ePHOENIX
Architecture
Version Package 6

Summary

This document describes the ePhoenix package 6 common architecture and variants.

Note:

This document is draft version and needs further extension and verification

Copyright notice

European Patent Office
Postbus 5818
NL-2280 HV Rijswijk
The Netherlands

Document information

Document title
GIM 1.1 ePHOENIX B.doc

Document number

Contact point in case of questions or problems	
Person	Peter Dedman
Location	Shell
Room	SO1M34
Extension	2542

Configuration history			
Version	**Date**	**Changes**	**Author**
1.1	26 APR 05	New draft	P.G.Dedman
1.1a	27 APR 05	Adapted for Open Source	P. Tubbing

Reviews			
Version being reviewed			
Reviewed by (Position in CAPITALS)	**Reviewed by (Name in CAPITALS)**	**Reviewed by (Signature)**	**Date reviewed**

Sign-off			
Version being signed off			
Signed off by (Position in CAPITALS)	**Signed off by (Name in CAPITALS)**	**Signed off by (Signature)**	**Date signed off**

Table of contents

Table of figures

1 Introduction

This document describes the ePhoenix package 6 common architecture and variant.

1.1 Audience

The intended audience is people need to know the ePhoenix package 6 common architecture and variants..

This audience is

- OSA testers
- highly experienced systems administrators.

2 Introduction

ePhoenix is a core component in the epoline suite of applications. In its simplest form, it is the storage system for electronic dossiers.

The ePHOENIX system supports the administrative and formal tasks of the European patent procedure in Search, Examination, Opposition and Appeal as well as those under the Patent Co-operation Treaty (PCT).

ePHOENIX is part of the MADRAS (Mother of All Dossier Related Application Systems (MADRAS) suite of tools, which runs under.

The main objective of ePHOENIX is to automate and replace paper dossiers and messages in patent applications:

- to eliminate the movement and storage of physical dossiers
- to reduce archive storage capacity
- to allow almost instantaneous access to dossiers irrespective of their location
- to eliminate the need to search for mislaid dossiers
- to make patent processing generally more efficient by enhancing automation support
- to provide a basis for electronic interaction between the EPO and its customers in the *epoline*® framework.

These objectives have been achieved by presenting the user with all the information needed to handle a specific piece of work, such that :

- all patent application and related documents are in electronic format
- the dossier handling is integrated with the user's mailbox
- there is a table of contents (TOC) for the dossier, from which the user can select documents to work on. In many cases reading the information in the TOC may avoid the need to view the document.
- data from other (legacy) servers handling procedural data is integrated with the user's view
- the system maintains a history of the application
- notes can be added to the applications by users.

Several display windows, each containing a set of documents or a range of bibliographic data, can be opened simultaneously. The user can then browse through all documents in a dossier or through all the pages of a document.

The system does nothing more than store the electronic dossiers, managing these dossiers and presenting the information from the electronic dossier together with legacy data from other sources. There is no dependency build in the system about the content of the dossiers. This makes it possible to use the ePHOENIX system for patent related dossiers, trademark dossiers or even administrative dossiers.

The system currently in production at the EPO has been ported to a java system, using a three-tier architecture. ePHOENIX is a collection of services that collects data about dossiers from the Dossier Management System (DMS) and images of documents from the ePHOENIX Image Archive (PXI) and presents them to the MADRAS interface for display. It also makes changes to these databases.

ePHOENIX also handles display of procedural data and import of documents from legacy systems and presents them to the user interface. For the user, this means that all data related to a given dossier is presented in one go.

The ePHOENIX architecture can be considered in terms of:

- application architecture
- data architecture
- business architecture
- security architecture

3 Application architecture

The application architecture describes the main components of the ePHOENIX architectural environment

3.1 Overview

ePHOENIX is at the core of a system consisting of the following main components:

- epoScan system, which allows paper documents to be scanned into ePHOENIX.
- MADRAS GUI, which presents the user's view of the system, integrating the ePHOENIX interface with other applications, including batch indexing of documents, emulators for mainframe systems, user configuration tools etc.
- ePHOENIX
- web-based monitoring
- back-end systems, including legacy (procedural) servers and the databases for the dossier images and associated data.

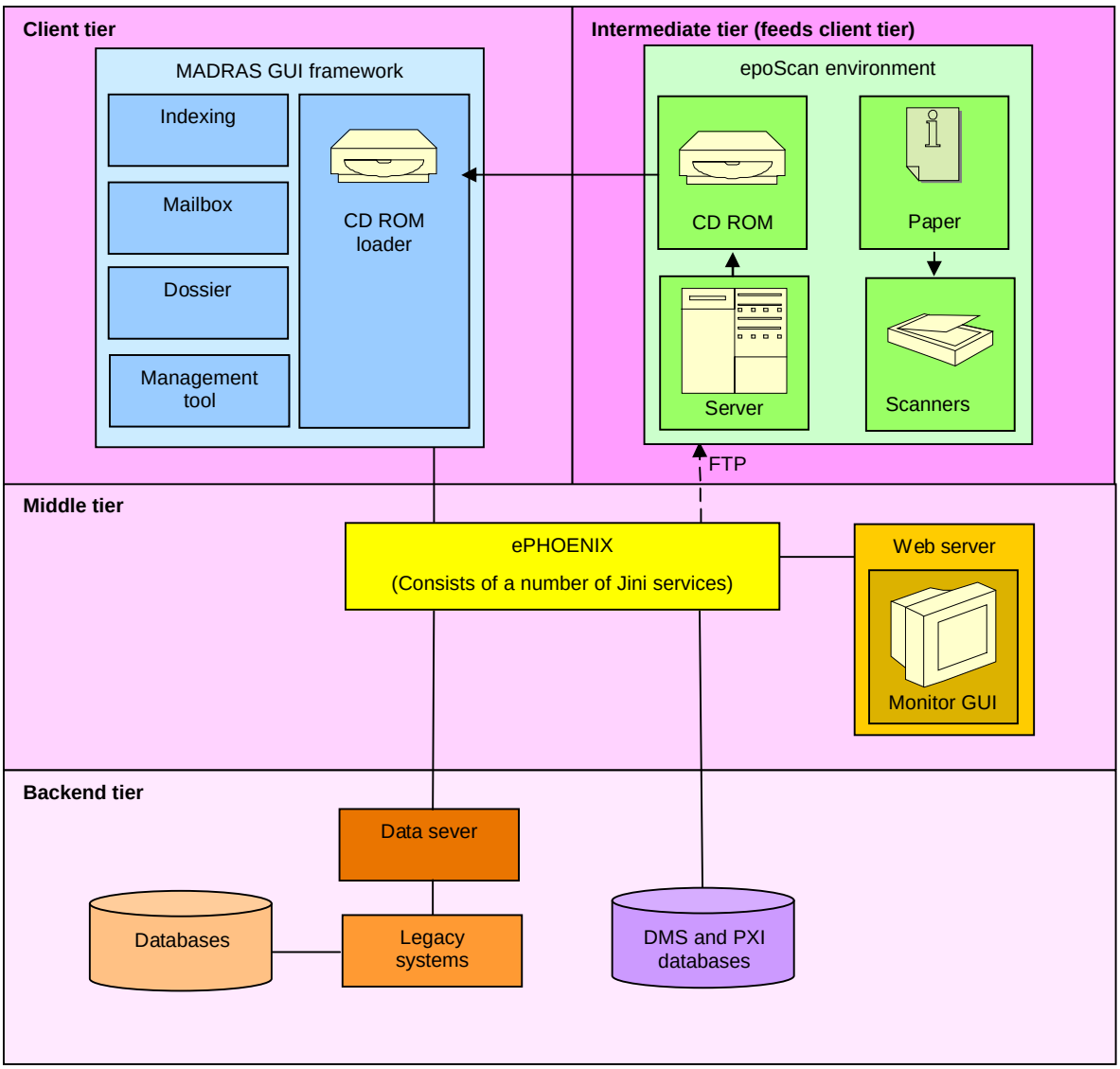

Figure 1: ePHOENIX architectural environment main components

3.2 Tiers

The ePHOENIX architectural environment can be described as a classic three-tier architecture. A thin client communicates to a middle tier server where the business code resides. The data is stored in the third tier, the data server.

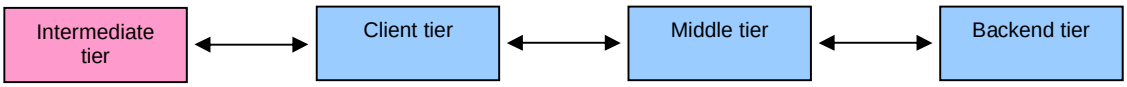

Figure 2: application architecture

ePHOENIX is a collection of services that collects data about dossiers from the Dossier Management System (DMS) and images of documents from the ePHOENIX Image Archive (PXI) and presents them to the MADRAS interface for display. It also makes changes to these databases.

3.2.1 EPO ePHOENIX services architecture

The system currently in production at the EPO has been ported to a Java system, using a three-tier architecture.

