The CIP4 JDF Editor – Visualization of JDF

– Design and implementation of a Graphical Viewer for JDF, combined with a Study of the Status of JDF Today

---

CIP4 JDF Editor - visualisering av JDF

– Design och implementering av ett grafiskt verktyg för visualisering av JDF, kombinerat med en undersökning av statusen av JDF idag

Master's Thesis in Publishing Technology at Media Technology and Graphic Arts, Nada, KTH, June 2003

Master´s Thesis by Evelina Thunell, evelina@kth.se, +46 736 26 60 85

Project assigner:
International Cooperation for Integration of Processes in Prepress, Press and Postpress, CIP4

Supervisors:
Björn Hedin, Media Technology and Graphic arts, Nada, KTH
Dr Rainer Prosi, CIP4/Heidelberger Druckmaschinen AG

Examiner:
Professor Nils Enlund, Media Technology and Graphic Arts, Nada, KTH

Abstract

The CIP4 JDF Editor – Visualization of JDF – Design and Implementation of a Graphical Viewer for JDF, combined with a Study of the Status of JDF Today

This master project includes the design and development of a viewer for visualization of JDF files, the CIP4 JDF Editor. JDF is an XML schema based XML format for the printing industry with the main purpose to enable automation and integration of multi vendor systems in a graphical environment. The project is sponsored by CIP4, the organisation responsible for the development and maintenance of JDF. The master project also includes the arrangement of an interoperability matrix, in order to give an opinion of how far the development of JDF has come today. The matrix will also be used for matchmaking by CIP4, to find possible test partners for interoperability test events. Almost every important actor in the graphic arts industry is a member of CIP4 and it is therefore likely that JDF will become the standard format for workflow automation in the future. The specification is very complex and comprehensive though, this is the main problem of JDF. The complexity makes it necessary for CIP4 to publish cookbooks to help vendors start implementations. JDF has not been so widely used in practice so there is not much experience from end-users yet.

Sammanfattning

CIP4 JDF Editor – visualisering av JDF – Design och implementering av ett grafiskt verktyg för visualisering av JDF, kombinerat med en undersökning av statusen av JDF idag

Det här examensarbetet har gått ut på att designa och utveckla den del av CIP4 JDF Editor som visualiserar JDF filer. JDF är ett XML-schemabaserat XML-format för tryckindustrin med huvudsyfte att möjliggöra automatisering och integration av system från olika leverantörer. Projektet har sponsrats av CIP4, vilken är den organisation som är ansvarig för utvecklingen och underhållet av JDF. Examensarbetet består också av att sammanställa en interoperabilitets-matris. Detta för att sedan kunna uttala sig om hur långt utvecklingen av JDF har kommit. Matrisen ska också användas för att hitta potentiella testpartners för interoperabilitets-evenemang. Nästan alla viktiga aktörer i den grafiska industrin är medlemmar i CIP4 vilket gör att det är troligt att JDF blir standardformat för arbetsflödesautomatisering i framtiden. Specifikationen är dock väldigt komplex och omfattande, vilket är det största problemet idag. Komplexiteten gör det nödvändigt för CIP4 att publicera kokböcker, för att hjälpa leverantörerna att få igång implementeringen. JDF har inte testats i så stor omfattning i verkligheten så det finns ännu inte många erfarenheter från slutanvändare.

Preface

This report is a master project at Media Technology and Graphic Arts, Nada, KTH. The assigner of the project is CIP4 and it has been performed in close collaboration with Anna Andersson. For a full understanding of the project I recommend the reader of this paper to read Anna Andersson's report "The CIP4 JDF Editor, the editing part" as well. The supervisor at KTH is Björn Hedin and the supervisor at CIP4 is Dr Rainer Prosi.

I would like to thank Anna Andersson for good cooperation, Dr Rainer Prosi for guidance, support and taking the time to supervise us, Kai Mattern, Dagmar Sclenz and Dietrich Mucha for advices and help during the project, Koen Van De Poel and Tom Hastings for feedback and advices, Tim Donahue, Frank Theede and Eckhard Bölke for information and Björn Hedin for support and feedback.

List of contents

1 Introduction

This chapter shortly describes the technologies Job Definition Format (JDF) and its complement part, Job Messaging Format (JMF). It also includes a description on how the development and maintenance of these technologies are handled. Chapter 3.1 will discuss JDF and JMF further.

