Winter '92 Task Force
Docuvision™ IP - Feasability Evaluation

Print and Distribution Services

Preliminary Draft

R039J5:RA48 V1.0w [8 February 1993]
IA Corporation F-Meylan

Abstract

This paper is a contribution to the Winter '92 Task Force whose objectives are to put under detail scrutiny the available software, hardware and functional modules in order to assemble solution packages which can be rapidly available.

Docuvision™ IP (Integration Package) is one of the targeted packages; this package should comprise a sufficiently complete set of functions to meet most electronic document imaging and management systems. Hybrid solution, Docuvision™ IP is not a product, but a clearly identified set of components with fixed functional levels, interfaces and hardware support providing for further customization - which can be done by the client himself !

This paper approaches the problem of document distribution, spooling and printing. For this, it first analyses existing modules which have been developed during these last years by IA Corporation's Operations Staff and installed in approximately half a dozen sites. The analysis identifies the functionality provided, the quality of the interfaces and ease-of-use of the existing applications, libraries and software templates. Based on this analysis, the possibility of merging these services into Docuvision™ IP building blocks is then discussed.

Table of Contents

table_of_content_will_replace_this_string

Chapter 1
Introduction

Distribution is a master word in electronic document management systems, since such systems handle live documents which are constantly being handled.

<Section intentionnaly left blank in this preliminary draft>

Chapter 2
The Print Server

This chapter is a discussion of the Print Server. It first gives a general overview of the Print Server, indicating in particular the functionality it provides. A discussion of its design principles is then provided. Follows a discussion on the reusability of the Print Server as a building block for futur systems, in particular for Docuvision IP.

2.1. Print Server Overview

The Print Server (“Serveur d’Impression”) was designed to fully handle batch printing. The functionality it provides can be summarized as follows :

◊ Print Ordering

Applications transmit print orders indicating at least what document should be printed. Table 2-1 lists the various attributes which may be specified; all but the document’s reference are optional. Default behaviours are provided for missing attributes.

Table 2-1. Print Order Attributes

DocumentReference	Reference of the document that needs to be printed
PaperFormat	Paper size and orientation properties
FileCoding	File format and document encoding scheme
CoverPage	Reference of a cover page template and associated fill-in data
PriorityLevel	Printing priority level
PrintCopies	Number of copies of the document that should be printed
User	User who issues the print order
ProcessingCenter	Computing platform where the print ordering will be handled
Printer	Printer on which the document whould be printed

◊ *Survey and Management of Print Jobs*

Administrators and applications can track and manage print jobs. Table 2-2 lists the various informations which may be displayed or acted upon. Error tracking and system administration can be supported through a RDMS in which status and error reports can be redirected

Table 2-2. Print Job Attributes

RequestIdentifier	Handle of print job used to uniquely identify it
DepositServer, DepositDate	Server who first handled the print request and date and time at which it was received

RemoteServer	Server in charge of the print job processing
FileCount, FileList	Number and list of all the files that make up the document
PageCount, PrintCount	Total number of pages in document and number of currently printed pages
PrintedCopies	Number of copies of the document that have been printed
StartedOn, FinishedOn	Date and time at which the printing was started and finished
DeletePostPrinting	Toggle for automatic deletion of the print job once the printing has been accomplished
Status	Status of the print job, including error status

◊ *Management of the Available Printers*

The server needs to know what printing ressources are available; instead of simply maintaining a list of these ressources and of those locations, additional information is also management in order to provide processing optimisations and the above-mentionned default behaviours. The additional attributes include the supported paper sizes and file encodings, and the printing speed.

The Print Server has already been used half a dozen times in such projects as AIDA, COPEDOC, ERUDIT, INIST and SIRPA. Because of the inherent structure of the Print Server (See 2.2. “Print Server Design”), it is not possible to say that version n has been used on system x; each system has a different Print Server. This is mainly due to the interfaces of the Print Server with the other components of the system.

The Print Server has been developed and is maintained by the same person, Marie Guiramand. Didier Fort was completely involved and responsible for the Print Server during several months and is the only other person who has a sufficient knowledge level on this software module.