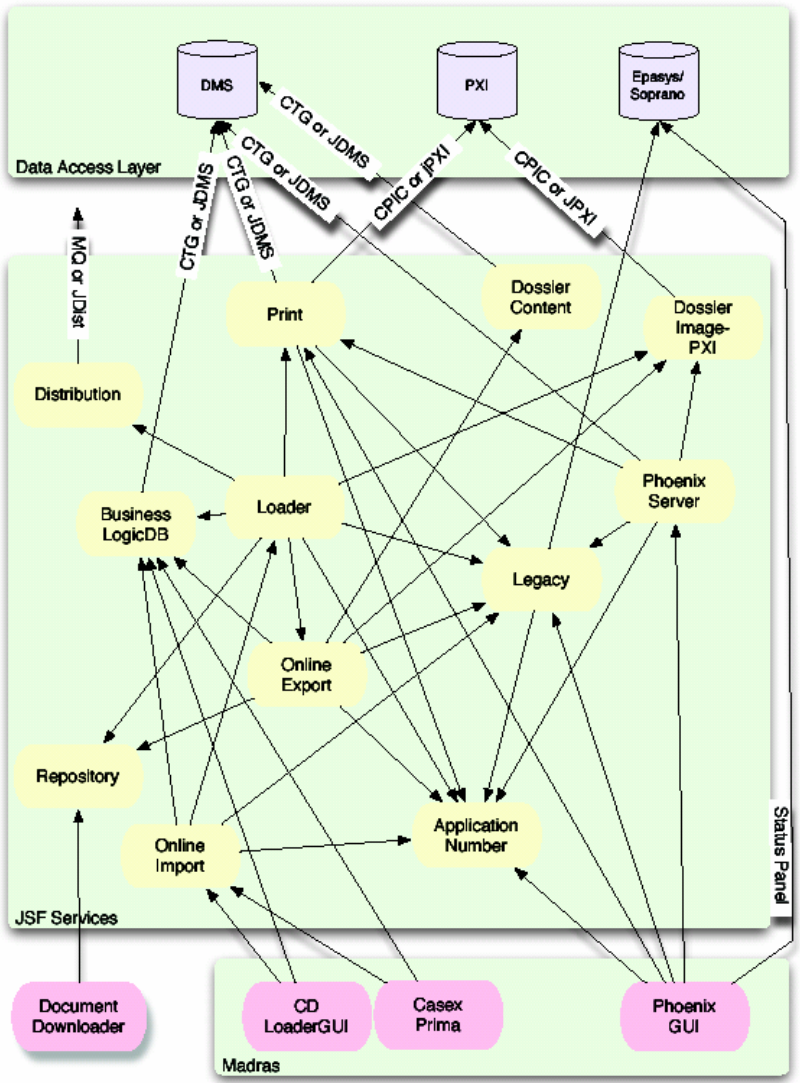

Figure 3: EPO ePHOENIX services architecture

3.2.2 ePHOENIX services architecture

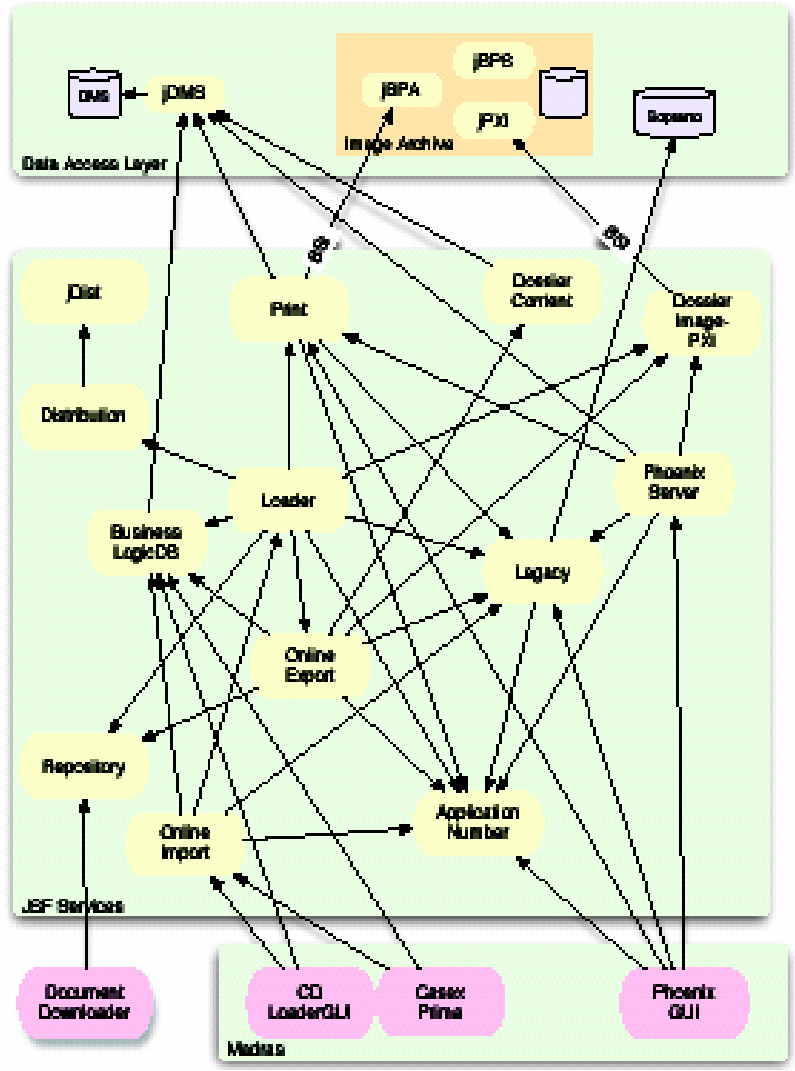

Figure 4: ePHOENIX services architecture

3.2.3 Intermediate tier

The intermediate feeds the client tier.

3.2.3.1 epoScan environment

The epoScan application is responsible for scanning documents from paper. The indexing application creates control files and these are checked during scanning to ensure that no pages are missed or miscounted. The result is a CD containing images, which are presented to the Import Manager by means of the CD ROM loader application in MADRAS.

3.2.4 Client tier

The client layer of the ePHOENIX system presents the electronic dossiers to the end-user. The client is 100% Java compatible.

The adapter is considered as part of the client tier.

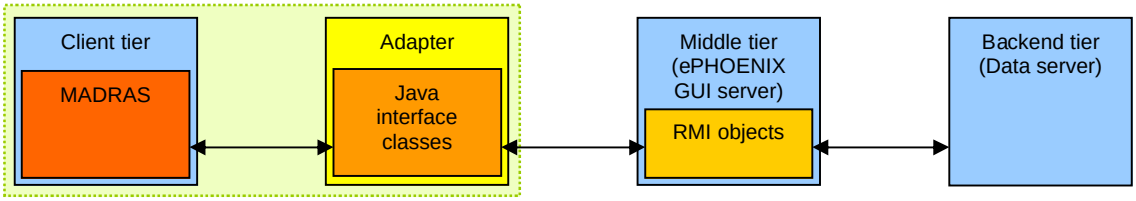

Figure 5: client tier relationships

In order to have a consistent user-interface that has maximum integration, a framework is used in which the client application run. This framework is called Mother of All Dossier Related Applications System (MADRAS). It is not within the scope of this document to describe this framework.

3.2.4.1 MADRAS GUI Framework

The user interface provides a series of windows and panels in an easily selectable notebook format containing comprehensive dossier-related data. Scanned documents are displayed in the Image Viewer, Immediate access is provided to legacy systems. All functions (indexing, mailbox etc.) are implemented with individual applications inside the MADRAS framework.

3.2.4.2 Adapter

The adapter consists of a set of Java interfaces. The adapter class hides the connection from the client to the middle tier. The only purpose of the adapter is to allow updating of the Client tier without having to reconstruct the middle tier.

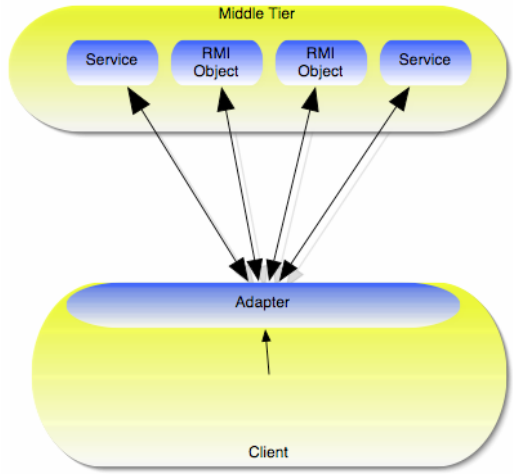

Figure 6: adapter architecture

3.2.5 Middle tier

The middle tier is also called the ePHOENIX GUI server. All business related functions are implemented on this tier. It is 100% Java compatible.

There are a set of Remote Objects (RO) that act as the interface to the client tier. Each Remote Object implements a certain set of functions.

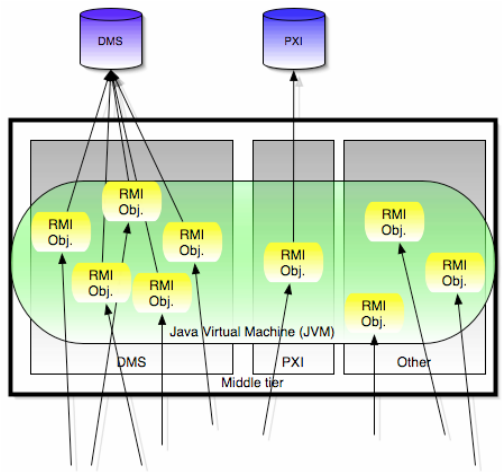

Figure 7: middle tier architecture

The Remote Objects can be divided in four groups, depending on the data they manipulate:

- Dosier Management System (DMS) related
- DMS meta data related
- ePHOENIX Image archive (PXI) related
- others.

The Remote Objects can be mapped to the business functions.

