INFLOW
Structured Print Job Delivery

Claes Buckwalter

2003-10-23
INFLOW
Structured Print Job Delivery

Examensarbete utfört i Medieteknik
vid Linköpings Tekniska Högskola, Campus
Norrköping

Claes Buckwalter

Handledare: Björn Kruse
Examinator: Björn Kruse

Norrköping 2003-10-23
More and more print jobs are delivered from customer to printer digitally over the Internet. Although Internet-based job delivery can be highly efficient, companies in the graphic arts and printing industry often suffer unnecessary costs related to this type of inflow of print jobs to their production workflows. One of the reasons for this is the lack of a well-defined infrastructure for delivering print jobs digitally over the Internet.

This thesis presents INFLOW — a prototype for a print job delivery system for the graphic arts and printing industry. INFLOW is a web-based job delivery system that is hosted on an Internet-connected server by the organization receiving the print jobs. Focus has been on creating a system that is easy to use, highly customizable, secure, and easy to integrate with existing and future systems from third-party vendors. INFLOW has been implemented using open standards, such as XML and JDF (Job Definition Format).

The requirements for ease-of-use, high customizability and security are met by choosing a web-based architecture. The client side is implemented using standard web technologies such as HTML, CSS and JavaScript while the server-side is based on J2EE, Java Servlets and Java Server Pages (JSP). Using a web browser as a job delivery client provides a highly customizable user interface and built in support for encrypted file transfers using HTTPS (HTTP over SSL).

Process automation and easy integration with other print production systems is facilitated with CIP4’s JDF (Job Definition Format). INFLOW also supports “hot folder workflows” for integration with older preflight software and other hot folder-based software common in prepress workflows.
INFLOW
Structured Print Job Delivery

Claes Buckwalter
Norrköping, 23 October 2003

A Master’s Thesis in
Media Technology and Engineering
Supervisor: Professor Björn Kruse

Department of Science and Technology (ITN)
Campus Norrköping, Linköping University
Abstract

More and more print jobs are delivered from customer to printer digitally over the Internet. Although Internet-based job delivery can be highly efficient, companies in the graphic arts and printing industry often suffer unnecessary costs related to this type of inflow of print jobs to their production workflows. One of the reasons for this is the lack of a well-defined infrastructure for delivering print jobs digitally over the Internet.

This thesis presents Inflow — a prototype for a print job delivery system for the graphic arts and printing industry. Inflow is a web-based job delivery system that is hosted on an Internet-connected server by the organization receiving the print jobs. Focus has been on creating a system that is easy to use, highly customizable, secure, and easy to integrate with existing and future systems from third-party vendors. Inflow has been implemented using open standards, such as XML and JDF (Job Definition Format).

The requirements for ease-of-use, high customizability and security are met by choosing a web-based architecture. The client side is implemented using standard web technologies such as HTML, CSS and JavaScript while the server-side is based on J2EE, Java Servlets and Java Server Pages (JSP). Using a web browser as a job delivery client provides a highly customizable user interface and built in support for encrypted file transfers using HTTPS (HTTP over SSL).

Process automation and easy integration with other print production systems is facilitated with CIP4's JDF (Job Definition Format). Inflow also supports hot folder workflows for integration with older preflight software and other hot folder-based software common in prepress workflows.
Contents

1 INTRODUCTION ......................................................... 1
   1.1 PROBLEM STATEMENT ........................................... 1
   1.2 GOAL ................................................................. 1
   1.3 METHOD ............................................................ 1
   1.4 DISPOSITION ....................................................... 1

2 INTERNET-BASED PRINT JOB DELIVERY .................... 3
   2.1 FILE TRANSFER PROTOCOL (FTP) ......................... 3
   2.2 FORM-BASED FILE UPLOAD ................................ 4
   2.3 WEBDAV .............................................................. 5
   2.4 PROPRIETARY PRINT JOB DELIVERY SOLUTIONS ....... 6

3 JOB METADATA ......................................................... 7
   3.1 METADATA FORMATS FOR THE GRAPHIC ARTS INDUSTRY ... 7
   3.2 EXTENSIBLE METADATA PLATFORM (XMP) ............... 7
      3.2.1 XMP and Print Job Delivery ......................... 8
   3.3 JOB DEFINITION FORMAT (JDF) ............................. 8
      3.3.1 JDF and Print Job Delivery ............................ 9

4 DEFINITION OF WEB-BASED JOB DELIVERY SYSTEM ...... 11
   4.1 ARCHITECTURAL MODEL ...................................... 11
   4.2 USE CASES .......................................................... 11
      4.2.1 Job Delivery Client ....................................... 12
      4.2.2 Administrator Client ..................................... 12
   4.3 FUNCTIONAL REQUIREMENTS ................................. 13
      4.3.1 Job Delivery Over the Internet ...................... 13
      4.3.2 Encrypted File Transfers .............................. 13
      4.3.3 User-friendly .............................................. 13
      4.3.4 Job-Centric ................................................. 13
      4.3.5 JDF Job Tickets for Job Metadata .................... 13
      4.3.6 Central JDF Job Ticket File Repository ............. 13
      4.3.7 Job File Repository with Dynamically Built Folder Structure ... 13
      4.3.8 Hot Folder Workflow Support ....................... 14
      4.3.9 E-mail Receipt to Job Deliverer .................... 14
      4.3.10 E-mail Notifications to Job Receiver ............... 14
      4.3.11 Web-based Administration ............................ 14
      4.3.12 Cross-platform Server Solution .................... 14
      4.3.13 Internationalized ....................................... 14
   4.4 TECHNICAL REQUIREMENTS ................................. 15
      4.4.1 Java 2 Platform, Enterprise Edition (J2EE) ............ 15
      4.4.2 Job Definition Format (JDF) ........................... 15
1 Introduction

1.1 Problem Statement
More and more print jobs are delivered from customer to printer digitally over the Internet. Although internet-based job delivery can be highly efficient, printers, and other companies in the graphic arts industry, often suffer unnecessary costs related to this type of inflow of print jobs to their production workflows. Two of the main reasons for these costs are due to:

1. The lack of a well-defined infrastructure for delivering print jobs electronically using the Internet
2. The lack of quality assurance, known as preflighting, of the print job files that are being delivered

1.2 Goal
This thesis focuses on the first issue above — an infrastructure for job delivery over the Internet. The goal was to design and implement a fully functional prototype of a user-friendly, internet-based client-server solution for delivering job files to production workflows in the graphic arts industry.