2.2. Print Server Design

Figure 2-1 gives the context diagram of the Print Server. Applications send print job requests, or commands, to the print server; their are four basic commands to add, delete, modify, and query print jobs currently queued by the Print Server. In order to process a command, the Print Server dialogs with the system’s Corpus Server, who handles the storage and retrieval of documents from optical or magnetic media. Processed commands result in the forwarding of a print job to the appropriate Ressource Pilot who is in charge of driving the printing peripheral to do the actual printing. Status and Error Reports are generated for system administration and statistical purposes. An exploded view of the Print Server is given in Figure 2-1 and described in the following paragraphs.

Figure 2-1. Print Server Context Diagram

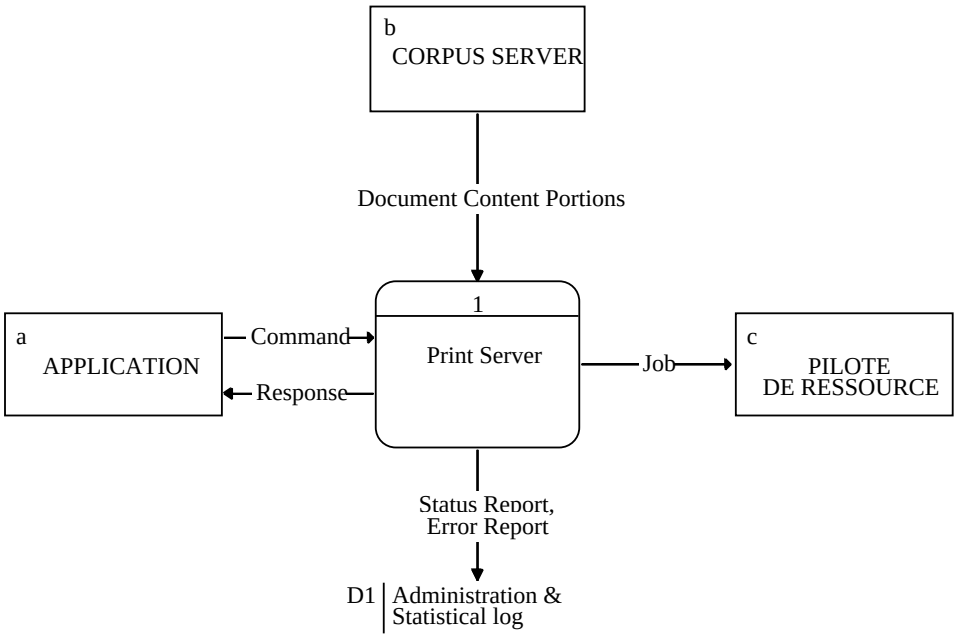
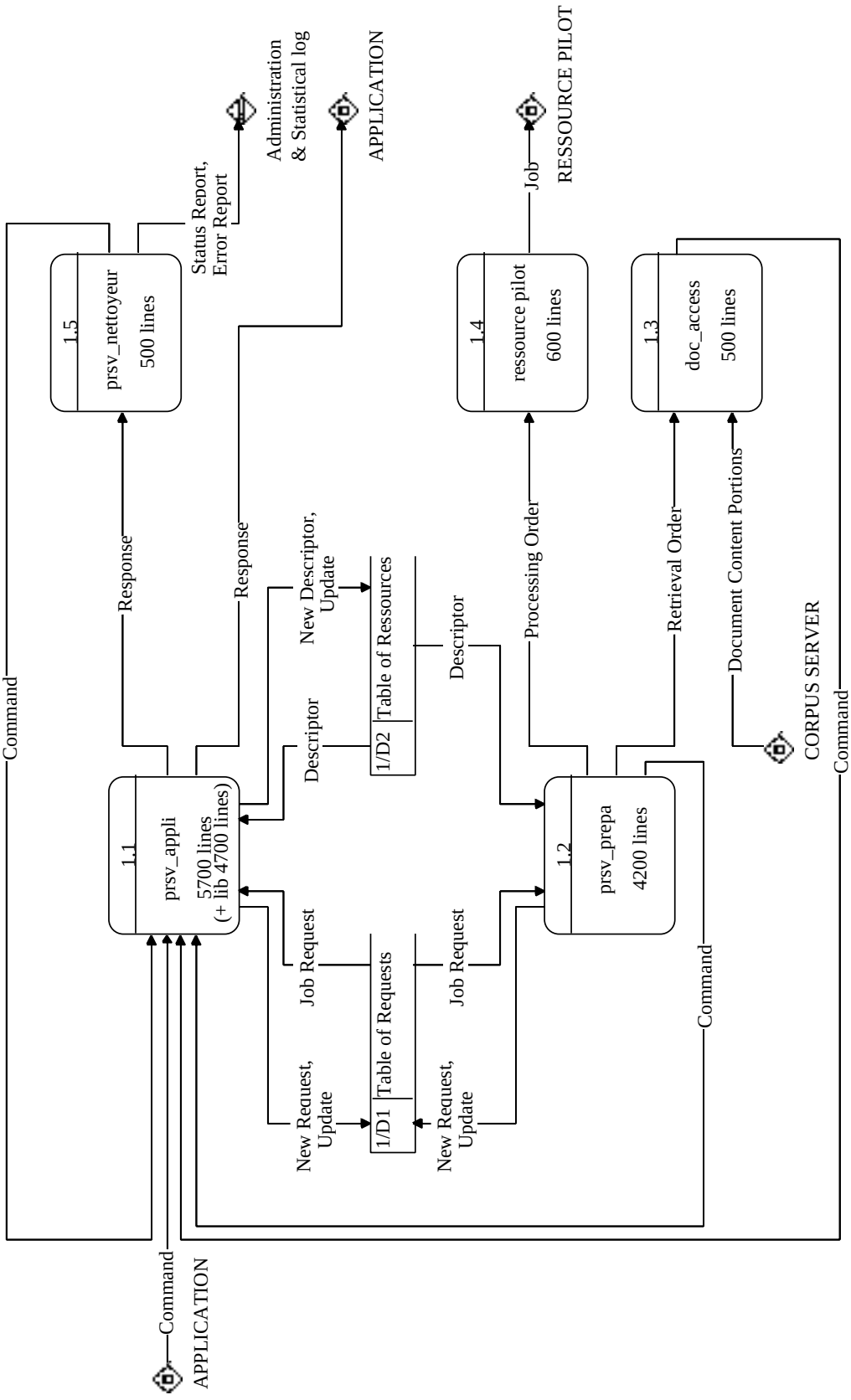