Remote objects		**Business functions (described in chapter 5, Business architecture)**													
Data types: • Dosier Management System (DMS) related (D) • DMS meta data related (M) • ePHOENIX Image archive (PXI) related (P) others (O).	**Data type**	**Create Batch**	**Create Back file**	**Edit Batch**	**Edit Package**	**Batch Transfer**	**CD Loader**	**Paper Storage**	**Mailbox**	**Dossier**	**Team Manager**	**Manager**	**Print Jobs**	**Print Queue**	**Select team**
Authentication	O	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Dosman	D														
DossierManager	D									X					
ImageServer	P									X					
Indexing	D	X	X	X	X	X		X		X					
Mailbox	D								X	X	X				X
Algorithm	M											X			
DocumentControl	M											X			
DocumentSet	M											X			

Remote objects		Business functions (described in chapter 5, Business architecture)													
Data types: • Dosier Management System (DMS) related (D) • DMS meta data related (M) • ePHOENIX Image archive (PXI) related (P) others (O).	Data type	Create Batch	Create Back file	Edit Batch	Edit Package	Batch Transfer	CD Loader	Paper Storage	Mailbox	Dossier	Team Manager	Manager	Print Jobs	Print Queue	Select team
Function	M											X			
Global	M											X			
Label	M											X			
Legacy	M											X			
MVSConfig	M											X			
OverlayLayout	M											X			
Paperfile	M											X			
Procedure	M											X			
Profile	M											X			
Range	M											X			
Team	M											X			
MemberDetails	M											X			
Printing	DP												X		
Queue	D													X	
ReportKeeper	O	X	X	X	X	X	X	X	X	X	X	X	X	X	X
UserSettings	d	X	X	X	X	X	X	X	X	X	X	X	X	X	X

The Remote Objects can be divided in four groups, depending on the data they manipulate:

- Dosier Management System (DMS) related
- DMS meta data related
- ePHOENIX Image archive (PXI) related
- others.

The Remote Objects can be mapped to the business functions.

Remote objects		Business functions (described in chapter 5, Business architecture)													
Data types: • Dosier Management System (DMS) related (D) • DMS meta data related (M) • ePHOENIX Image archive (PXI) related (P) others (O).	Data type	Create Batch	Create Back file	Edit Batch	Edit Package	Batch Transfer	CD Loader	Paper Storage	Mailbox	Dossier	Team Manager	Manager	Print Jobs	Print Queue	Select team
Authentication	O	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Dosman	D														
DossierManager	D									X					
ImageServer	P									X					
Indexing	D	X	X	X	X	X		X		X					
Mailbox	D								X	X	X				X
Algorithm	M											X			
DocumentControl	M											X			
DocumentSet	M											X			
Function	M											X			
Global	M											X			
Label	M											X			
Legacy	M											X			
MVSConfig	M											X			
OverlayLayout	M											X			
Paperfile	M											X			
Procedure	M											X			
Profile	M											X			
Range	M											X			
Team	M											X			
MemberDetails	M											X			
Printing	DP												X		
Queue	D													X	
ReportKeeper	O	X	X	X	X	X	X	X	X	X	X	X	X	X	X
UserSettings	d	X	X	X	X	X	X	X	X	X	X	X	X	X	X

3.2.6 Backend tier

Services access the backend systems or data servers:

- Dossier Image Service: accessing PXI Image Archive
- Dossier State Service: accessing all legacy data of applications that is not stored in DMS.
- DMS Service: accessing/updating all DMS related data.
- Print Service: printing/soft-copying all PXI data.

The business functions in the server can be divided in the following categories:

- indexing
- dossier management
- team management
- message management
- metadata management.

All business functions are supported by one or more services.

3.2.6.1 EPO data server

The EPO data server is implemented as a separate tier using a mainframe CICS/DB2 System

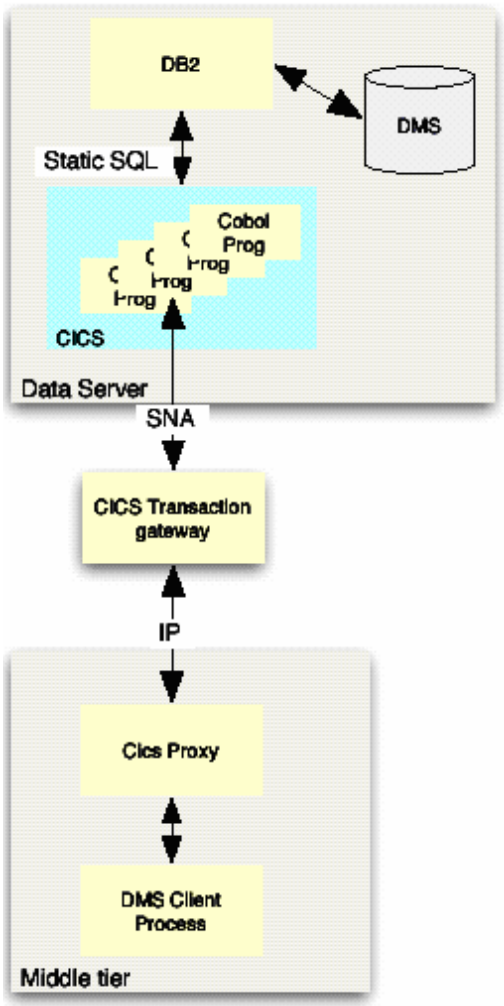

Figure 8: CICS data server

3.2.6.2 Legacy systems

The data from legacy systems is available to the user interface by means of data server. This means that any legacy system can be used as long as the data server is created to pass data consistently across to the Legacy Interface. ePHOENIX also requires access to the documents that were printed by the legacy systems in the past.

3.2.6.3 Softcopy

If a system which is external to ePHOENIX can generate a print stream or file this can be captured. The process of capturing the print stream/file and then importing it became known as softscanning.

An image already in ePHOENIX can be copied. ePHOENIX has its own printing facility for printing images and so, if it captures its own print stream, called softcopying and simply reloads it the same effect as softscanning is achieved.

Hence softcopying is really the same as softscanning, but with different sources.

3.2.6.3.1 Softcopy EPO

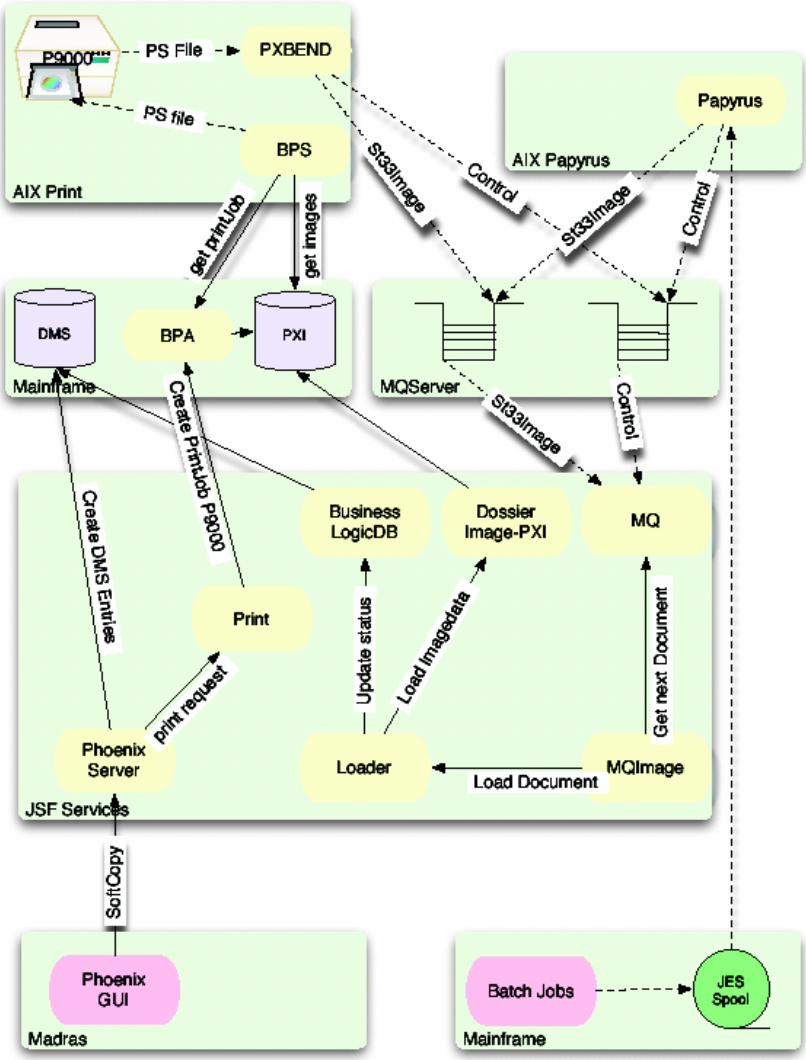

Figure 9: softcopy EPO

4 Data architecture

The data architecture describes how the data is stored.

The current data architecture can de divided in two parts:

- the Dossier Management System (DMS) contains the dossier related information describing the electronic dossier
- the ePHOENIX Image Archive (PXI) contains the image related information.

The corresponding data is stored in different subsystems.

4.1 DMS

DMS is a database that stores ePHOENIX data for each dossier such as

- document details
- annotations
- history
- checklists.

Its main task, for the end user, is to control the table of contents. Whenever a change is made to a dossier, the DMS is updated, in most cases when the dossier is closed. PXI

This is the database system that holds all document images. It delivers images to the workstation at least as fast as they can be displayed on the screens (that is, two pages per second) and all images Application architecture.

The database can be split in two different parts:

- dossier related information
- meta-data.

The dossier related data is the live part of the database. For each electronic dossier in the system, records will be saved in this part of the database. The following classes are involved:

- Dossier
- Package
- Package History
- Document
- Document Note
- Procedural Group
- Participant
- Dossier Key
- Dossier Note
- Dossier Lock
- Dossier Count
- Action Log
- History
- Owner Team
- Owner Member
- Check Item Values
- Check Item Comment
- CDR
- Batch
- Box
- Box History
- Soft Batch
- Message
- Message data

The relationships between these objects is shown schematically in Figure 10: dossier related class diagram. This is not a complete class diagram, but it shows the main dependencies.