1.1 Background on CIP4

Four prominent companies in the graphic arts industry, Adobe, Agfa, Heidelberg and MAN Roland, originally took the initiative to JDF. In 1999 they handed over the development of JDF to the International Cooperation for Integration in Prepress, Press and Postpress (CIP3), an association that supports the development of digital standards for the graphic arts industry. In July 2000 CIP3 became International Cooperation for Integration of Processes in Prepress, Press and Postpress (CIP4), see Figure 1.1. The new word in the name, Processes, represents the ability to connect the production systems with the Management Information Systems (MIS).

Figure 1.1 The CIP4 logotype

CIP4 has currently 156 vendors, other industry associations, end-users and research institutes as members. There are three different membership levels: partner, full member and associate member, which offer different influence over the development process and different membership fees. The members themselves decide how much work they want to do for CIP4. The development of JDF and JMF is carried out in working groups responsible for different areas. There are currently 18 working groups:

- Advertising / Magazine publishing
- Capabilities
- Color Workflow
- Conventional printing
- Device messaging / Job tracking
- Digital Printing
- eCommerce
- Finishing
- Gravure
- Newspaper
- Origination and Prepress
- Packaging & Label
- Process resources and definitions
- System Behavior and Interoperability
- Tools and infrastructure
- Framework
- User Group
- Web/Rotary printing

1.2 Background on JDF and JMF

The primary reason why the initiative to JDF was taken is the increasingly more complex workflows the graphic arts industry is facing today. Steering systems, economic systems and management systems from different vendors are used to handle these workflows. In order to do so efficiently, and to avoid problems, they need a common language for information exchange. JDF is that language; it is one step towards a more automated workflow, which would make the printing processes more productive, flexible and transparent through the increased control of the production. The JDF specification intends to cover the whole production chain, from order, via the creative phase, prepress, press and postpress to delivery. In the graphic arts industry, there is no standard workflow and therefore automation is quite complex. JDF and JMF make it possible for equipment from different vendors to work together and will therefore make the production more flexible. JDF provides primarily the following benefits to the printing industry:

- Ability to carry a print job from genesis through completion. This includes a detailed description of the creative, prepress, press, postpress and delivery processes.
- Ability to bridge the communication gap between production and Management Information Services. This ability enables instantaneous job and device tracking as well as detailed pre- and post-calculation of jobs in the graphic arts.
- Ability to bridge the gap between the customer's view of product and the manufacturing process by defining a process independent product view as well as a process dependent production view of a print job.
- Ability to define and track any user defined workflow without constraints on the supported workflow models. This includes serial, parallel, overlapping and iterative processing in arbitrary combinations and over distributed locations.
- Ability to do the above mentioned under nearly any precondition. [1]

JDF is an XML-based format and proposed industry standard based on the technologies of CIP3´s Print Production Format (PPF) and Adobe's Portable Job Ticket Format (PJTF). eXtensible Markup Language (XML), is a descriptive computer language, which is used in JDF and JMF to describe information. The JDF specification was presented at the Seybold fair in February 2000. Version 1.0 was released in April 2001 and version 1.1 in April 2002. CIP4 is working on the next version of the specification, which is planned to be released in the third quarter of 2003.

JMF, is the messaging format of JDF and functions as an interface between the production equipment and the Management Information Systems (MIS) to establish a dynamic interaction in a JDF workflow system. JMF provides a job tracking functionality and could, for example, be used to establish a queue, find out the capabilities of a JDF-enabled component or determine its status. [1]

1.3 Problem

This master project includes two primary parts. The first part is the development of a tool for visualizing JDF files, as a part of the CIP4 JDF Editor. The second part is a study of the JDF development today and summary of JDF supporting products. This is done with focus on the development, status and problems of JDF in newspaper production, digital printing and MIS.

There are no products, such as the CIP4 JDF Editor, available on the market today. Viewing and editing of JDF files are often done in Microsoft Internet Explorer or in regular XML-editors, such as XMLSpy [2]. But these editors are not adjusted to the information JDF incorporates. JDF files describe processes and relations between resources and processes - functions that are not available in any XML editor. Anna Andersson's part, editing and validation after editing, will be other features of the application. This master project deal with the development of an application that handles the above described features:

- What user aspects should be considered for the viewer in the CIP4 JDF Editor?
- What is the difference compared to ordinary XML editors?
- Which information in a JDF file shall be visualized?
- How is a JDF file best visualized?
- The developments of the graphical user interface, what design and features shall the CIP4 JDF Editor have?
- A description of the viewer.
- Result from the user test.