Figure 2-2. Print Server Model



The Print Server as such consists of three processes, prsv_appli, prsv_prepa and prsv_nettoyeur. Communications between processes is done using Unix mailboxes. The prsv_appli process is in charge of handling all print requests; it performs the lexical analysis and updates the request table. The prsv_prepa process is responsible for the processing of the requests; this includes routing from one server to another, retrieving the document corpus files and driving the printer pilots. These two processes have not changed from one implementation to another.

A special daemon process, called prsv_nettoyeur, is in charge of clearing processed print jobs; this is done at fixed time intervals set as a command line argument when launching the daemon. It also provides the status and error reports needed for systems administration and statistical control. This process is provided as a source code skeleton adapted by each project team; the essential reason for doing this is that the destination of the output reports is not known. Reports are generated using either time or job status discriminations and forwarded, in the current implementations, to an Oracle or Ingres table through an SQL order.

The printer pilots are dedicated device driving processes; they support scaling, rotation and format conversion. Scaling is done using the bit throwing algorithm; this enables to offer all possible scaling variations but does not garantee the quality or the resulting scaled image. Rotation is provided in steps of 90°. Table 2-3 lists the format conversions which are currently supported by the Print Server. Table 2-4 lists the various printer pilots which have been implemented for each Print Server installation. The development of new ressource pilots is facilitated by a source code skeleton. Uptill now most pilots have been developped by Marie Guiramand in the course of a project; the only exception being the INIST project where the pilots were designed by Philippe Rerole.