The classes coloured yellow are part of the dossier data. The classes coloured grey are links to the meta data.

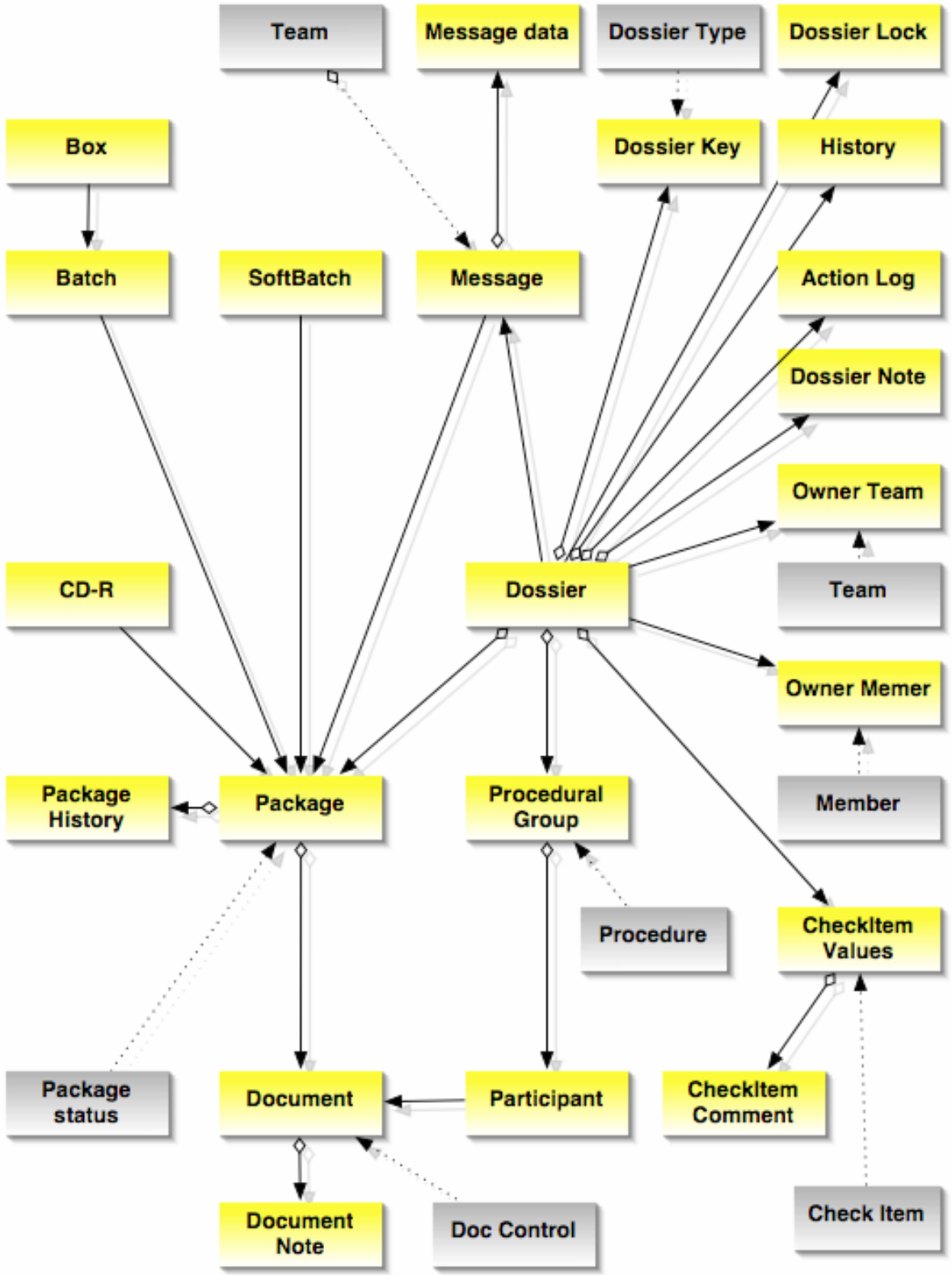

Figure 10: dossier related class diagram

The meta data part of the DMS database contains all the data that drives the rest of the application or that describes general information about the live data.

The following classes are defined:

- Algorithm
- Checklist Item
- Checklist Type
- Comment
- Document Set Item
- Document Set
- Document List
- Doc Control
- Dossier mask
- Dossier Type
- Form
- Global
- Information Sheet Item
- Information Sheet
- Label Cover
- Legacy Server
- PMSetting
- Profile
- MVS Config
- Office
- Overlay
- Overlay Layout
- Range Item
- Paper file
- Paper file Item
- Paper file Type
- Package Status
- ePHOENIX Function
- Profile
- Profile Item
- Procedure
- Status Screen Group
- Status Screen
- Status Screen Item
- Request Data
- Request
- Request Queue
- Team
- Member
- User.

The relationships between these objects is shown schematically in Figure 11: meta data class diagram. This is not a complete class diagram, but it shows the main dependencies.

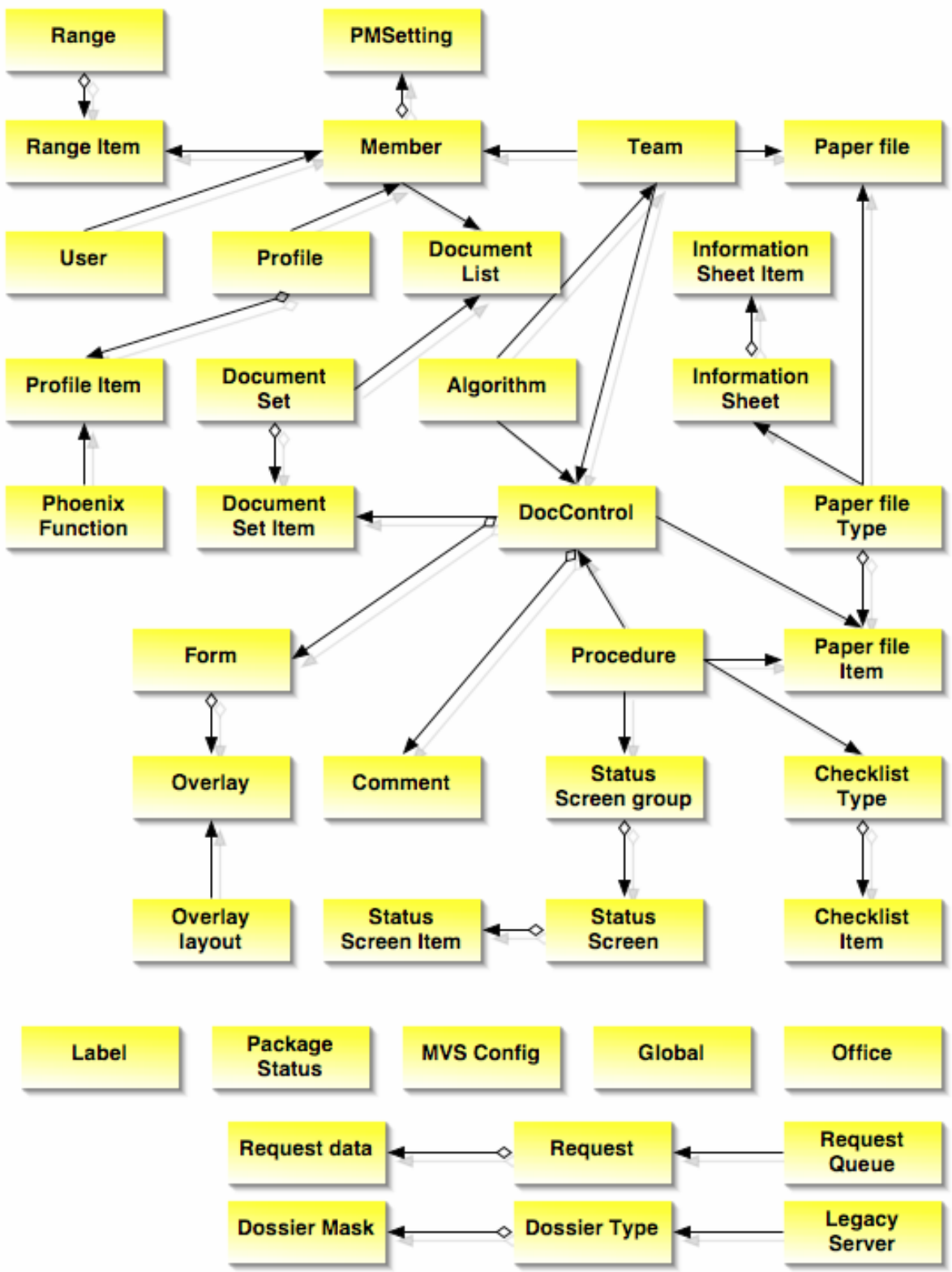

Figure 11: meta data class diagram

Business processes and data entities														
	Create Batch	Create Back file	Edit Batch	Edit Package	Batch Transfer	CD Loader	Paper Storage	Mailbox	Dossier	Team Manager	Manager	Print Jobs	Print Queue	Select team
Box History							Rw							
Box			R				Rw							
Batch	Rw	Rw	Rw	Rw	Rw	R	R		R					
SoftBatch									R					
Message								Rw	Rw	Rw				
Message Data								R		R				
CD-R						Rw			R					
Dossier	Rw	Rw							Rw					
Dossier Key	Rw	Rw							R					
Dossier Lock	R	R							Rw					
Dossier Count														
Dossier Note									Rw					
History									Rw					
Action Log									Rw					
Owner Team									Rw					
Owner Member									Rw					
Check Item Values									Rw					
Check Item Comment									Rw					
Package	Rw	Rw	Rw	Rw	R	Rw			Rw					
Package History	Rw	Rw	Rw	Rw		Rw			Rw					
Document	Rw	Rw	Rw	Rw	R				Rw					
Procedural Group	Rw	Rw	Rw	Rw					Rw					
Participant	Rw	Rw	Rw	Rw					Rw					
Document Note	Rw	Rw	Rw	Rw					Rw					