The prototype, named Inflow, was developed in cooperation with the Swedish software company Frog Software [1]. The intention was that the prototype later be used as a proof-of-concept for one of Frog Software's future products [2].

1.3 Method
Different technologies for Internet-based print job delivery and job metadata where evaluated. Based on this information, interviews with end-users, and the author's previous knowledge of the problem domain a set of requirements were specified. Using these requirements the prototype, Inflow, was designed and implemented.

1.4 Disposition
Chapter two presents and compares common software solutions used for job delivery over the Internet. Apart from print job files, the printer often needs additional information about the print job from the customer. This kind of information is known as job metadata and an introduction to job metadata is given in chapter three.

Chapter four provides a specification for a prototype for print job delivery system and chapter five presents the proposed solution — Inflow.
2 Internet-based Print Job Delivery

In the graphic arts industry efficient job delivery is of key importance. The success of the job receiver’s business to some extent depends on an efficient and controlled inflow of job files from its customers. Companies in the graphic arts industry have a wide range of options when choosing a software solution or technology for handling their Internet-based job delivery.

It is often the party at the receiving end, for example the printer, which is responsible for providing a software solution for Internet-based job delivery. These solutions are typically based on a client-server architecture, with the job receiver hosting a server that the job deliverer connects to with some kind of client software.

This chapter gives an overview of the strengths and weaknesses of the most common Internet-based job delivery solutions available, focusing on the client software.

2.1 File Transfer Protocol (FTP)

<table>
<thead>
<tr>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wide user base</td>
<td>Security</td>
</tr>
<tr>
<td>Inexpensive client software</td>
<td>Usability</td>
</tr>
<tr>
<td>Inexpensive server software</td>
<td>Does not facilitate delivery of job metadata</td>
</tr>
</tbody>
</table>

The File Transfer Protocol (FTP) [3] is one of the oldest and most widely used protocols for transferring files over the Internet and other TCP/IP-networks. Most modern server operating systems ship with the required server software, and almost every all client operating systems come with a FTP client of some sort. In addition, most web browsers support the File Transfer Protocol. Hence, almost all Internet users have access to a FTP client for transferring files.
A serious security weakness with FTP is that passwords are sent in plain text over the network. This makes it easy for hackers to listen in and steal passwords from users to gain access to FTP accounts. It is possible to tunnel FTP through an encrypted network connection. However, there is no standardized way of doing this so support varies between FTP servers and clients. SFTP (SSH File Transfer Protocol) [4] and SCP (Secure Copy Protocol) [5] are the two most common secure FTP-alternatives and rely on SSH.

FTP is a common solution for delivering print jobs. The job receiver hosts a FTP server that the job deliverer connects and transfers job files to. FTP’s strength is its wide spread user base, almost all Internet users have a FTP client.

FTP’s weaknesses are its security issues (mentioned above) and usability. Connecting to an FTP server and transferring files with a FTP client can be daunting task for the casual user. It is not possible for the job receiver to customize the user interface of the job deliverer’s client software. This makes it impossible for the job receiver to place constraints on the job files being delivered or enhance the job deliverer’s user experience. For example, there is no way for the receiver to get job metadata from the deliverer without having the deliverer transfer the metadata in a separate file or in some way embed it in the actual job files.

### 2.2 Form-based File Upload

<table>
<thead>
<tr>
<th>Strengths</th>
<th>All Internet users are familiar with browsing the Web and submitting web forms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Highly customizable user interface</td>
</tr>
<tr>
<td></td>
<td>Facilitates delivery of job metadata</td>
</tr>
<tr>
<td></td>
<td>Security (HTTPS)</td>
</tr>
<tr>
<td></td>
<td>Inexpensive client software</td>
</tr>
<tr>
<td></td>
<td>Inexpensive server software</td>
</tr>
<tr>
<td></td>
<td>Goes through firewalls</td>
</tr>
</tbody>
</table>

| Weaknesses | Primitively implemented in web browsers                                         |
|           | Only one file can be attached per file field                                    |

The web browser is probably the killer application of all time. Virtually all operating systems, even for mobile devices, bundle a web browser. Almost all computer users know what a web browser is and how to operate one adequately.

Most Internet users have downloaded files by clicking links on a web page. This type of file transfer is only one way — from server to client. For web pages to facilitate file transfers from client to server a method called form-based file upload [6] is used. The job receiver hosts a web page containing a form with special form fields for attaching files. When the job deliverer submits the web form to the job receiver’s web server the attached files are submitted together with the rest of the form.

Figure 2-3 Form-based File Upload
Since form-based file upload is based on web pages with forms, the underlying network protocol is HTTP [7] just as it is for the rest of the World Wide Web. Thanks to HTTP’s strong authentication, encryption, and proxy (firewall) support form-based file upload is both secure and reliable.

Form-based file upload has usability pros and cons. Its major weakness is that most web browsers have very primitive implementations of form-based file upload. Rarely, if ever, is attaching files by drag and drop supported. Even worse, web browsers fail to give user-feedback of the progress of the file upload to the server. Another weakness is that the form-based file upload method itself has the limitation that only one file can be attached per form field, and that folders cannot be uploaded.

Despite the weaknesses above, flexibility is form-based file upload’s main usability strength. Since form-based file upload uses ordinary web pages with forms users will be familiar with the user interface. In addition, the job receiver hosting the job delivery web form has infinite possibilities when it comes to customizing the user interface and the information that the job deliverer must submit together with the job files. This way job metadata can easily be submitted together with the job files.