Since the JDF specification is rather new it is interesting to find out how far the development of the standard has reached, how widely adopted it has been and how many products that supports it. The study of the status of the JDF development today will deal with and discuss the following issues:

- The state of the industry before JDF, why is JDF needed?
- What will JDF do for the vendor and the print shop, respectively?
- Status of the JDF and JMF development?
- Study of the JDF development in the areas digital printing, newspaper production and MIS.
- Time perspective.
- Problems.
- Presentation of the interoperability matrix – how far has the development come?

The focus will be on describing the development, status and problems newspaper production, digital printing and MIS are dealing with in the JDF context. Newspaper production is interesting because of the widely spread opinion that JDF is more focused on conventional printing. In digital printing I will try to describe how JDF is used to handle personalized printing and variable data. MIS and JMF are close to each other, could the reason why the development of JMF not has reached so far be due to the lack of MIS vendors involved?

1.4 Delimitations

Due to the time limit there are some features of JDF the CIP4 JDF Editor does not support; schema validation, refElements and partition display. There are also some general features that are left out: drag and drop, undo, redo, a source view, printing and the ability to hide and show the attributes in the tree view.

We have delimited the investigation of JDF-supported products to the vendors who are CIP4 members. It is important to mention that the interoperability matrix does not give a complete overview because it is likely that a lot of products are under development and the companies do not want to contribute information about them yet.

1.5 Purpose

The purpose of developing the CIP4 JDF Editor is to enable visualization of JDF and JMF files in a suitable way, and to be able to follow processes and display the errors. The user interface shall be simple and user friendly. The JMF file is something that will never be seen by the user, and the possibility to create and modify JMF files has been implemented for an educational purpose. The CIP4 JDF Editor will also be a tool for validation and editing of JDF. There is no such application available on the market today that provides the features described above.

The purpose of the study of the JDF development today is to find out how far the development of the standard has reached, how many companies that actually have adopted it, how many products that support it and on which production areas the focus is. Another function of the interoperability matrix will be matchmaking, to find possible test partners for interoperability testing events.

I will also try to give the reader an understanding for the JDF and JMF technologies and give the background on why they were developed.

2 Method

Since JDF is a rather new specification, there is not much literature about JDF and JMF today, and most of the information has been found in the JDF specification and through interviews with CIP4 members.

The CIP4 JDF Editor is based on the CIP4 Java JDF Library, an open source Java library that contains classes for creation, modification and validation of JDF and JMF files. The application was to be developed in Java to realize the requirements of platform independence and in order to be able to make use of the CIP4 Java JDF Library. Java Version 1.4.1 has been used. The library was diligently used but a thorough study had to be done in order to find and make use of the right methods. The development was done in Eclipse 2.0 [3], a development environment used by the Java developers at Heidelberg. ClearCase [4] was used for version handling and merging since we were two developers on the project. JProfiler [5] was used to find performance problems of the editor.

The first step, to decide the requirements and guidelines of the application, was done through discussions with the developers of the CIP4 Java JDF Library, GUI experts and the members of the CIP4 working group Tools and Infrastructure. This group is responsible for defining, implementing and supporting open source tools. An evaluation of various XML editors was also done to further decide the look and the features of the application. The CIP4 requirements and the result of the evaluation gave the guidelines for the interface of the application. As window manager Swing, AWT and SWT were considered. Swing was chosen because of previous experience. The validation in the CIP4 JDF Editor is done programmatically and is based on an existing CIP4 open source application, CheckJDF.

The project was performed in close cooperation with Anna Andersson, although we were responsible for different parts. We had discussions on decision-making and problem solving during the entire project. The project was monthly presented during the teleconferences of the CIP4 Tools and Infrastructure working group for feedback and checking. Problems that arisen were solved in collaboration with experts within CIP4 and Heidelberg. The members of CIP4 had the possibility to test the application and come with error reports and input.

In order to find out how user friendly the final user interface of the application turned out, the CIP4 JDF Editor was compared, in form of a user test, to three of the XML editors that were evaluated in the beginning of the project. The test did not cover any JDF specific features; only the features that could be compared to the other editors were tested.

To find out the status of the JDF development today an interoperability matrix was put together. In order to get data to the matrix all vendors within CIP4 was contacted and asked to fill in a form that was arranged in collaboration with Anna Andersson and the System behavior and interoperability working group. The data supplied concerned their products and what kind of information those could exchange, expressed in processes produced and consumed, and what JDF versions and JMF level they supported. Another important source of information has been interviews with representatives from CIP4 members. The interoperability matrix has been done in collaboration with Anna Andersson.