Business processes and data entities														
	Create Batch	Create Back file	Edit Batch	Edit Package	Batch Transfer	CD Loader	Paper Storage	Mailbox	Dossier	Team Manager	Manager	Print Jobs	Print Queue	Select team
Range										R	Rw			
Range Item										R	Rw			
Member	R	R	R	R				R	R	Rw	Rw			R
PmSetting									Rw	Rw	Rw			
Team	R	R	R	R				R	R	Rw	Rw			R
Paper File									R		Rw			
Paper File Type									R		Rw			
Paper File item											Rw			
Information Sheet									R		Rw			
Information Sheet Item									R		Rw			
User	R	R	R	R				R	R	R	Rw			R
Profile									R	R	Rw			
Profile Item									R	R	Rw			
ePHOENIX Function									R	R	Rw			
Document List	R	R	R	R					R	Rw	Rw			
Document Set	R	R	R	R					R	R	Rw			
Document Set Item	R	R	R	R					R	R	Rw			
Algorithm											Rw			
Doc Control	R	R	R	R	R				R		Rw			
Form									R		Rw			
Overlay									R		Rw			
Overlay Layout									R		Rw			
Comment											Rw			
Procedure	R	R	R	R					R		Rw			
Checklist Type									R		Rw			

Business processes and data entities														
	Create Batch	Create Back file	Edit Batch	Edit Package	Batch Transfer	CD Loader	Paper Storage	Mailbox	Dossier	Team Manager	Manager	Print Jobs	Print Queue	Select team
Checklist Item									R		Rw			
Status Screen Group									R		Rw			
Status Screen									R		Rw			
Status Screen Item									R		Rw			
Label									R		Rw			
Package Status	R	R	R	R					R		Rw			
MVS Config											Rw			
Global	R	R	R	R					R		Rw			
Office														
Request Queue											Rw	R	R	
Request											Rw	R	R	
Request Data											Rw	R	R	
Legacy Server	R	R	R	R					R		Rw			
Dossier Type	R	R	R	R					R		Rw			
Dossier Mask	R	R	R	R					R		Rw			

4.2 PXI

The actual image data is stored in the PXI. The PXI system is package based. Each package has a unique key. This key corresponds to the key of a Package in the DMS context. As a result, multiple documents in DMS are stored as a single package in PXI. Each document in DMS contains an offset in order to link it to the correct image pages in PXI.

Documents may be stored in different formats:

- WIPO St33
- TIFF
- PDF
- BLOB

Access to the documents is page based except in the case the documents are stored in BLOB format.

The PXI is for any client application treated as a black box. The interface to the PXI is called Bacon [Back-file Conversion] Service Interface (BSI).

Printing is handled outside of JPXI.

5 Business architecture

The business architecture describes the data creation process and defines each of the functions used within that process

A functional split can be made in processes that support the creation of data in the system and the processes that use and manipulate the data stored in the system.

5.1 Data creation processes

The data creation process stores the paper documents in electronic format in the ePHOENIX system. The document capture process covers the following steps:

- paper handling
- indexing of documents and creation of batches
- transfer of batch data
- scanning of batches
- image enhancement
- CD-ROM creation
- CD-ROM loading
- paper storage.

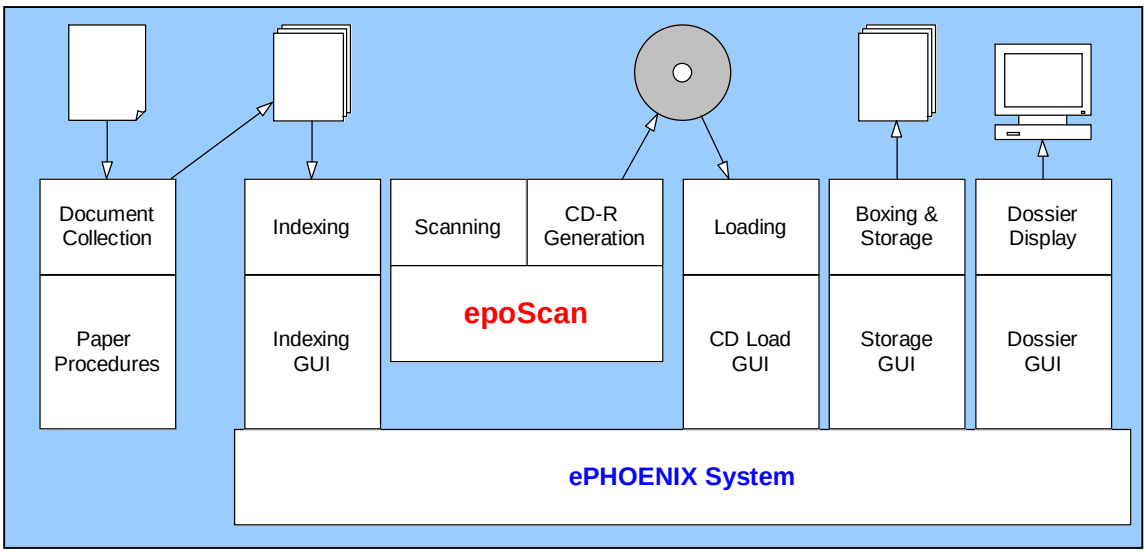

Figure 12: data creation process

All documents entering the ePHOENIX system must be placed into batches and be indexed. In ePHOENIX, indexing is equivalent to creating a batch.

The information entered during indexing of documents determines what happens to those documents in the system. Documents are stored inside Dossiers and, within Dossiers, inside Procedures (PCT etc.). Also, each document receives a Document Code or a name and this determines whether a MADRAS user will receive a message and which user that is. Hence, the Document Code given at this stage determines who will process the document later on. Once that person has seen the document they may determine that a different code is more appropriate. Hence, this initial indexing is sometimes called pre-indexing since the values used may be changed later on.

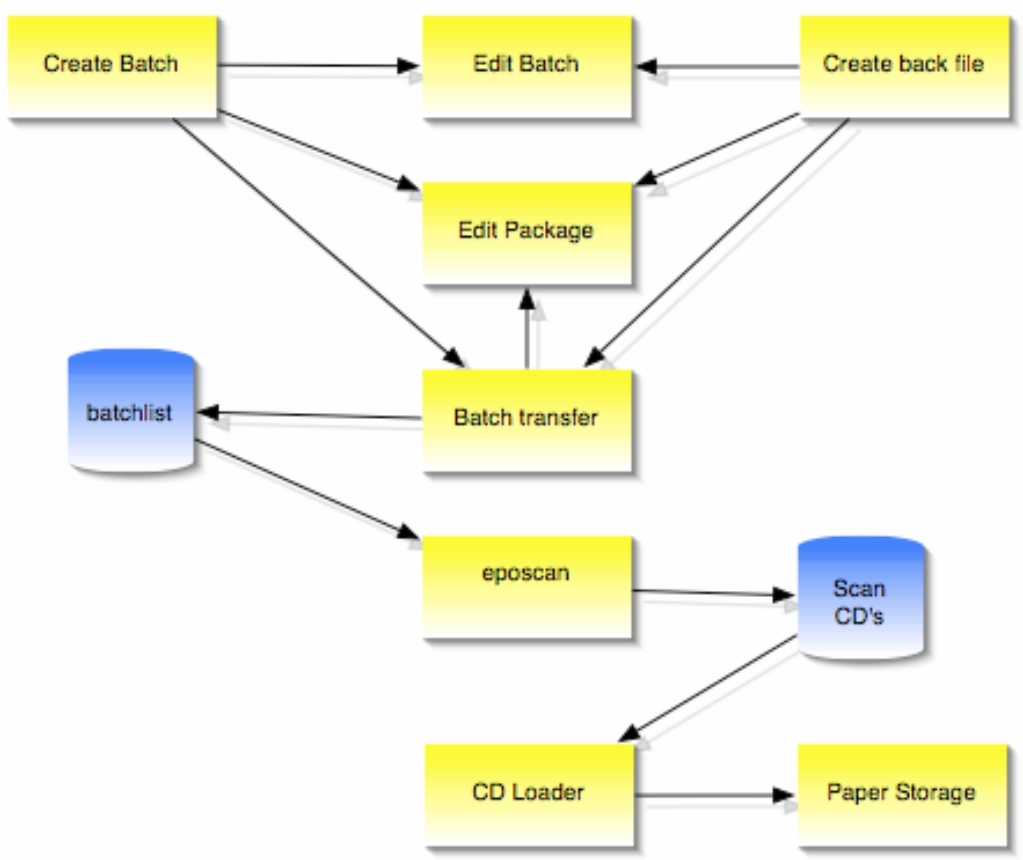

Figure 13: indexing business processes

5.1.1 Create Batch

The Create Batch function supports the indexing of paper documents prior to scanning. This function is only used only for placing newly arriving documents into existing electronic dossiers.

The scanning process requires batches to be created, a batch is made up of packages, and a package is a set of documents for a dossier that has the same legal date. The maximum number of pages in a batch is determined by the amount of pages that fit in a box that is used for storage of the paper.

The indexing function is the main part of the Create Batch function. The following data can be indexed on a document:

- document code
- legal date
- number of pages
- indication fax
- annotation
- procedure and participant
- message distribution details.

The function also allows previously indexed packages to be added to a batch.

5.1.2 Create Back File

The Create Back File function supports the indexing of paper documents prior to scanning.