2.3 WebDAV

**Strengths**
- Client support in applications from Adobe and Microsoft
- Client support in Windows operating systems
- Is emerging as a network file system for the Internet
- Security (HTTPS)

**Weaknesses**
- User interface is not customizable

WebDAV [8] is a network protocol and stands for *Web-based Distributed Authoring and Versioning*. It is a set of extensions to the HTTP network protocol and is an open standard that is managed by the Internet Engineering Task Force (IETF) [9]. Originally, WebDAV was designed for collaborative management of web sites on remote web servers. But WebDAV can be used for much more than authoring web pages. Many see WebDAV as a network file system, which makes it suitable for print job delivery.

Since WebDAV is based on HTTP it shares the protocol’s benefits. Strong authentication, encryption, and proxy support comes built in [10].

Although WebDAV has many interesting features it has a few weaknesses when it comes to print job delivery. WebDAV clients share the same usability weaknesses as FTP clients: there is no way of customizing the user interface of the job deliverer’s client. Also, being a general file transfer protocol, WebDAV clients provide no support for metadata specialized for the graphic arts industry.
2.4 Proprietary Print Job Delivery Solutions

**Strengths**
- Optimized for print job delivery
- Often support submission of job metadata

**Weaknesses**
- Job deliverer may need to install extra client software
- Client and server software may be expensive
- Client software might not support all operating systems

There are a number of Internet-based job delivery solutions available on the market today. *Extensis PrintReady* [11], *Adobe PDF Transit* [12] and *Markzware Job2Print* [13], just to mention a few. The majority of these are client-server solutions. The client software ranges from web browser plug-ins downloaded from a web page to standalone clients that are installed on the job deliverer’s computer.

![Proprietary client server solution](image)

**Figure 2-5** Proprietary client server solution

There are just as many different solutions for the server-side portion of the job delivery system. Some solutions are web server plug-ins, others are a small components that are part of large print production systems. There are also solutions where the server software is hosted by a third party (not the job receiver), so called Application Service Providing (ASP).

When it comes to meeting the needs of the graphic arts industries, specialized job delivery clients often have a clear advantage over other file delivery client software, at least regarding usability. Ease-of-use is in focus and there is often support for submitting job metadata together with the job files. Some clients even have preflight functionality built-in.

On the downside, proprietary clients often place restrictions on the job deliverer’s software environment. Web browser plug-ins might only work with certain web browsers, and standalone clients might only run on certain operating systems. The extra effort required if the job deliverer needs to download and install the client software must also be taken into consideration. Security might also be an issue. It seems that some vendors have chosen to focus entirely on usability and skip support for encrypted file transfers.
3 Job Metadata

In an information technology context the prefix meta often means “an underlying definition or description”. Thus, metadata means a definition or description of data.

When delivering a print job, job metadata can be delivered together with the print job. Metadata can be used to describe the files that a print job consists of or be used to describe the entire print job. Each job file can have individual metadata that for example describes how the file was created or provides information about the file’s author. This type of metadata is specific for each content file (for example a TIFF image file) and can be seen as file-centric or content-centric metadata. This type of metadata is often embedded in the file that it describes.

Metadata that describes an entire print job (consisting of several job files) can be seen as job-centric metadata. This type of metadata could provide information about the customer that ordered the print job, specify how many copies of the print job should be printed, or define how the print job should be produced by combining and processing the print job’s files. Job-centric metadata is often contained in a separate file that is distributed together with the job files that it describes.

3.1 Metadata Formats for the Graphic Arts Industry

There are several metadata formats developed specifically for the graphic arts industry. Historically, the majority of these metadata formats have been proprietary. The few formats that have been so called “open formats” have mostly been developed by and specialized for the newspaper industry.

The current trend however is towards open formats. Vendors are joining forces and pooling their resources to develop open metadata formats that can be used to describe and define both the content and the production workflow. The two most interesting of these new open formats are XMP and JDF, both based on XML.

3.2 Extensible Metadata Platform (XMP)

Extensible Metadata Platform (XMP) [14][15] is a metadata format developed by Adobe. As described by Adobe: “[XMP] is a labeling technology that allows you to embed data about a file, known as metadata, into the file itself. With XMP, desktop applications and back-end publishing systems gain a common method for capturing, sharing, and leveraging this valuable metadata — opening the door for more efficient job processing, workflow automation, and rights management, among many other possibilities.” XMP is a method for embedding XML-structured metadata in binary streams (files).

XMP is developed by Adobe but is based on XML (eXtensible Markup Language) [16][17] and RDF (Resource Description Framework) [18][19]. Both XML and RDF are W3C [20] standards, making the format relatively open. Adobe’s XMP software development tools are free and available under an open source license, making it easy for other vendors to implement and leverage XMP in their products.

XMP’s greatest strength is that it is flexible. As its name suggests, XMP is extensible. By using XMP as a container, other metadata formats can be embedded into files in a well-defined way. Another important strength of XMP is that InDesign, Acrobat, Photoshop and Adobe’s other applications have built in support for it [21].
3.2.1 XMP and Print Job Delivery

XMP is an ideal metadata format for delivering file-centric metadata. For example, each job file can contain an XMP packet that describes the file, provides information about how the file was created, and defines how the file should be processed further down the production workflow.

XMP works especially well when the print job being delivered only consists of one job file. The single job file can then contain the metadata describing the entire print job. However, if a print job consists of many files one might be better off having a standalone metadata file, known as a job ticket, that describes the print job.

3.3 Job Definition Format (JDF)

Job Definition Format (JDF) [22][23] is an XML-based job ticket format for describing print jobs in their entirety. In other words, it is a job-centric metadata format. What JDF does is provide a standard way for describing the processes involved in producing a print job. The clever thing about JDF is that the process descriptions are done in such a way that the actual implementations of the processes are abstracted. Only the way processes should communicate and exchange data is described. This way JDF-speaking software and hardware from different vendors can be connected to create an entire production workflow.