3 Result

3.1 The JDF Concept

This chapter will shortly summarize the basic features of JDF. It also includes a part of what happens to the JDF job during execution.

3.1.1 XML and JDF

JDF is used to describe how a printing job is to be executed; it includes job parameters and instructions. The JDF file is a job ticket. The JDF schema establishes the rules for how the information in the file shall be described. A job is the sum of a JDF project and contains all information needed to complete the job. The information is stored in XML elements, called nodes. A job often contains more than one node and they are arranged in a tree structure. The node at the top, the JDF root, describes the overall intention of the job, the intermediate nodes describe the increasingly process-oriented aspects of the job, and the nodes at the bottom describe a single process. An XML element is an XML syntactic construct and refers in the JDF specification to the subparts of JDF nodes. An attribute is also an XML syntactic construct, describing the characteristics of an element. An example of a JDF file is available in Appendix A.

3.1.2 JDF Nodes

There are three kinds of JDF nodes, Product, Process and ProcessGroup nodes, and they describe accordingly a process, a product or a combination of the previous two. The Product node includes information about the intent of the job, the ProcessGroup node describes groups of processes and the Process node defines individual workflow steps. Examples of processes are ColorCorrection, Imposition and ConventionalPrinting. The Product node on top in Figure 3.1 can for example describe the entire book, while the other two Product nodes describe the content and cover, respectively. The ProcessGroup nodes describe for example the production of the cover pages and the content pages. The Process nodes could describe printing of cover, printing of content, finishing and binding of the entire book.

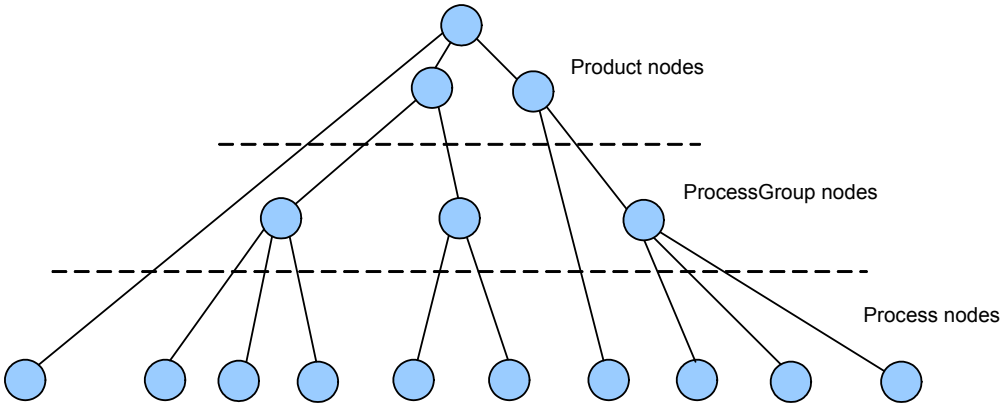

Figure 3.1 Hierarchical tree structure [1]

The structure of JDF nodes can be described in two different ways, hierarchical and lateral. The hierarchical structure is a nested structure with parent nodes containing child nodes, the node at the top describes the whole job and its children nodes at the bottom describe a single process, see Figure 3.2.