This function is only used for back file scanning of complete dossiers. Although the function looks very similar to the Create Batch function, there are some specific business rules associated with this function.

- Packages get a different status
- User that first saw the electronic document is pre-filled.
- The indexing rules for the rest are the same as in normal indexing.

5.1.3 Edit Batch

The Edit Batch function supports the editing of already indexed data that has not yet been scanned.

This function is executed prior to scanning to optimize the batches that are sent to scanning. In addition, mistakes made during indexing can be corrected.

5.1.4 Edit Package

The Edit Package function supports the editing of previously indexed data that has not yet been scanned. This function is normally executed when the scanning process reports problems such as page miscount. Whenever a package is edited, the package is removed from its original batch. In order to get it scanned, it should be added again in a batch during running the Create Batch function.

5.1.5 Batch Transfer

The Batch Transfer function supports the creation of the indexing data in a form that allows the epoScan environment to use it. The epoScan environment runs without a link to the ePHOENIX databases.

5.1.6 CD Loader

The CD Loader function supports the loading of CD's/DVD's with image data into the ePHOENIX system.

Either complete CD's/DVD's can be loaded or a set of documents from the medium can be loaded in one session.

The following list of media is supported:

- Scan CD, CD created by epoScan application
- Japanese Priority CD, CD from JPO containing Japanese priority documents in SDIF format
- WIPO DVD, the new DVD from WIPO containing pamphlets, priority documents and others.
- WIPO DVD, DVD from WIPO containing priority documents
- INPI CD, CD containing dossiers from INPI for which EPO performs searches.

5.1.7 Paper Storage

The Paper Storage function supports the management of where paper is located. Normally as soon as the CD with the scanned documents is loaded, the paper will be placed batch by batch in a box. This box is then stored in a storage warehouse. The management of this warehouse is out of the scope of the phoenix paper management. Only the fact that it is there is recorded.

5.1.8 Data manipulating processes

The data manipulation processes are performed by the functions

- Mailbox
- Dossier
- Team Manager
- Manager
- Print Jobs
- Print Queue
- Select Team

5.1.9 Mailbox

The Mailbox function supports the viewing and managing of messages that are assigned to the current user in the current team.

The following actions may be performed on messages:

- open; open the Dossier Viewer application for the dossier linked to this message
- close; set the status of the message to closed (message is handled)
- forward; removes the message from this mailbox and forward it to someone else.
- reply; removes the message from this mailbox and send it back to the sender.
- delete; removes the message from the mailbox (logical delete)
- print/close; prints the documents associated with the message and closes the documents in ePHOENIX (used for non-ePHOENIX dossiers).
- print paper file; opens the Print Paper file dialogue in order to create paper file for dossier without opening the dossier.
- refresh; refreshes the content of the current mailbox.
- get legacy data; gets legacy data for each of the selected messages.

A user may be defined in several teams, and can switch teams and can see all of their messages from all teams.

Users can also see all messages for the team that are not assigned to any team member. They may assign one or messages to themselves, and these messages will thenappea in their own mailbox.

5.1.10 Dossier

The Dossier function supports the viewing and managing the contents of an electronic dossier. The functions can be accessed directly by typing the required dossier number, or by opening a message in the mailbox.

The content of the dossier is split in several groups:

- cover; contains general information that used to be on the cover of paper dossiers.
- table of contents (TOC); contains a list of all documents that are stored for this dossier.
- checklist; contains several checklists that help users to control the overall procedure.
- status; contains information from the legacy system (external data not stored inside ePHOENIX)
- structure; shows the structure of the dossier such as procedural groups and participants within groups.
- history; shows the access history to the dossier and action history on the dossier.
- image viewer; displays the images from documents.

The following actions can be performed on a dossier:

- paper file print
- open a legacy application (by means of emulator) using the current dossier number.
- add/edit annotations
- add actions.
- add document(s)
- get or remove permanent ownership of the dossier.

For each document in the dossier, the following set of actions can be performed. Some of the actions are restricted to certain kind of documents:

- change status of document (open/closed)
- add/remove to viewer
- view all document details
- print document or part of document
- send document(s) to client
- add predefined overlays to document (stamp)
- copy to one or more dossiers
- create an exact copy within dossier (clone)
- move to another Participant within dossier
- join the document with one or more other documents within the dossier
- split the document in one or more documents.
- indicate that the document is handled.

A message to a team or team member may be created from the dossier function.

5.1.11 Team Manager

The Team Manager function supports the team managers in managing a team. It allows distribution of unassigned messages to team members. It also gives an overview of all members of the team.

The following actions can be performed for one or more messages in the team mailbox:

- assign to a team member.
- forward to another team or member in another team.
- close
- get legacy data.

Team members can be added and removed from the team, member details can be edited, and the following information can be displayed for each member:

- number of unopened messages
- number of pending messages
- print jobs
- complete list of messages.

Messages that are in a members mailbox can be re-assigned to another member or put back into the team mailbox.

5.1.12 Manager

The Manager function is function used only by support desk people. It supports the editing of the metadata that is stored in the ePHOENIX database. The function allows to create, read, update or delete metadata. The following metadata items can be manipulated:

- algorithms for distribution
- document codes
- document sets
- functions that require user privileges
- global parameters
- labels that are visible on cover page of dossier
- overlay layouts for standard overlays
- paper file types
- procedures including cover flags, checklists and status screens
- user profiles
- ranges for distribution
- teams
- users.

5.1.13 Print Jobs

The Print Jobs function supports the overview of the user's print jobs. All possible types of job are displayed.

5.1.14 Print Queue

The Print Queue function supports the management of paper file print jobs. Each job can be monitored. The following stages can be monitored:

- waiting for print manager
- submitted for printing
- printed
- error
- automatically triggered

5.1.15 Select Team

The Select Team function allows users to be member of multiple teams. The user may switch between teams using this function. The user may have different roles in each team.

5.2 Business dependencies

The ePHOENIX server has some business dependencies to other systems. ePHOENIX does not store any procedural data on the dossiers; however, it integrates the views of ePHOENIX data and the procedural data. Procedural data is typically stored in legacy systems.

Since there is no knowledge in ePHOENIX about the procedural data, but only about the electronic documents, there is as such no real dependency on legacy systems. However there are certain areas where some legacy data is required.

The most important area where legacy data is required is related to the so-called Single Dossier concept. A dossier in ePHOENIX can be linked to one or more keys. The knowledge of this mapping of keys is not built into ePHOENIX but relies on an interface to the legacy systems that store the procedural data.

Procedural data is also used is in the display of procedural data in the ePHOENIX GUI.

Since there is no knowledge in ePHOENIX about the procedural data, but only about the electronic documents, there is no real dependency on legacy systems but there are certain areas where some legacy data is required. The most important one is related to the so-called single dossier concept.

6 Security architecture

6.1 Security services terminology

The International Standards Organisation (ISO) document ISO 7498-2 defines a security architecture within the OSI basic reference model. Within that architecture, the following basic security services and corresponding PKI-based mechanisms relevant for *epoline*® are defined:

- confidentiality, which prevents disclosure of the information to unauthorised individuals, entities or processes, which can be achieved using encryption (encipherment).
- authentication provides verification of a claimed identity and can be divided into:
  - peer entity authentication, used during online data transfer, which can be achieved by a challenge-response mechanism using digital signatures and is either
    - one-way
    - mutual
  - data origin authentication, which provides corroboration of the source of a data unit, and can be achieved by hashing the data using and applying a digital signature
- data integrity, which provides detection of any modification of a data unit, is usually achieved by hashing the data unit and then applying a digital signature
- non-repudiation, which prevents or discourages a communicating party (sender or receiver) from falsely deny sending or receiving the data or its contents, exists in two varieties:
  - non-repudiation with proof of origin, also known as "electronic signature", protects against any attempt by the sender to falsely deny sending the data or its contents, where service also based on hashing and digital signatures, but may also require additional mechanisms, such as time-stamping
  - non-repudiation with proof of delivery, also known as "receipt", where the sender of data is provided with proof of delivery of data in order to protect against any subsequent attempt by the recipient to falsely deny receiving the data or its contents
- key management, which deals with the secure generation, distribution, authentication, and storage of keys.

Other non-PKI based services can be built based on these PKI-based services and mechanisms:

- authentication of the sender, which is a necessary prerequisite for access control and accountability.
- access control, which is used to determine whether the sender is authorised to use the required resource.
- accountability, which ensures that the actions of an entity may be traced uniquely to the entity.

6.1.1 Security patterns

Security patterns provide techniques for addressing known security issues. Security patterns work together to form a collection of best practices to support the office security policy.

6.1.1.1 Single access point pattern

The single access point pattern describes a system with a single entry point for all requests. This is the only way to gain access to the system. All users or other applications must pass through this entry way.

6.1.1.2 Check point pattern

The checkpoint pattern is a pattern that centralises and enforces authentication and authorisation. It is the responsibility of this mechanism to determine if a user has sufficient privileges to be granted access to the requested resource. The centralisation of this logic enables easier management of application policies and business rules.

6.1.1.3 Role pattern

The role pattern specifies a separation of a user from their privileges. A user is assigned one or more roles and each role is granted one or more privileges.

An extension of this pattern is known as the Role Based Access Control (RBAC). RBAC applies the following concepts:

- inheritance of roles, where privileges are the sum of those granted to your current role, as well as those from associated roles.
- separation of duties, such that real world conditions are modelled, where roles may be mutually exclusive.