Version 1.0 of JDF was released in April 2001. It was originally developed by Adobe, Agfa, MAN Roland and Heidelberg, four major vendors in the graphic arts and printing industry. Today the standard is owned and developed by CIP4, the International Cooperation for the Integration of Processes in Prepress, Press and Postpress [22]. CIP4 is a consortium that currently has more than 160 members ranging from print shops to hardware vendors to newspapers to research institutes. The member organizations can all actively take part in the development of JDF if they wish. At the time of writing JDF has reached version 1.1. There has been some skepticism about how successful JDF will be as a standard. It seems though as if enough JDF enabled products will be released during 2003 for JDF to reach critical mass and wide acceptance. In addition, there are high expectations on drupa 2004 [24] being a “coming out party” for JDF products from all major vendors.

Apart from process automation, JDF also provides a standard communication interface making it possible for systems from different vendors to communicate with each other. This communication protocol is called JMF, Job Messaging Format [25]. Thanks to JDF and JMF all processes in the production system can be linked together. Furthermore, using JMF, the production system can communicate with the management information systems (MIS), the systems responsible for an organizations business data.
3.3.1 JDF and Print Job Delivery

JDF is well suited as a format for submitting job metadata when delivering print jobs. With JDF a job ticket can be created that defines the entire print job as JDF processes. Number of copies, paper quality, binding, the files the print job consists of — all these parameters can be specified in a JDF job ticket. If the job receiver, for example a print shop, has a JDF enabled workflow then the entire print job can be automated. Furthermore, using JMF the print job can automatically be entered into the print shops economy and accounting system.

A weakness JDF 1.1 currently has is that it does not specify a process that models digital job delivery. Such a process will however be added in JDF 1.2, which will be released during 2004. Since JDF is an open and extensible format, vendors have the possibility of adding their own job delivery process as a temporary workaround. By joining CIP4, vendors can contribute to the development of the specification and maybe have their own modifications become an official part of JDF.
4 Definition of a Web-based Job Delivery System

The goal of this thesis is to present a prototype for a user-friendly, Internet-based client-server solution for delivering job files to production workflows in the graphic arts industry.

This chapter presents the requirements placed on the job delivery system. The architectural model is defined; use cases and detailed functional requirements are listed; lastly, technical requirements are given.

4.1 Architectural Model

*Internet-based* and *client-server* suggest that a web-based client-server architecture would be a suitable solution. Using a standard web browser as the client in a client-server architecture is a well-proven solution. The web browser acts as a thin client and is only responsible for presenting the graphical user interface (GUI) and handling user-input. Requests are sent for processing to a server and results are sent back as a response to the web browser client. The entire World Wide Web is built on this architectural model.

![Client-server architecture](image)

**Figure 4-1** Client-server architecture

To summarize, the job delivery system will have a server component and a client component that is web-based.

4.2 Use Cases

Two different user roles can be identified for the job delivery system. Given the client-server architecture defined previously, it would be viable to have a web-based client component for each user role. The two user roles and corresponding clients components would be:

- Print job deliverer — Job delivery client
- Administrator — Administrator client

Each client has a set of use cases. A *use case* defines a piece of behavior of a system without revealing the system’s internal structure. Using use cases a high-level view of the functionality of each of the job delivery system’s clients can be defined.
4.2.1 Job Delivery Client
The job delivery client is used by the print job deliverer to deliver print jobs consisting of job files and job metadata. The job deliverer might be a print buyer delivering print job files to a printer.

![Use case diagram for job delivery client](image)

Figure 4-2 Use case diagram for job delivery client

4.2.2 Administrator Client
The administration client is used by the administrator to configure and manage the job delivery system. The administrator most likely belongs to the organization receiving the print jobs.

![Use case diagram for administration client](image)

Figure 4-3 Use case diagram for administration client
4.3 Functional Requirements

Based on market research performed by Frog Software combined with the use cases above the following functional requirements where placed on the job delivery system:

4.3.1 Job Delivery Over the Internet
The job delivery system should be a client-server solution for delivering print jobs between customer and printer over the Internet. The client software must be cross-platform and be able to run on Windows, Mac OS Classic and Mac OS X.

4.3.2 Encrypted File Transfers
There should be support for secure job delivery over encrypted network connections.

4.3.3 User-friendly
The job delivery system should be simple yet effective. First time job deliverers should find the client software easy to use and returning job deliverers should not have to re-key frequently submitted job information.

4.3.4 Job-Centric
The job delivery system should be job-centric (as opposed to file-centric) and handle the delivery of entire print jobs, not just individual job files. A delivered print job can consist of job metadata, several job files and individual metadata for each job file.

4.3.5 JDF Job Tickets for Job Metadata
All information about a delivered print job — job metadata — should be stored in a job ticket compliant with the Job Definition Format version 1.1. This provides easy integration with other JDF compliant devices from Frog Software or third-party vendors.

4.3.6 Central JDF Job Ticket File Repository
JDF job tickets should be stored in a central file repository, a folder on a server, where other JDF compliant devices can connect to the job delivery system according to the file-based JDF interchange protocol [27]. The location of the JDF repository must be configurable.

4.3.7 Job File Repository with Dynamically Built Folder Structure
A print job’s delivered job files should be stored in a dynamically built folder structure. The folder structure may be built from pre-defined folder names or from values retrieved from the corresponding print job’s JDF job ticket. The folder structure of the job file repository must be configurable by administrator users.
4.3.8 Hot Folder Workflow Support
Hot folder workflows are common in prepress and consist of several software applications linked together using folders in a file system.

![Figure 4-4 Hot folder workflow](image)

There are certain rules that applications taking part in a hot folder workflow must follow. These rules must be honored so that the job file repository can be used as an input folder by a hot folder workflow. Supporting hot folder workflows provides easy integration with non-JDF systems common in prepress, such as older preflight and RIP software.

4.3.9 E-mail Receipt to Job Deliverer
When a print job has been delivered the job deliverer should receive a receipt for the job delivery via e-mail. The receipt must be customizable by the system administrator and be able to contain information from the print job’s job ticket, such as which files were delivered and when.

4.3.10 E-mail Notifications to Job Receiver
The printer that receives the print job should get a notification via e-mail when a new print job has been delivered. As with the deliverer’s e-mail receipt, the e-mail notification must be customizable and be able to contain information from the print job’s job ticket.