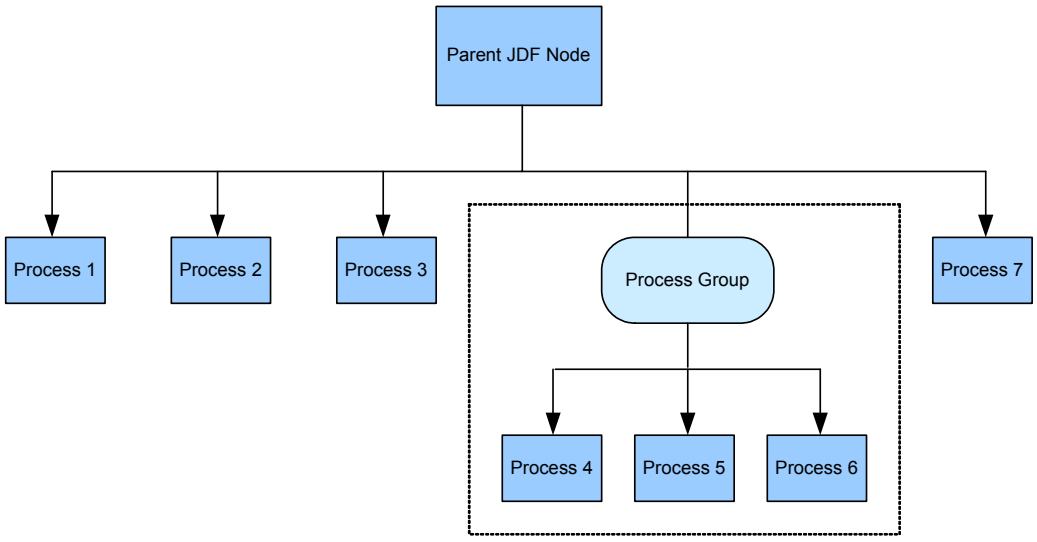

Figure 3.2 Hierarchical tree structure of JDF nodes

The lateral structure shows the relation between nodes when following resource links. The output resource of one node is often the input resource of another node. Nodes often share resources. Figure 3.3 is an example of lateral workflow.

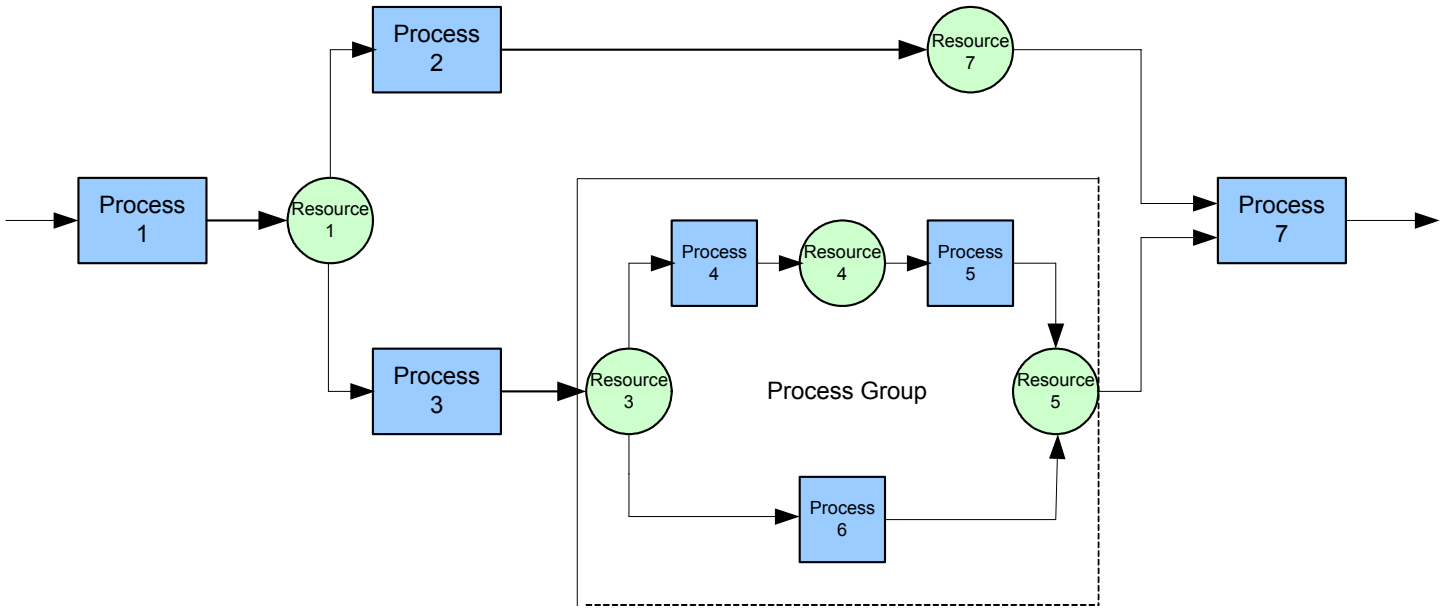

Figure 3.3 Lateral structure of JDF nodes accomplished by resource linking

3.1.3 Resources and Resource Links

Resources are the things to be produced, modified, consumed or used by processes. Examples of resources are inks, plates, files or device settings. A process is defined by its input and output resources. The resources a process uses for input and output are

defined in the element ResourceLinks within that process. The input and output resources of processes make it possible to determine process dependencies. The resources are stored in the ResourcePool element of a node. A node can only reference resources within its own ResourcePool or in ResourcePools within nodes closer to the JDF root. A process cannot be executed until all of its input resources are available.