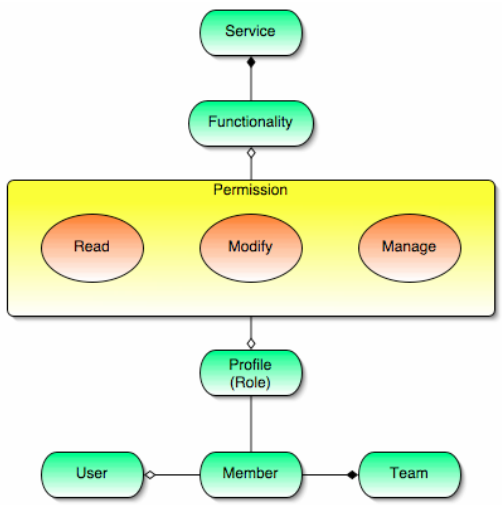

Figure 14: RBAC

6.1.2 Risk assessment and management

Security is risk management. The architectural constraints are to achieve the goal in a secure manner and not to consume unnecessary resources. The following factors influence the risks directly:

- threat, measured by the frequency of attempts
- vulnerability, measured by the probability of success
- cost, measured by the value of resources.

6.1.3 Authoritative data source

The application needs to recognize which is the single authority of data. This implies that the application must be able to find out where the data is coming from and to which extend the data is to be trusted. This can be translated into the premise of validating all information received.

Public Key Infrastructure (PKI) is a public key encryption system that uses digital certificates to verify and authenticate the validity of each party involved in an electronic transaction. The implementation concentrates on assuring that the Epoline Authorisation Service (EAS) data is reliable. Therefore PKI is used for communication between services and the EAS. To avoid performance problems the EAS proxy adds local caching.

All services accessing the backend systems need to establish trusted relations. ePHOENIX works with the DMS data and PXI images stored on our backend. This information is as a rule trusted unconditionally.

To provide optimal functionality and performance in combination with the DMS Light Objects, any ePHOENIX service needs to validate the client-request information. Hence any request must add an encoded request digest. The responsibility for validating the request lies with the EAS proxy.

6.2 Architecture security requirements

- The user should be identified [authenticated] for all services and information that they want to access.
- An authenticated user does not require re-authentication for every service they request [single sign-on].
- The user should be granted access [authorised] to services [application/products] according to the entitlements [roles and profiles] given to them during the user enrolment procedure.
- The system should support confidential [encryption, SSL] transport of information.

The current ePHOENIX security architecture is based up on a central component called the PHOENIX Authentication Server (AS), which is an integral part of the PHOENIX GUI Server.

6.3 Infrastructure supporting ePHOENIX security

In order to access any ePHOENIX information, a user must be defined on the following layers:

- Workstation/LAN (Win2000): A user needs to log on locally to their WS. This layer is not used in the current ePHOENIX implementation.
- Application (DMS): A user must be defined in the ePHOENIX DMS database. These definitions include group membership and profiles.

6.4 ePHOENIX access control model

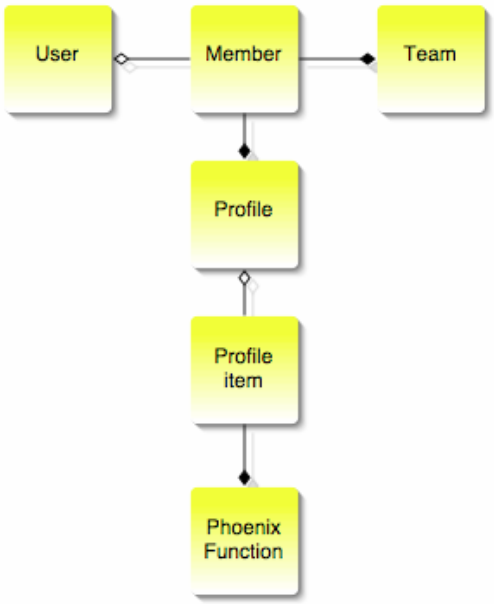

Figure 15: ePHOENIX access control model

Users are grouped as members of one or more teams. Users can only execute ePHOENIX functionality as a member of a team.

Each member is assigned an ePHOENIX profile. A profile is defined as a collection of ePHOENIX functions, by means of Profile Items). ePHOENIX functions typically refer to the GUI functionality, such as creating a batch.

6.4.1 Logon protocol

Madras is the single sign on point. Whenever a user logs on, a call is made to the access server (AS), passing it the user credentials, which currently are the user id, password, and application.

The AS delegates the authentication to the DMSService.

The AS returns a User Authenticator object to the caller, while maintaining information about the user, including their password, in the user`s local memory cache. This UserAuthenicator is only valid for a predefined length of time.

The AS also defines high-level, hard coded, permissions such as MANAGE_MAILBOX. These permissions link ePHOENIX server components to a set of Phoenix functions defined in DMS.

6.4.2 Subsequent requests

All subsequent calls to the Phoenix server include UserAuthenicator. This avoids going through the user ID and password process for every call that needs to access the backend.

Functionality in the Phoenix GUI Server needs one or more permissions. Before executing any request from the user, the GUI server ask authorisation from the AS. The AS verifies that the user has sufficient privileges. Failure to pass this test leads to a Security Exception.

The philosophy used by the current ePHOENIX implementation is not to trust any request from the Web Server (WS). A request to modify persistent data is validated against the backend as the authoritative source before proceeding to execution.

7 ePHOENIX Jini Service Framework

7.1 Jini Service Framework

The Jini Service Framework (JSF) is a layer separating the Jini world from the developers at the EPO It administers the dynamic behaviour of Jini services, can start and stop services or even arrange them into different groups at runtime.

The JSF allows developers to create Jini services. Developers need to have knowledge of only a single class to create a Jini service, that class is the `JiniService` class for creating a service and the `ServiceProvider` class to find services for client applications.

Besides administering Jini services, the JSF also handles load balancing.

7.1.1 Jini programming model

The Jini programming model provides a set basic building blocks for distributed applications: leases, distributed events and downloadable proxies. The building blocks are used in the centrepiece of Jini, the lookup service, which is a directory where service proxies register themselves and where clients search for what they need.

When a service registers itself with a lookup service, it receives a lease on the registration. If the service doesn't renew the lease, say the service is disconnected from the network, the registration is automatically discarded from the lookup service. The lookup service can be seen as a meta-service that provides access to all other services.

7.1.2 Services

When deploying a service on the network a group list must be specified in the service's property file. The services are joined with the LUS located somewhere on the network which is configured to accept services in at least one of these group(s).

7.1.3 Lookup server (LUS)

The Jini lookup server is a primary means for applications to find services on the network belonging to one or more groups The lookup service can be seen as a meta-service that provides access to all other services.

Each LUS must be configured for one or more groups specified as public, which means that every service on the network can register at it.

7.1.4 Proxies

All communication goes through proxies, which are local objects that implement some well-known interface, such as LoadBalancingProxyInterface. Since Jini proxies are written in Java, Jini also claims operating system independence. This in contrast with other service location protocols, which usually use non-portable device drivers.

Proxies can be simple Remote Method Invocation (RMI) stubs that marshal (write in a stream of bytes) objects over the network. They can also implement part of the functionality in the proxy itself. Some services do not require network communication at all, in which case the proxy alone implements the service. The receiving JVM will convert the stream of bytes back into a java.lang.Object subclass if the related code as specified in the code base annotation in the marshalled object is downloaded.

7.1.5 JSF Core

This module is the main component of the Jini Service Framework, under which the JSF services operate.