Table 2-3. Format Conversions Supplied by the Print Server

	Output Format		
Input Format	*Bitmap*	*Group IV*	*CBV*[a]
ASCII[b]	Yes	Yes	Yes
Group IV	Yes	Yes	No
PostScript	Yes	No[c]	No
SunRaster	Yes	Yes	Yes
TRIF	Yes	Yes	Yes

a Compact Block Versatec.

b ASCII conversion is supported for the automatic generation of cover pages using the Print Server's cover page templates facility.

[c] PostScript to Group IV conversion could be supported using third party products.

Table 2-4. Currently Implemented Printer Pilots (Sheet 1 of 2)

Project	*Printer Pilot*	*Server Platform*	*Input Format*	*Output Format*	*S*
AIDA	Fujitsu 3723	Sun 3 Sun IBoard	TRIF	Bitmap	No
	Fujitsu 3723	Sun 4 IFF3 (old)	TRIF	Bitmap	Yes
	Ricoh	PC LaserPrint	TRIF	Bitmap	No
	Versatec 400 dpi		TRIF	CBV	Yes
COPEDOC	Fujitsu 3723	Sun 3 Sun IBoard	TRIF	Bitmap	No
	Fujitsu 3723	Sun 4 IFF3 (new)	TRIF	Bitmap	Yes
	Fujitsu 3723	Sun 4 PRDS	TRIF	Bitmap	Yes
	Versatec 200 dpi		TRIF	CBV	Yes
ERUDIT	Fujitsu 3743	PC Kofax	Gr. IV	Gr. IV	No
INIST	Fax Group III	PC	Gr. IV	Gr. IV	
	Fax Group IV	PC	Gr. IV	Gr. IV	
	Fujitsu 3743	PC Kofax	Gr. IV	Bitmap	No
	Fujitsu 3743	PC Kofax	Gr. IV	Bitmap	No
	Fujitsu Luna II	UCI PRDS	Gr. IV	Bitmap	
INPI	Fujitsu 3723	Sun 4 PRDS	Gr. IV	Bitmap	Yes
	Fujitsu 3743	Sun 4 PRDS	Gr. IV	Bitmap	No
	Fujitsu 3743	PC Kofax	Gr. IV	Bitmap	No

SIRPA	Fujitsu 3743	PC Kofax	Gr. IV	Bitmap	No

The retrieval of documents is done through a process called doc_access. This process has been rewritten for each project. Initially provided as a source code skeleton, the doc_access process is intimetaley dependant on the retrieval software used in the project. Since the INIST, INPI and SIRPA projects all relied on the same Corpus Server (“Serveur de Corps”), one would expect the doc_access process to be the same in all those systems. This is not the case.

A standard library is available for applications to access and use the Print Server. It supports five verbs (ADD, MOD, DEL, MOV, LIST) which can be applied either to print job requests or to printer ressource descriptors.

The data manipulated by the Print Server is stored in two sequential files; one for the printer ressources and another for the print job requests. The access to records within these files uses home-made functions. Furthermore, each ressource pilot has its own configuration file.

All together, the Print Server consists of approximately 15,000 lines of C code plus another 600 lines per printer pilote. It has been built with a unique library - of approximately 5,000 lines. The source codes skeletons representing about 1,000 lines. All this code is exclusively Unix based.

The computing platform requirements are those common to the majority of our system developments : Unix environment using the ARPA TCP/IP protocol suite including, in particular, FTP (File Transfer Protocol).

2.3. Print Server Analysis

The Print Server can really be considered as a building block. The major problems faced in the reusability of this software modules are twofold; firstly in its intended use, and secondly in its interface with other, presumably standard, Docuvision™ IP modules, in particular with the storage and retrieval module. Minor adaptations need to be made; these can be done as part of futur projects. These conclusions are further discussed below.

2.3.1. Intended Use of the Print Server

The Print Server is a generic spooler which allows the sharing of various ressources in a distributed environment. These ressources may be convenience, departemental or central printing utilities. These may also be any device or program requiring the manipulation of objects stored within the system. In the Erudit system, for instance, the Print Server is used to dispatch documents to workstations for direct manipulation by users. In the Inist project, The Print Server is used to forward documents which need to be transmitted using Group III or Group IV telecommunication peripherals.

