteWipItem(Route the item)

Format

RouteWipItem(*ItemID, NextWBID)*

Purpose

Use the **RouteWipItem** API to move an item from the workbasket where
it currently resides to the workbasket you specify. You can move the item to any
other workbasket in the routing list for its assigned workflow, or to a workbasket
that is not in the workflow. You can also use this function to move an item that is
not in a workflow, and can move that item to any workbasket.

Parameters

ItemID

ITEMID& - input
The identifier of an item you want move. This identifier is the item ID.
.

NextWBID

ITEMID& - output
The identifier of an workbasket in which you want to move the item. This identifier is the item ID.

Return Values

On successful completion, this function returns zero.

FindWBForItem(Find Out which Workbasket an Item Is in)

Format

FindWBForItem(*ItemID, WorkBasketName)*

Purpose

Use the **FindWBForItem** API to return information about the workbasket an item is in.

Parameters

ItemID

ITEMID - input
The identifier of an item of which you want to get the information. This identifier is the item ID.
.

WorkBasketName

char* & - output
The name of the workbasket in which the give item resides.

Return Values

On successful completion, this function returns zero.The workbasket name is returned in *WorkBasketName.*

GetWBName(Get the workbakset name)

Format

GetWBName(*WorkBaksetID, WorkBasketName)*

Purpose

Use the **GetWBName** API to get the name of the workbasket.

Parameters

WorkBaksetID

ITEMID - input
The identifier of workbasket of which you want to get the workbakset name. This identifier is the workbasket ID.
.

WorkBasketName

char* & - output

The name of the workbasket.

Return Values

On successful completion, this function returns zero.The workbasket name is returned in *WorkBasketName.*

GetNextWBForItem (Get the next workbakset in workflow)

Format

GetNextWBForItem (*ItemID, WorkBasketID)*

Purpose

Use the **GetNextWBForItem** API to get the ID of the next workbasket in workflow for a given item.

Parameters

ItemID

ITEMID - input

The identifier of item for which you want to get the ID of the next workbakset . This identifier is the item ID.

.

WorkBasketID

ITEMID& - output

The identifier of the workbasket.

Return Values

On successful completion, this function returns zero.The ID of the next workbasket name is returned in *WorkBasketID.*

GetWorkFlowID (Get the workflow ID)

Format

GetWorkFlowID(*WorkFlowName, WorkFlowID)*

Purpose

Use the **GetWorkFlowID** API to get the ID of the workflow.

Parameters

WorkFlowName

char* - input
The name of the workflow of which you want to get the *WorkFlowID*..

WorkFlowID

ITEMID& - output
The identifier of workflow. This identifier is the workflow ID.
.

Return Values

On successful completion, this function returns zero.The workflow ID is returned in *WorkFlowID.*

GetWorkBasketID (Get the workbasket ID)

Format

GetWorkBasketID(*WorkBasketName, WorkBasketID)*

Purpose

Use the **GetWorkBasketID** API to get the ID of the workbasket.

Parameters

WorkBasketName

char* - input
The name of the workbasket of which you want to get the *WorkBasketID*..

WorkBasketID

ITEMID& - output
The identifier of workbasket. This identifier is the workbasket ID.
.

Return Values

On successful completion, this function returns zero.The workBasket ID is returned in *WorkBasketID.*