4.3.11 Web-based Administration
There must be a secure web-based administration client so that the job delivery system can be configured over an encrypted Internet connection using a standard web browser.

4.3.12 Cross-platform Server Solution
The server component must be cross-platform and run on Windows, Linux and Mac OS X.

4.3.13 Internationalized
The job delivery system should be built from the ground up with internationalization (i18n) in mind. Adding support for additional languages without modifying the application itself should be easy.
4.4 Technical Requirements

Frog Software placed the following technical requirements on the job delivery system:

4.4.1 Java 2 Platform, Enterprise Edition (J2EE)

It was required that the job delivery system be implemented using the Java programming language and built on the Java 2 Platform, Enterprise Edition (J2EE) [28].

J2EE, originally developed by Sun Microsystems [29], is today developed by Sun in corporation with Oracle, Sybase, IBM and other leading vendors in the software industry. Building software with J2EE means that the software will be scalable in both performance and reliability in the sense that the customer can choose which hardware and operating system to run the software on. Frog Software reasons that since J2EE is an open standard developed by many companies in cooperation, neither Frog Software nor its customers risk becoming unhealthily dependent on one single supplier.

4.4.2 Job Definition Format (JDF)

It was required that the latest publicly available release of Job Definition Format (JDF) must be supported.

From a vendor’s point of view, building software on open standards means shorter time-to-market. Less time is needed for research and development and there are often high-quality open source software components available that can be leveraged. From an end-users point of view, buying systems that are based on open standards means a certain influence over future product versions and hopefully a guarantee that systems from different vendors will be compatible.
5 Prototype: Inflow

This chapter presents Inflow, a prototype for a job delivery system based on the requirements from the previous chapter. A detailed high-level presentation of Inflow is given in this chapter. For a low-level description of Inflow’s design and implementation the reader is referred to Appendix A.

Inflow is a web-based client-server system for managing the delivery of print jobs to graphic production workflows. The server portion of Inflow is hosted by the job receiver and runs in a J2EE application server environment (see Appendix C). Inflow’s client side is web-based. The job deliverer uses a standard web browser to connect to Inflow and deliver print jobs. Inflow also features web-based administration and configuration. Figure 5-1 gives a high-level view of Inflow’s architecture and main components.

Out of context one might look upon Inflow as merely a system for transferring files. Seen as part of production workflow however, Inflow, as the name suggests, is a system for handling the inflow of print jobs to a production workflow. This “inflow” of print jobs is the central concept that Inflow is based on.

Figure 5-1 Inflow’s architecture and components

Figure 5-2 Inflows of print jobs to a production workflow
Inflow lets the job receiver configure one or more inflows to her production workflow. Inflows can be individually customized. For example, a job receiver could set up an inflow for a certain type of print jobs, or an inflow could be tailored for a specific customer, see Figure 5-2.

Each inflow consists of the following configurable components:

- A job delivery web form and a web receipt
- A folder structure where delivered job files are stored
- A JDF job ticket template for storing job metadata
- An e-mail receipt sent to the job deliverer
- E-mail notifications that can be sent to the job receiver

To deliver a print job the job deliverer logs on, selects an inflow to deliver a job to, and then submits the job using the inflow’s web form. The job files are stored on the receiver’s server in the inflow’s pre-defined folder structure and a JDF job ticket is created containing the information, job metadata, submitted with the job’s web form. When the job has been delivered the deliverer is presented with a web receipt and gets an additional receipt by e-mail. Finally, the job receiver gets notified via e-mail that a new print job has arrived. Figure 5-3 illustrates the workflow of delivering a print job.

![Job Delivery Workflow Diagram](image-url)
5.1 An Inflow’s Components

What makes Inflow so flexible are the inflows’ customizable components. By customizing an inflow’s components the job receiver can tailor the inflow for a specific production workflow and a specific business case. Figure 5-4 shows the components that an inflow consists of and the relations between the inflow and its components. Some components, such as job delivery web forms and job ticket templates, can be reused between inflows. Other components are unique for each inflow, such as folders structures and e-mail receipts.

Inflows are linked to user groups that job deliverer users are members of. A job deliverer can only deliver print jobs to inflows that her user groups have access to. This way a job receiver can control who gets to deliver print jobs to which inflows.

![Figure 5-4 A diagram of an inflow’s components and their relationships](image)

5.1.1 Job Delivery Web Form and Web Receipt

Inflow uses form-based file upload (see 2.2) for job delivery. This means that the job receiver has a lot of flexibility when it comes to customizing the job deliverer’s user experience. Each inflow can have a unique user-interface and the job delivery form facilitates the submission of job metadata together with the print job files.
5.1.2 Folder Structure
Delivered job files are stored on the job receiver’s server in a central file repository (a folder). Each inflow stores its job files inside the central file repository according to a folder structure that the job receiver has configured. The folder structure can be defined using variables. Variables can be date and time variables or job metadata from the job delivery form, see Figure 5-5.

![Figure 5-5 A defined folder structure and the resulting file hierarchy](image)

For example, print jobs could be sorted into folders depending on the date and time they were delivered. Another example is to define a folder hierarchy that is imported by an asset management system where the folder names are used as keywords for indexing.

Hot folder workflows can be connected to an inflow’s folder structure, as specified by requirement 1.1.1.

5.1.3 JDF Job Ticket Template
INFLOW creates a JDF job ticket for each delivered print job. The job’s JDF job ticket is created from a template that the job receiver defines. When the job deliverer submits a new print job the information from the job delivery web form is inserted into the job ticket template and a job ticket is created. The job ticket also contains references to the delivered job files.

JDF job tickets are stored in a central JDF repository and the job tickets are compliant with Job Definition Format version 1.1, see requirements 4.3.5 and 4.3.6. Seen as a component in a JDF workflow, INFLOW is a combination of what in JDF terms is called an agent and a device [30].

5.1.4 E-mail Receipt and E-mail Notifications
An inflow can be configured to send an e-mail receipt to the job deliverer and e-mail notifications to one or more recipients within the receiving organization when a print job has been delivered.