From a conceptual standpoint, the Print Server should be able of handling jobs for the processing of objects without knowing anything about these objects. The only necessity being to know :

◊ where the object is located,

◊ who initiated the job request,

◊ to what pilot the job request should be forwarded,

◊ what are the processing requisites (eg. priority level, preferred dispatch mode, activity logging, etc.).

The Print Server meets these conditions but for one main assumption : the objects manipulated by the Print Server are docuuments which are necessarily retrieved through the doc_access process. This would have been further studied in this report if we were Docuvision™ IP was intended to be a long term offering.

The Print Server does batch processing. This might seem as an handicap; I am convinced it is not. In all the systems installed in Europe so far by IA Corporation, I do not see a single system that cannot accomodate such processing.

2.3.2. Print Server Interfaces

The Print Server needs to communicate with the rest of the system; this requires clearly identified interfaces, as shown in Figure 2-1. In all past implementations of the Print Server, skeleton source codes have been provided to do this interfacing. This was because the system components were presumably non-standard. These skeletons have not always been appreciated and, even though well documented through Unix man pages, are not the ideal approach for client-manipulated interfaces. Furthermore, since Docuvision™ IP will typically be composed of standard modules with clearly identified functionalities and interfaces, the flexibility provided by these skeletons is no longer necessary.

Discarding the skeleton source codes implies making certain assumptions; these are listed below.

◊ *Interface with the Corpus Server (“Serveur de Corps”)*
(currently supported through the doc_access skeleton)

The Print Server must be capable of transferring to the Corpus Server the attributes past by the application that indicate the object or subset of component portions which need to be retrieved; for instance, the application may post a print job request for pages 1 to p of document d. The Corpus Server must be capable of indicating where and how these objects have been downloaded. The interface must of course provide a complete handshaking and error handling protocol.

This interface need not be redesigned from scratch. All that needs to be done is to reuse the doc_access process developped for Corpus Server-based system and use it’s interface protocol as the standard interface for futur systems. Hence the doc_access becomes a standalone binary code which can be incorporated into the Print Server kernel along with the prsv_appli and prsv_prepa programs.

◊ *Administration and Statistical Logging Interface*
(currently supported through the prsv_nettoyeur skeleton)

The reason for having a skeleton here is that it is never none beforehand what RDBMS system will be used and in what tables the status and error reports should be deposited. Furthermore, an underlying intention of the prsv_nettoyeur skeleton was to enable custom statistical functions.

The suggested approach is to provide standard Unix-like logging where detailed reports are appended to a same and clearly identified ASCII file. This implies that the prsv_nettoyeur process is no longer dependant on the rest of the system and, as for the doc_access process, can become part of the Print Sever Kernel. This is not currently available; however this an extremely simple update : we are reudcing functionality !

The skeleton can be kept as an alternate method for custom developments made directly by IA Corporation. It is not advised to provide this skeleton to clients since basic Print Server functions are accessible and could be corrupted.

◊ *Ressource Pilot Interface*
(currently provided through the imprimeur skeleton)

It is assumed that the development of new ressource pilots will necessarily be carried out by IA Corporation. There are three main reasons for this. Firstly, it is unreasonable to conceive a generic ressource pilot that can cope with any possible ressource - we wouldn’t exist if that was the case ! Secondly, the number of new pilots we will need to include in the next six to nine months is presumably nill. And thirdly, any incorporation of a new ressource pilot is added commercial value to the Print Server.

◊ *Application Interface*
(currently provided through the prsv_pabel skeleton)

Access to the Print Server is done using commands which are contained in a C library and defined in a standard C header (prsv_interface.h). The prsv_panel skeleton provides a short, simple and well documented template which could be provided as such to clients for them to develop their own application. No change is planned here.

2.3.3. Print Server Enhancements

Table 2-5 lists the major enhancements which should be planned. They do not require a specific R&D investment, but could be planned as part of futur projects incorporating the Print Server. The main reason for this is that they do not impact the overall design and philosophy of the Print Server. Further these enhancements do not have a major cost impact and can be financed directly by the project. Finally, some major modifications which need to be made, fully depend on the design choices and implementations that will be made on the Corpus Server.