ResourceLinks describe what resources a node uses and how it uses them. All logical and chronological dependencies within a node are specified using ResourceLinks. The ResourceLinks are stored in the ResourceLinkPool of a node, the ResourceLink define if the resource is an input or an output resource. The ResourceLink element can also contain optional attributes in order to select a part of a resource, this is called resource partitioning.

3.1.4 Intent

A job always begins with a goal of how the product will look like when the job is completed. In JDF these goals are described by using product intents, represented by intent resources. In contrast to process resources that describe specific values, intent resources define the details of the product to be produced without defining the process to produce it.

The agent initiating a job defines product intents. The product intent is defined in Product nodes and these are required to contain a resource that represents the physical result specified by the node.

3.1.5 Components

The JDF specification defines four components to create, modify, route, interpret and execute a JDF job:

- *Agents*, an agent is writing JDF and has the ability to create a job and to add and modify existing nodes. An agent can for example be a software process or an automated tool.
- *Controllers*, a controller route the JDF job to the correct device. A controller has to be able to initiate processes on at least one device and be able to work together with other controllers and must therefore support the JDF file-exchange protocol. A controller has to be able to determine which nodes that are executable and route them to the right device.
- *Devices*, a device must be able to execute the information specified by the agent and routed by the controller and initiate machines that can perform the execution.
- *Machines*, a machine will execute the process.

3.1.6 MIS

The Management Information System (MIS) oversees the relations between all parts in a workflow; it controls scheduling, execution and control of work in progress. To accomplish this feedback JMF messaging or audit records within JDF can be used. Data feed back during production is continually prepared for checking by the MIS. These checks include parameters concerning deadlines, output, quantities, costs and quality assessments. The quality and completeness of the data are decisive for the integration capability of the network planning and control. [6]

3.1.7 JMF

JMF provides means for real time communication with MIS and for interactive control of elements in a workflow system. This is needed to make the workflow run efficiently.

The structure of a JMF message is seen in Figure 3.4. A JMF message contains one or more of the following five high level elements, called message families:

- A *Query* is a message that retrieves information from a controller without changing the state of that controller. A Query is sent and a Response is returned.
- A *Command* not only retrieves information, it also causes a state change in the device it was sent to.
- A *Response* is returned after a Query or a Command is sent from a controller.
- A *Signal* message is a unidirectional message sent to other controllers, it used to automatically broadcast status changes.
- An *Acknowledge* is an asynchronous answer to a Command. It is unidirectional and is generated if Commands with long latency have been executed in order to inform the Command sender the results.