The e-mail receipt provides the job deliverer with a persistent receipt (as opposed to the temporary web page receipt) that can be referred to. The e-mail notifications make sure that the job receiver always knows when new jobs have arrived.

Information submitted with the job’s web form can be inserted into both e-mail receipts and notifications. For example, the job deliverer’s receipt can contain a list of the delivered files and the job receiver’s e-mail notification can contain the contact information that the job deliverer entered.
5.2 User Interface

Inflow’s user interface is web-based, meaning that a standard web browser is used to interact with Inflow. The user interface is implemented using common web technologies. HTML, JavaScript, Cascading Style Sheets (CSS) and Java Server Pages (JSP) are used.

5.2.1 Job Delivery Client

The job delivery client realizes the use cases in 1.1.1 and consists of a number of views, implemented as web pages, that the job deliverer interacts with in sequential order when delivering a print job. As mentioned earlier, these views are highly customizable. This way the job receiver can tailor the job deliverer’s user experience. Figure 5-6 lists the job delivery client’s different views and their sequence. Screenshots of the views are found in Appendix B.

![Diagram of Job Delivery Client Views](image)

Figure 5-6 Job delivery client’s views

5.2.2 Administrator Client

The web-based administrator client is used to configure inflows, users, groups and other settings that Inflow requires. The administrator client realizes the use cases in 4.2.2. Appendix B shows screenshots of the administrator client’s views.

5.3 Security

Inflow is web-based and relies on and leverages the functionality of the job deliverer’s web browser for delivering print jobs. Authentication and encrypted network traffic is one of the features all modern web browsers have. Inflow leverages this to support secure job delivery over HTTP using SSL (Secure Socket Language), commonly known as HTTPS.

Thanks to HTTPS, Inflow’s encrypted file transfers are as confidential as other web-based services that also use HTTPS, such as online banking.
6 Conclusions

This thesis presented Inflow — a fully functional prototype for a print job delivery system for the graphic arts and printing industry. Inflow is a web-based job delivery system that is hosted on an Internet-connected server by the organization receiving the print jobs. Focus has been on creating a system that is easy-to-use, highly customizable, secure, and easy to integrate with existing and future systems from third-party vendors. Inflow has been implemented using open standards, such as XML and JDF, and with open source technology to as large an extent as possible.

The requirements for ease-of-use, high customizability and security are met by choosing a web-based architecture. The client side is implemented using standard web technologies such as HTML, CSS and JavaScript, while the server-side is based on J2EE, Java Servlets and Java Server Pages (JSP). Using a web browser as a job delivery client provides a highly customizable user interface and built in support for secure file transfers using HTTPS (HTTP over SSL).

![Figure 6-1 Inflow’s Architecture](image)

Process automation and easy integration with other print production systems is facilitated with CIP4’s Job Definition Format (JDF). Inflow also supports hot folder workflows for integration with older preflight software and other hot folder-based software common in prepress workflows.

A central concept of Inflow is, as the name suggests, the inflow of print jobs to production workflows. Inflow lets the print job receiver configure any number of inflows. Inflows can effortlessly be tailored. For example, an inflow could be customized for receiving a certain type of print jobs from a specific customer or simply be used as a replacement for FTP.

Each inflow has an individual web form where the job deliverer fills out a job ticket and attaches job files. When the web form is submitted the job files are stored on the job receiver’s server in the inflow’s custom folder structure, a JDF job ticket is created, an e-mail receipt is sent to the job deliverer and an e-mail notification is sent to the job receiver.

Based on Inflow and the work presented in this thesis, Swedish software company Frog Software has developed a commercial shrink-wrapped product — Frog Inflow [31]. Frog Inflow has received unanimously positive press, has been presented at several trades shows, and is successfully deployed and in production at a number of printing companies in Sweden, among others two of Sweden’s largest printing groups.
7 Acknowledgements

A word of gratitude goes out to everyone who has supported me and contributed to this project. I would like to thank Professor Björn Kruse at ITN\(^1\), Campus Norrköping, Linköping University for his patience and advice. I would like to thank Anders Jönsson and the people at Network Innovation AB for their valuable input and for sharing their knowledge of the graphic arts and printing industry. I am very grateful to Henrik Pettersson for his extreme patience and his much appraised user interface design. Last but not least, I would like to thank Arne Ekstrand and Svante Hedin for making it possible to turn the work in this thesis into a commercial project and for making our visions of Frog Software and Frog Inflow become reality.

This master’s thesis project was conducted in cooperation with the research programme Tryckteknisk Forskning (T2F)\(^2\).

\(^1\) Department of Science and Technology (ITN), Linköping University, [http://www.itn.liu.se](http://www.itn.liu.se)

\(^2\) Tryckteknisk Forskning (T2F), [http://www.t2f.nu](http://www.t2f.nu)
Appendix A

Design and Implementation

This appendix is a detailed description over Inflow’s design and implementation. Knowledge of object-oriented design, design patterns, the J2EE platform, Enterprise JavaBeans (EJB) and Java web applications is required.

A.1 Domain and Persistence Layer

Inflow was designed from the bottom up starting with the domain model: “A domain model represents the nouns in a business problem: the people, things, places and ideas.” [32]

Infused from Inflow’s use cases and functional requirements, the domain model objects were derived. For example, central is the concept of inflows. An inflow needs to be modeled. The functional requirements specify that when a user delivers a print job to an inflow e-mail notifications are to be sent. Inferred is that e-mail notifications need to be modeled. Continuing in this way the domain objects in Figure A-1 were defined.

![Figure A-1 Inflow’s domain model](image)

The domain objects, entities, were then implemented using Enterprise JavaBeans (EJB) [33][34][35] entity beans. J2EE provides two different types of entity beans, bean-managed persistent (BMP) entity beans and container-managed persistent (CMP) entity beans.

With BMP entity beans the developer is responsible for coding all database logic. With CMP entity beans the EJB container takes care of persistence for the developer. Although BMP provides more control of the object-to-database mapping, CMP was chosen for its rapid development and prototyping features.