Table 2-5. Suggested Print Server Enhancements

Enhancement	*Cost*
◊ Standardisation of the Corpus Server access through the adaptation of an existing doc_access source code	
◊ Conversion of the prsv_nettoyeur skeleton into a defintive source code where administration and statistical logging is done in a Unix-like fashion in a predefined ASCII file	
◊ Consolidation of ressource pilot so as to use a unique binary code dynamically linked to the various ressource-dependant libraries	
◊ Support for portion content printing (ie. for a document, printing of page x to y out of a total of z pages)	
◊ Standardisation of ressource pilot error-codes	
◊ Adaptation of home-made records management to accomodate indexing	

Chapter 3

The Distribution Server

The Distribution Server (“Serveur de Diffusion”) was intended for the automated handling of high speed batch printing on industrial printers (Fujitsu Luna II). It has been developed for INPI (Sophia-Antipolis, France) and will shortly be fully operational.

Chapter 4

Document Workflow Considerations

Workflow has received lately a great deal of corporate attention, and development opportunities and plans are currently being evaluated by our american counter-part. My intent here, is not to compete in any way with these undergoing activities. Based on the evaluation of the Print Server and Distribution Server and using the experienced gained in the Elfos project, I believe we've got the means of providing, on short notice, a simple but robust workflow engine. This is described in this chapter.

This discussion has voluntarily been dissociated from the previous analysis since its implications are a bigger financial investment and a product with a longer life span.

4.4. What is Workflow ?

An aura of mystique surrounds the term ‘workflow’, and I have heard to date, many definitions - often extremely complex. Workflow, at its most basic level, refers to the flow of work through an organization. Document workflow, concentrates on the movement of documents from one processing step to another, with little or no regard to the reasons for the movement nor for the interrelationships between processing steps along the way.

Hence, workflow becomes both conceptually rather simple and not at all difficult to implement. Does this imply that the problem has been over-simplified ? No, but it does rely on one very important axiom :

Most companies already monitor their working procedures electronically.

The above mentionned reasons for the movement of a document from one processing step to another - from one desk to another, and the interrelationships between these processing steps along the way are handled by computer programs. The implications are that the rules that govern a specific flow of work are electronically coded. These need to be enhanced with imaging facilities which will enable the act of transferring a document from location A to location B to be carried out electronically.

It is the rare case that electronic document management systems are installed as totally stand-alone systems. Rather, nearly all such systems require an interface with existing corporate computing ressources - usually a mainframe host. The clients want the added imaging handling features to be controlled by the host - in effect, the electronic management system becomes a sophisticated mainframe peripheral device. Most such companies have made significant investments in mainframe programs which control the flow of work through their organisations. The European Patent Office (EPO), for instance, launched in 1988, a five year automation plan involving over 20 systems in which ELFOS and BACON - with an internal budget of approximately 400,000 DM - were only components !

Re-creating these programs that control the flow of work in an electronic document management (sub)system would be prohibitively expensive and strategically absurd. As an example, the system that controls the working procedures for the processing of patent granting applications at the EPO, called EPASYS, is an IBM mainframe CICS system, started early 1975 when the EPO didn’t yet exist, and in which over 50 man/years have been invested in development and 3 man/years in annual maintenance. The minimal risk of recreating such a system would be the disruption of day-to-day work !

4.5. Implementing Workflow Features

A common misconception is that a special language is required to implement a workflow system. In fact, a fully functional workflow system can be built using the facilities available in any modern programming languages.

When considering the four basic workflow requirements summarised in Table 4-6, one can see that, at the lowest level, workflow queues are nothing more than database tables. Queue entries (ie. descriptors of documents awaiting processing) are stored as records in the table; the attributes associated with the queue entries (eg. priority level, associated documents, status data) are stored in columns (fields) in the records. Associated with the queues are pointers to the images themselves; when the queue entry is retrieved, the system uses these pointers to locate and retrieve the one or several documents (or content portions within these documents).