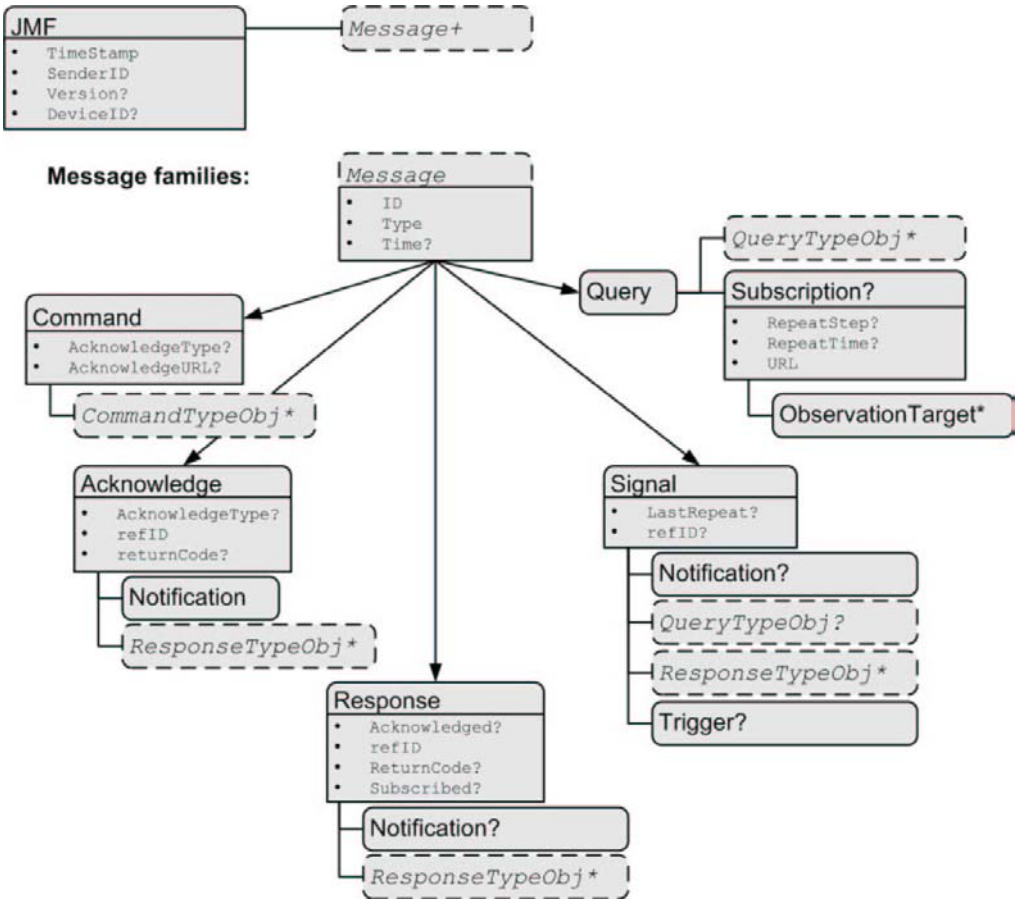

Figure 3.4 The JMF Message families [1]

There are five JMF messaging levels that a controller can support.

- *No messaging*, the controller does not support messaging at all. To record data, JDF includes Audit records for the process.
- *Notification*, this is the most basic level of support. A notification message from a device informs the controller when it begins executing or completes a process. They can also provide notice of some error conditions

- *Query support*, controllers that support queries respond to requests from other controllers by communicating their status.
- *Command support*, if a controller supports this level of messaging it has the ability to process commands. The controller can receive commands to interrupt a job, restart a job or to change status of a job in a queue.
- *Submission support*, this is the highest level of messaging. A controller that has submission support can accept a JDF job via an http post request to the messaging channel. [1]

The sending controller creates a JMF message. A JMF file with a message of the message family Response is available in Appendix B.

3.1.8 JDF Extensibility

One of the main benefits of JDF is the flexibility it supplies through its extensibility. Extensibility means that a vendor can add their own elements etc. to their application in order for it to include all data needed. These so-called private extensions are implemented by using XML namespaces. Private extensions that have become widely used are in the future intended to be included in the next version of the specification.

3.1.9 The JDF Workflow

A JDF job is normally created by one agent and is then modified by other agents during execution. A job is modified during execution, for example will the status of a process or the availability of a resource change.

A controller can route the JDF job first when all input resources and parameters of a node in that very job, are available. The controller routes it to a device, which will execute the process. After the process is completed, the agent or controller will modify the node to record the results of the process, as shown in Figure 3.5 and 3.6.

```
<JDF>
…
     <AuditPool>
          <Created Author="Agent" TimeStamp="2003-05-07T15: 32:05+01:00"/>
     </AuditPool>
</JDF>
```

Figure 3.5 The file before modification by the agent [1]

```
<JDF>
…
   <AuditPool>
      <Created Author="Agent" TimeStamp="2003-05-07T15: 32:05+01:00"/>
      <Modified Author="SecondAgent: task=*" TimeStamp="2003-05-07T17: 46:54+01:00"/>
      <PhaseTime End="2003-05-07T16: 58:50+01:00" Start="2003-05-07T16: 56:20+01:00" Status="Setup"
          TimeStamp="2003- 05-07T16: 58:50+01:00"/>
      <PhaseTime End="2003-05-07T17: 35:23+01:00" Start="2003-05-07T16: 58:50+01:00" Status="InProgress"
          TimeStamp="2003-05-07T17: 35:23+01:00"/>
     <PhaseTime End="2003-05-07T17: 46:54+01:00" Start="2003-05-07T17: 36:40+01:00" Status="Cleanup"
          TimeStamp="2003-05-07T17: 46:54+01:00"/>
     <ProcessRun End="2003-05-07T17: 46:54+01:00" Start="2003-05-07T16: 56:20+01:00" End-Status="Completed"
          TimeStamp="2003-05-07T17: 46:54+01:00"/>
   </AuditPool>
</JDF>
```

Figure 3.6 Code after modification by the agent [1]

JDF supports four kinds of process sequences:

- *Serial processing*, where the nodes are executed sequentially with no overlap.
- *Overlapping processing* means that a process can start to execute before another process producing one of its input resources has completely finished producing them. This is accomplished by using a method called pipes.
- *Parallel processing* makes it possible to run more than one process at once. A part of the job is spawned off and executed individually. When the process has completed, the spawned node is merged back into the job.
- *Iterative processing* is more suitable for some processes where serial or parallel processing cannot be used. Iterative processing means that the production is carried out iteratively in cycles, for example when a page is not approved and needs to be edited and approved again.