Figure A-1 shows the class names of the CMP entity beans in italics. These classes are abstract classes and contain the entity bean implementations. The fully qualified class names of these classes are listed in List 1.
Using the Ant [36] build tool in combination with the code generation tool XDoclet/EJBDoclet [37] the EJB local interfaces and local home interfaces [33][34][35] are automatically generated from the implementation classes above. Figure A-2 shows the InflowBean’s generated interfaces. In addition, XDoclet generates the class InflowCMP (shown in Figure A-2) and some other classes that are covered in detail below.

A.1.1 Domain Layer Patterns
The following design patterns were applied to the domain layer:

A.1.1.1 Universally Unique Identifier (UUID) for EJB
For generating primary keys for entity beans the universally unique identifier (UUID) pattern implemented as a stateless session bean was used [32]. The UUID algorithm generates a primary key by combining enough system information to make the key unique over space and time. This pattern provides good performance and does not require database access or synchronized code.

The stateless session bean implementation responsible for generating the UUIDs for Inflow’s entity beans is the class: se.frog.idgenerator.ejb.IDGeneratorBean.
A.1.1.2 Primary Key Class

Instead of simply letting the UUID primary key be represented by a string, each entity has its own primary key class that wraps the UUID string. Using primary key classes instead of strings provides compile time checking for the primary keys.

Using XDoclet a primary key class was automatically generated for each entity bean, see List 2.

List 2 Primary key classes generated by XDoclet

```java
se.frog.account.pk.UserPK
se.frog.account.pk.UserProfilePK
se.frog.account.pk.RolePK
se.frog.config.pk.PropertiesPK
se.frog.inflow.config.pk.EmailReceiptPK
se.frog.inflow.config.pk.FileDestinationPK
se.frog.inflow.config.pk.InflowPK
se.frog.inflow.config.pk.WebFormPK
se.frog.inflow.config.pk.WebReceiptPK
se.frog.inflow.jdf.pk.JDFLitePK
```

A.1.1.3 Domain Data Transfer Object (DTO)

The Data Transfer Object (DTO) pattern is used to minimize network calls when a client accesses or manipulates a domain object on the server. Data transfer objects are created that copy the properties of the domain objects (entity beans). Clients manipulate the DTOs instead of the domain objects. The DTOs perform bulk read/writes that require only one network call per read/write.

As with the primary key classes, XDoclet was used to automatically generate the DTO classes.

List 3 DTO classes generated by XDoclet

```java
se.frog.account.model.UserData
se.frog.account.model.RoleData
se.frog.config.model.PropertiesData
se.frog.inflow.config.model.EmailReceiptData
se.frog.inflow.config.model.FileDestinationData
se.frog.inflow.config.model.InflowData
se.frog.inflow.config.model.WebFormData
se.frog.inflow.config.model.WebReceiptData
se.frog.inflow.jdf.model.JDFLiteData
```
A.2 Services Layer

The services layer of a J2EE application is responsible for delegating calls from clients to the 
domain layer, handling the transactions that the calls to the domain layer run in, and handling 
any workflow between domain objects.

A.2.1 Services Layer Patterns

A.2.1.1 Session Façade

For optimal performance Inflow’s entity beans were wrapped with stateless session beans as 
described in the Session Façade design pattern presented in [38] and [39]. Clients make remote 
calls to the session façade that makes local calls to the entity beans. This design pattern 
minimizes network traffic and provides a strict separation of business logic from presentation 
layer.

The following stateless session bean implementation classes represent Inflow’s session façade 
layer:

List 4 The stateless session beans implementing the session façade

<table>
<thead>
<tr>
<th>Bean Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>se.frog.account.ejb.UserControllerBean</td>
</tr>
<tr>
<td>se.frog.config.ejb.PropertiesControllerBean</td>
</tr>
<tr>
<td>se.frog.inflow.config.ejb.InflowControllerBean</td>
</tr>
<tr>
<td>se.frog.inflow.jdf.ejb.JDFLiteControllerBean</td>
</tr>
</tbody>
</table>

XDoclet was used to generate the remote interfaces, home interfaces, local interfaces and local 
home interfaces for the stateless session beans above. Table 1 shows the naming convention 
used for the interfaces.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Naming convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote interface:</td>
<td>se.frog.inflow.config.interfaces.InflowController</td>
</tr>
<tr>
<td>Home interface:</td>
<td>se.frog.inflow.config.interfaces.InflowControllerHome</td>
</tr>
<tr>
<td>Local interface:</td>
<td>se.frog.inflow.config.interfaces.InflowControllerLocal</td>
</tr>
<tr>
<td>Local home interface:</td>
<td>se.frog.inflow.config.interfaces.InflowControllerLocalHome</td>
</tr>
</tbody>
</table>

Table 1 Naming conventions for session bean interfaces

A.2.1.2 Business Delegate

Business delegates are plain Java classes that provide one-to-one mappings to session façades. 
Clients locally invoke business delegates that delegate the calls to the corresponding session 
façade. Using the business delegate layer pattern clients are shielded from EJB complexity.

List 5 Business delegate classes for Inflow

<table>
<thead>
<tr>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>se.frog.config.PropertiesService</td>
</tr>
<tr>
<td>se.frog.accountUserControllerService3</td>
</tr>
<tr>
<td>se.frog.inflow.config.InflowService</td>
</tr>
<tr>
<td>se.frog.inflow.jdf.JDFLiteService</td>
</tr>
</tbody>
</table>

³ Should be renamed se.frog.account.UserService for consistency with other business delegate 
class names.
Using the inflow entity’s domain model, Figure A-3 illustrates the calls from client to business delegate to session façade, and the classes involved. For clarity the EJB implementation classes are shown in the figure. However, in reality it is the corresponding EJB interfaces generated by XDoclet that are invoked (see Figure A-2 and Table 1).

Figure A-3 Communication between the different layers
A.3 Application Layer

Inflow is implemented as a J2EE web application called Saturn. The Saturn web application consists of a set of Java Servlets, JSP pages, configuration files and other web resources that are bundled into a Web A(Rchive file with a .war extension.