Table 4-6. The Four Basic Workflow Requirements

Automatic Routing	Work is routed from one knowledge worker to another under the control of the system, following a set of predefined workflow rules.
Pending Documents	During the normal course of work, a knowledge worker may require an as-yet-not arrived or as-yet-not-available document. The current document is placed in suspense awaiting the required document.
Exception Handling	A knowledge worker may find that they are not, for one reason or another, capable of, or authorized to, complete working on a document. There must be a mechanism for them to identify this fact to the system and have the system route it to the appropriate individual.
Manual Routing	Each knowledge worker must have the ability to tell the system where to send a docuent next, regardless of the ‘normal’ routing rules.

As mentionned above (4.4. “What is Workflow ?”), seldom are the companies that do not have programs which control the flow of work through their organisation. These programs contain the procedural data required to define the workflow rules; adding code to these applications, even if mainframe-based, is easily justified.

Implementing workflow features therefore implies to :

1. Add to the RDBMS system handling the procedural data fields allowing the client programs to be ‘aware’ of the electronically available documents;

   This is evidently done by the client and has no consequence except for the definition of Application Programming Interfaces (APIs) that makes the “connection” between the programs request for images and the electronic document management system’s retrieval and distribution mechanisms.

2. Keep track of each knowledge worker’s workload (ie. the pile of documents in his in- and out-baskets and on his shelves)

   As mentionned before, this primarily consists in maintaining queues. Where this is done is not of vital importance; however, since such queues are only usefull to the electronic document management system, these can be maintained in that system using a standard RDBMS and it’s associated fourth-generation language. The tables (queues) which need to be handled for each knowledge worker are his :

   a) In-basket that contains the workload assigned by the client’s program;

   b) Out-basket that contains processed documents and for which procedural updates are necessary in order for the client’s system to determine the next processing step;

   c) Hold-basket that contains pending documents (and for which notification to the client’s system may be necessary);

   d) Associated documents-basket that contains additional reference litterature which the knowledge worker may require to perform the required task on an in-basket document.

3. Provide in the electronic document management system a basic distribution engine which will (pre)fetch documents and place them in the appropriate knowledge worker’s in-basket.

This basic distribution engine can be fully provided by the Print and Distribution Server. This is discussed in the following paragraph: 4.6. “Creating a Workflow Enabler Engine”.

4.6. Creating a Workflow Enabler Engine

Chapter 3
The Distribution Server

The Distribution Server (“Serveur de Diffusion”) was intended for the automated handling of high speed batch printing on industrial printers (Fujitsu Luna II). It has been developed for INPI (Sophia-Antipolis, France) and will shortly be fully operational.

<Section intentionnaly left blank in this preliminary draft>

Chapter 4
Document Workflow Considerations

Workflow has received lately a great deal of corporate attention, and development opportunities and plans are currently being evaluated by our american counter-part. My intent here, is not to compete in any way with these undergoing activities. Based on the evaluation of the Print Server and Distribution Server and using the experienced gained in the Elfos project[1], I believe we've got the means of providing, on short notice, a simple but robust workflow engine. This is described in this chapter.

This discussion has voluntarily been dissociated from the previous analysis since its implications are a bigger financial investment and a product with a longer life span.

[1] W.Novinger, of Rothschild Consultants, intensively participated in the specifications of the Elfos project; he was among the first to give a precise definition of workflow, and the Elfos project is a perfect representation of a system requiring all the features of workflow processing.

4.4. What is Workflow ?

An aura of mystique surrounds the term ‘workflow’, and I have heard to date, many definitions - often extremely complex. Workflow, at its most basic level, refers to the flow of work through an organization. Document workflow, concentrates on the movement of documents from one processing step to another, with little or no regard to the reasons for the movement nor for the interrelationships between processing steps along the way.

Hence, workflow becomes both conceptually rather simple and not at all difficult to implement. Does this imply that the problem has been over-simplified ? No, but it does rely on one very important axiom :

Most companies already monitor their working procedures electronically.

The above mentionned reasons for the movement of a document from one processing step to another - from one desk to another, and the interrelationships between these processing steps along the way are handled by computer programs. The implications are that the rules that govern a specific flow of work are electronically coded. These need to be enhanced with imaging facilities which will enable the act of transferring a document from location A to location B to be carried out electronically.