8 ePHOENIX Jini services and interdependencies

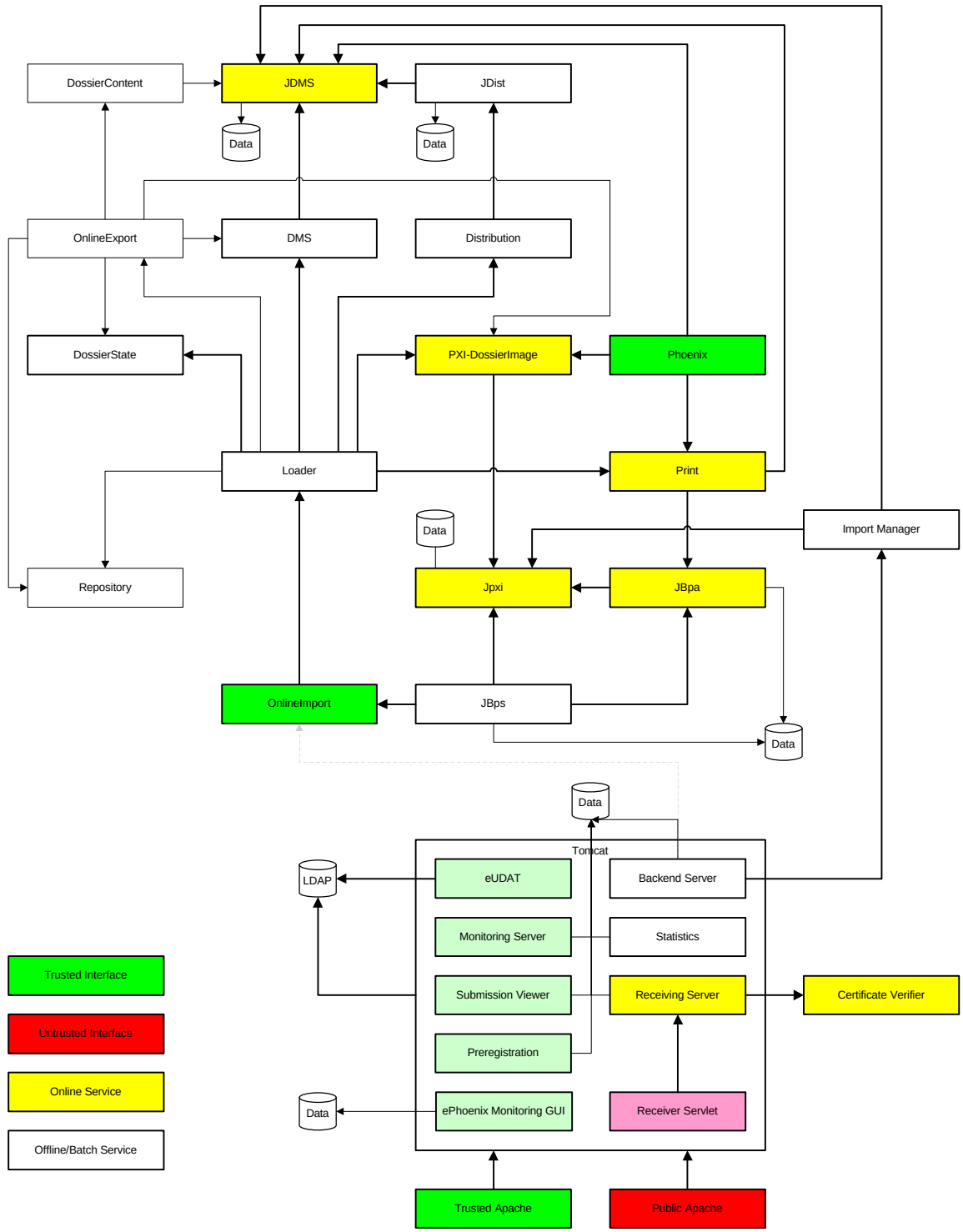

Figure 16: ePHOENIX JSF services and interdependencies

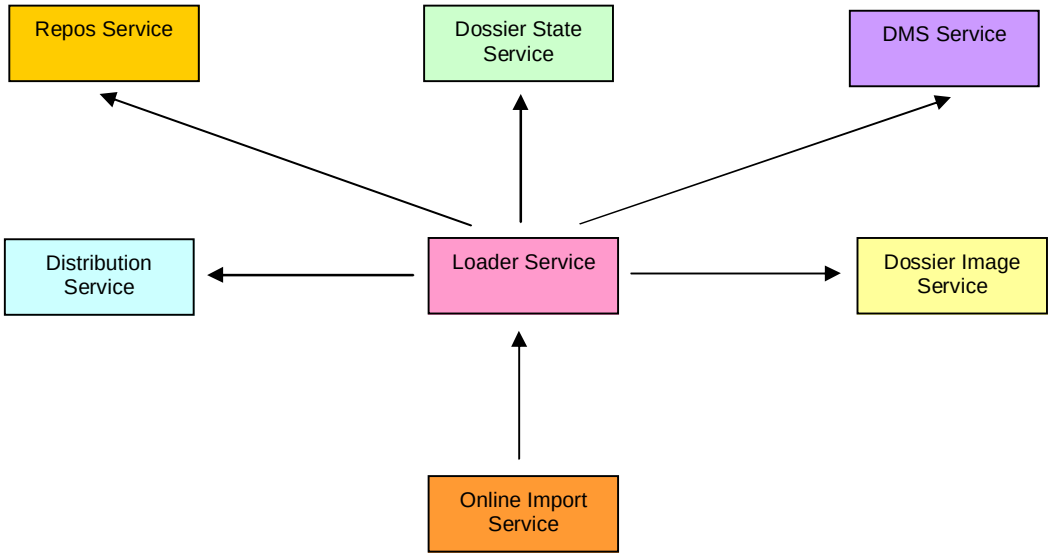

Figure 17: subset: import manager services

9 ePHOENIX Jini services

The ePHOENIX Jini services consist of

- backend services that connect to backend systems
- processing services that run as daemons without interaction with other services or applications
- normal services that are used by other services or applications.

9.1 Backend and processing services

9.2 DMSService

This service enables high-level update/retrieval functions on the DMS database. The DMS classes are not exposed at the interface of the service. There is no unit of work support at the interface of the service.

9.3 DossierService

This service combines the functionality of

- the DossierStateService,
- the DossierImageService
- the DossierContentService.

As extra functionality it implements the filter for the dossier documents for internal and public users and owners of dossiers. This filter is pluggable. A property is used to specify the specific instance of the filter to use.

9.3.1.1 DossierStateService (DSS)

This service is able to retrieve legacy data for a dossier. It is retrieve only, so no updates are supported. This service is also responsible for identifying dossier numbers. A dossier has a set of attributes. The attributes supported are specified in the configuration file.

9.3.1.2 DossierImageService

This service is able to retrieve images from and load images into the PXI system. It supports the retrieval of the images in PDF and TIFF format. It can retrieve single pages, ranges of pages or complete PXI documents.

9.3.1.3 DossierContentService

This service can retrieve DMS information about dossiers. The DMS classes are not exposed at the interface of the Service. A dossier has a set of attributes and contains a set of Documents. Each Document also has a set of attributes. The attributes supported are specified in the configuration file.

```
<Item Name="DossierAttributes" Env="ALL">
      <Attribute Name="Number"       Type="String"  Method="getNumber"/>
</Item>

<Item Name="DocumentAttributes" Env="ALL">
      <Attribute Name="docCode"             Type="String"  Method="getDocCode"/>
      <Attribute Name="descriptionEN"       Type="String" Method = "getDescriptionEN"/>
      <Attribute Name="descriptionFR"       Type="String" Method = "getDescriptionFR" />
      <Attribute Name="descriptionDE"       Type="String" Method = "getDescriptionDE" />
      <Attribute Name="isIncoming"          Type="Boolean" Method="isIncoming"/>
      <Attribute Name="isOutgoing"          Type="Boolean" Method="isOutgoing"/>
      <Attribute Name="isInternal"          Type="Boolean" Method="isInternal"/>
      <Attribute Name="isPublic"            Type="Boolean" Method="isPublic"/>
      <Attribute Name="isViewed"            Type="Boolean" Method="isViewed"/>
      <Attribute Name="isEP"                Type="Boolean" Method="isEP"/>
      <Attribute Name="isDisclaimer"        Type="Boolean" Method="isDisclaimer"/>
      <Attribute Name="legalDate"           Type="Date"    Method="getDate"/>
      <Attribute Name="procedure"           Type="String"  Method="getProcedure"/>
      <Attribute Name="source"              Type="String"  Method="getSource"/>
</Item>
```

9.3.2 jDMSService

This service acts in the ePHOENIX world as a replacement for the CICS layer. The service is activated by specifying `db.connection.path = JINI.` The service itself then makes a connection to an Oracle or DB/2 database.

9.3.3 JDistService

Manages the distribution of messages within ePHOENIX to teams and individuals, using one of two predefined algorithms. May be extended by an NO to support additional algorithms.

9.3.4 JPXIService

ePHOENIX image archive. Please see 0 PXI and 4.2 PXI.

9.4 Normal services

9.4.1 RepositoryService

This service enables storing and retrieving data. The data is kept persistent in the file system. The service provides session support in order to allow to add or delete multiple items as one unit of work. The service handles multiple repositories in parallel.

9.4.2 LoaderService

This service is responsible for loading images into ePHOENIX. The service is derived from the Repository service, so it also has the full functionality of that service. Because it is derived from the RepositoryService it is required to set a Jini Service Framework (JSF) attribute. This attribute is `REPOS_KIND` and has a value `REPOS_IN`.

It controls the main functions of the Import Manager and coordinates the following tasks relating to documents that need to be loaded:

- Creation of DMS Document
- Updating of DMS Document
- Loading Image Data
- Updating Package status
- Distribution of Message
- Printing of document
- Backup Document
- Extracting Document after loading.

The following tasks can be switched off:

- Distribution of Message
- Printing of document
- Backup Document
- Extracting Document after loading.

9.4.3 CacheService

This service caches objects in memory. The objects stay only a limited time in the cache. If the service is stopped, the objects are no longer memorized (no persistency).

9.4.4 OnlineImportService

This service can accept images to be loaded into ePHOENIX. It has session support, so it can be used for all or nothing, like transactions. This is the service that is used by other applications such as FAX Server, Online Filing Server, CASEX and CD-Loader.

9.4.5 OnlineExportService

This service can extract image data from ePHOENIX and store it in a repository.

9.4.6 JDistributionService

This service sends the data, for a message that needs to be distributed, to jDIST for distribution.

9.4.7 PrintService

This service can be used to print any document from PXI. It also supports all the management calls to deal with print jobs. Paperfile printing is additionally supported by this service.

9.4.8 JBPAService

This service is used to configure and process the formats to be applied to the printed document, such as

- layout
- paper size.

9.4.9 JBPSService

This service is used to define the final output as either a printed paper document or as a ditrectly printable file in electronic format.

MADRAS is the Graphical User Interface (GUI) of the system. It performs validation of the user and controls access according to the modules that are relevant to that user.

The PHOENIX server is the core application module which governs all of the services within ePHOENIX.

10 Non-Jini services

10.1 Import/export manager

Only the import manager is started. The Remote Method Invocation (RMI) version is used, for compatability with eOLF.

10.2 Monitoring GUI

This is a Tomcat Web Service, which allows most of the ePHOENIX services to be monitored

Glossary

Glossary	
Abbreviation	**Meaning**
AS	Authentication Server
BSI	Back-file Conversion Service Interface
DMS	Dossier Management System
DSS	DossierStateService
EAS	epoline Authorisation Service
GIM	General Information Manual
GUI	Graphical User Interface
ISO	International Standards Organisation
JSF	Jini Service Framework
LUS	Lookup Server
MADRAS	Mother of All Dossier Related Application Systems
PCT	Patent Co-operation Treaty
PKI	Public Key Infrastructure
PXI	ePHOENIX Image Archive
RABAC	Role Based Access Control
RMI	Remote Method Invocation
RO	Remote Objects
TOC	Table of contents
WS	Web Server

Index