A.3.1 Saturn

The Saturn web application handles the delivery of print jobs and realizes the uses cases defined in section 4.2.1. Saturn is responsible for receiving print jobs, storing job files in predefined folder structures, creating JDF job tickets and sending e-mail notifications. Saturn also has an administration client that realizes the use cases in section 4.2.2.

Saturn consists of the six servlets shown in Figure A-4, the most central one being InFlowUploadServlet. Another central component is Saturn’s JDF template processor used to process job delivery web forms and creating JDF job tickets based on templates. Using these templates job tickets can easily be customized.

![Figure A-4 Saturn's Java servlets](image)

A.3.1.1 se.frog.inflow.saturn.servlet.ControllerServlet

All calls to the Saturn web application go through ControllerServlet. As the name implies, ControllerServlet acts as a controller [40], intercepting all client calls and forwarding them to the appropriate servlet. When intercepting, ControllerServlet performs security checks and preprocesses requests before forwarding them.

Most of Saturn’s ControllerServlet’s functionality is inherited from a more generalized controller implementation found in its abstract parent class se.frog.servlet.ControlServlet.

A.3.1.2 se.frog.inflow.saturn.servlet.StartupServlet

After ControllerServlet, StartupServlet is the first servlet to initialize when the Saturn web application starts up. It loads global resources needed by Saturn’s other servlets and is responsible for initializing the web application the first time it is run. For example, StartupServlet creates Inflow’s default administrator user.
A.3.1.3  se.frog.inflow.servlet.UserAdminServlet
UserAdminServlet is used to create, modify and remove users and groups. It is part of Saturn’s administrator client.

A.3.1.4  se.frog.inflow.servlet.saturn.InFlowAdminServlet
InFlowAdminServlet is part of Saturn’s administrator client and is used to administrate inflows. It creates, modifies and removes inflows and e-mail notifications.

A.3.1.5  se.frog.inflow.servlet.saturn.SettingsAdminServlet
Also part of the administrator client, SettingsAdminServlet handles Saturn’s general settings, such as the location of the JDF job ticket repository and the address of the outgoing mail server used for sending e-mail notifications.

A.3.1.6  se.frog.inflow.servlet.saturn.InFlowUploadServlet
InFlowUploadServlet is Saturn’s core servlet and realizes the use cases in section 4.2.1. Print jobs, consisting of job files and job metadata, are received and processed; JDF job tickets are created; job files are stored in folder structures; e-mail notifications and receipts are sent.

Figure A-5 shows a class diagram of the most important classes related to InFlowUploadServlet. The two classes that provide the bulk of InFlowUploadServlet’s functionality are JDFTemplateProcessor and UploadBean.
JDFTemplateProcessor is the heart of InFlowUploadServlet. When InFlowUploadServlet receives a job delivery, a HTTP File Upload [6] request, a new instance of JDFTemplateProcessor is created to process the request. JDFTemplateProcessor processes job metadata extracted by UploadBean; loads a JDF job ticket template and inserts the job metadata creating the print job’s JDF job ticket; stores the job files in their folder structure; and sends e-mail notifications and receipts. The most complex part is the JDF job ticket template processing.

UploadBean is responsible for parsing the HTTP File Upload request from the job delivery client, most likely a web browser. It parses the submitted web form fields (print job metadata) and extracts and decodes the submitted files (print job files). UploadBean also monitors the HTTP File Upload’s progress and notifies the job delivery client of how much of the job delivery that remains, etc.
Appendix B

User Interface Screenshots

B.1 Job Delivery Client

B.1.1 Login

B.1.2 List of User's Inflows
B.1.3  Job Delivery Form 1

B.1.4  Job Delivery Form 2
B.1.5 Job Delivery Form 3

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field 1</td>
<td></td>
</tr>
<tr>
<td>Field 2</td>
<td></td>
</tr>
<tr>
<td>Field 3</td>
<td></td>
</tr>
<tr>
<td>Field 4</td>
<td></td>
</tr>
<tr>
<td>Field 5</td>
<td></td>
</tr>
</tbody>
</table>

Click **Save & Next**
B.2 Administration Client

B.2.1 List Inflows

B.2.2 Add Inflow
B.2.3 List E-mail Receipts and Notifications

B.2.4 Add E-mail Receipt
B.2.5 List Users

B.2.6 Add User
B.2.7 List Groups

B.2.8 Add Group
B.2.9 Edit Settings

**3D-F-mapp**
- Absolut väg till den mapp där 3D-filer sparas

**Filera**
- Absolut väg till den mapp där lämnade filer sparas. Alla infödda delar samma "Filera".

**Temporär mapp**
- Absolut väg till den mapp där temporära filer sparas. Temporära filer sparades under filera.

**Total filera**
- Den totala filera som använd av tidigare filera.

**Utgående mailserver**
- E-postadress eller domän som får e-post (SMTP), som används för utgående mail.
Appendix C

A Short Introduction to the J2EE Platform

The J2EE platform provides server-side and client-side support for developing distributed, multi-tier applications. Typically, such applications consist of a client tier providing the user interface, a middle tier providing application logic, and a back-end tier providing data management.

Central to the J2EE platform are components, containers and services. Components are reusable software units that are assembled to build an enterprise application. The components are what the programmer actually codes.

A component is designed to run in a container. A container is a standardized runtime environment that provides a specific set of services to components. For example, a J2EE Web container provides services for responding to client requests and returning results to the client. The J2EE platform provides containers for the different layers in the multi-tier architecture.

![Figure C-1 Some of the containers provided by the J2EE platform](image)

Containers are often built into application servers and it is up to the application server vendors to implement the services specified by the J2EE platform. The application server abstracts the J2EE components from the underlying operating system. This way, developers that build their applications from J2EE components can write an application once that runs on any operating system and any hardware.

![Figure C-2 The layers of a J2EE application’s environment](image)
References

[25] Job Messaging Format (JMF), see JDF Specification release 1.1 revision A; chapter 5 JDF Messaging with the Job Messaging Format

[27] See JDF Specification release 1.1 revision A; chapter 8.2.1 File-Based Protocol (JDF only)


[30] See JDF Specification 1.1, 2.1.2 Workflow Component Roles