It is the rare case that electronic document management systems are installed as totally stand-alone systems. Rather, nearly all such systems require an interface with existing corporate computing ressources - usually a mainframe host. The clients want the added imaging handling features to be controlled by the host - in effect, the electronic management system becomes a sophisticated mainframe peripheral device. Most such companies have made significant investments in mainframe programs which control the flow of work through their organisations. The European Patent Office (EPO), for instance, launched in 1988, a five year automation plan involving over 20 systems in which ELFOS and BACON - with an internal budget of approximately 400,000 DM - were only components !

Re-creating these programs that control the flow of work in an electronic document management (sub)system would be prohibitively expensive and strategically absurd. As an example, the system that controls the working procedures for the processing of patent granting applications at the EPO, called EPASYS, is an IBM mainframe CICS system, started early 1975 when the EPO didn't yet exist, and in which over 50 man/years have been invested in development and 3 man/years in annual maintenance. The minimal risk of recreating such a system would be the disruption of day-to-day work !

4.5. Implementing Workflow Features

A common misconception is that a special language is required to implement a workflow system. In fact, a fully functional workflow system can be built using the facilities available in any modern programming languages.

When considering the four basic workflow requirements summarised in Table 4-6, one can see that, at the lowest level, workflow queues are nothing more than database tables. Queue entries (ie. descriptors of documents awaiting processing) are stored as records in the table; the attributes associated with the queue entries (eg. priority level, associated documents, status data) are stored in columns (fields) in the records. Associated with the queues are pointers to the images themselves; when the queue entry is retrieved, the system uses these pointers to locate and retrieve the one or several documents (or content portions within these documents).

Table 4-6. The Four Basic Workflow Requirements

Automatic Routing	Work is routed from one knowledge worker to another under the control of the system, following a set of predefined workflow rules.
Pending Documents	During the normal course of work, a knowledge worker may require an as-yet-not arrived or as-yet-not-available document. The current document is placed in suspense awaiting the required document.
Exception Handling	A knowledge worker may find that they are not, for one reason or another, capable of, or authorized to, complete working on a document. There must be a mechanism for them to identify this fact to the system and have the system route it to the appropriate individual.
Manual Routing	Each knowledge worker must have the ability to tell the system where to send a docuent next, regardless of the ‘normal’ routing rules.

As mentionned above (4.4. “What is Workflow ?”), seldom are the companies that do not have programs which control the flow of work through their organisation. These programs contain the procedural data required to define the workflow rules; adding code to these applications, even if mainframe-based, is easily justified.

Implementing workflow features therefore implies to :

1. Add to the RDBMS system handling the procedural data fields allowing the client programs to be ‘aware’ of the electronically available documents;

   This is evidently done by the client and has no consequence except for the definition of Application Programming Interfaces (APIs) that makes the “connection” between the programs request for images and the electronic document management system’s retrieval and distribution mechanisms.

2. Keep track of each knowledge worker’s workload (ie. the pile of documents in his in- and out-baskets and on his shelves)

   As mentionned before, this primarily consists in maintaining queues. Where this is done is not of vital importance; however, since such queues are only usefull to the electronic document management system, these can be maintained in that system using a standard RDBMS and it’s associated fourth-generation language. The tables (queues) which need to be handled for each knowledge worker are his :

   a) In-basket that contains the workload assigned by the client’s program;

   b) Out-basket that contains processed documents and for which procedural updates are necessary in order for the client’s system to determine the next processing step;

   c) Hold-basket that contains pending documents (and for which notification to the client’s system may be necessary);

   d) Associated documents-basket that contains additional reference litterature which the knowledge worker may require to perform the required task on an in-basket document.

3. Provide in the electronic document management system a basic distribution engine which will (pre)fetch documents and place them in the appropriate knowledge worker’s in-basket.

This basic distribution engine can be fully provided by the Print and Distribution Server. This is discussed in the following paragraph: 4.6. “Creating a Workflow Enabler Engine”.

4.6. Creating a Workflow Enabler Engine

<Section intentionnaly left blank in this preliminary draft>