To detect errors in a JDF job, it could go through a test run before it is executed. The test run could be routed through all the devices and controllers just like it is during execution. The devices and controllers that receive the job during a test run read and analyse it without initiating execution.

All nodes and resources have a unique identifier, an ID. This makes it possible to keep track of individual components of a job. The ID is also used for resource linking. The creating agent is responsible for providing an ID that is unique within the file as long as it is used.

The device capabilities description makes it possible for a device to report what parts of JDF it can and cannot handle. An individual JDF job description can be compared to the device capabilities of the device. The job can be executed if all job parameters are within the ranges specified by the device capabilities. [1]

Figure 3.7 shows what the interaction between MIS and production can look like.

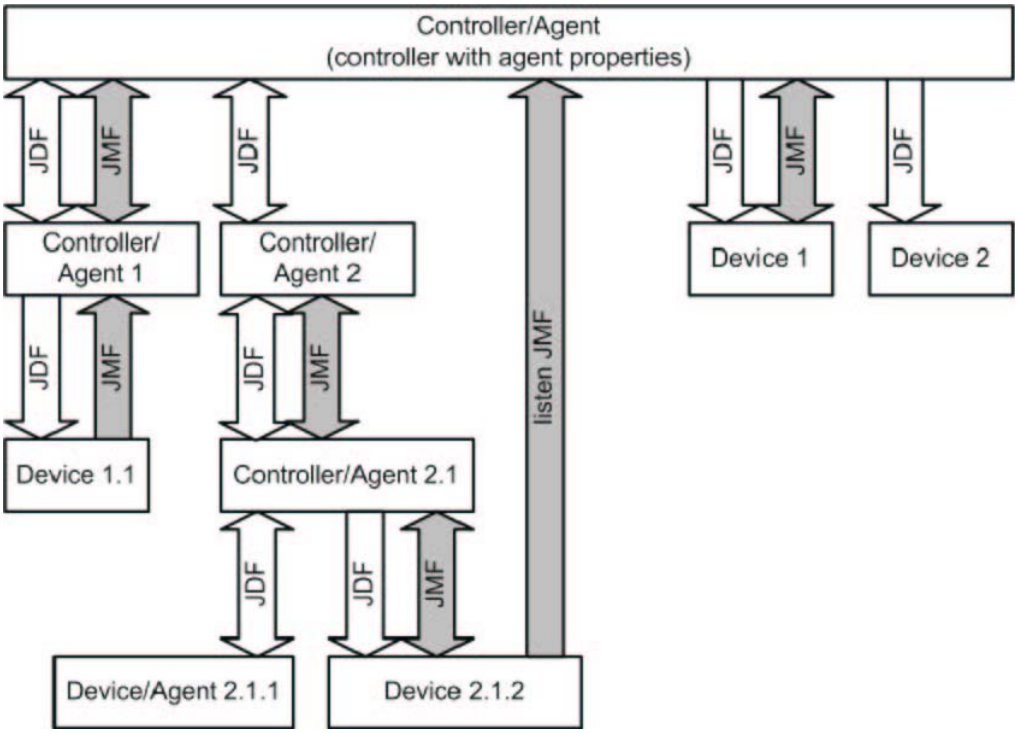

Figure 3.7 JDF and JMF workflow interaction [1]

3.2 The CIP4 JDF Editor

The CIP4 JDF Editor has been developed in close cooperation with Anna Andersson. This thesis describes the development of the viewer part, and the design of the user interface whereas Anna's thesis includes the development of the editing features of the application. This chapter shortly describes the application and the implementation.

3.2.1 Evaluation of XML Editors - What User Aspects should be Considered for the CIP4 JDF Editor?

The first draft of the application was made after an evaluation of six XML editors, three of them commercial and three of them open source or freeware. There are many XML editors available and these six were chosen after tips from competent users, they were all considered good. The three commercial editors were also chosen since it was possible to get a free testing period. The open source editors could be of value if some feature that would be suitable in the CIP4 JDF editor was found in the code. The six editors were:

- XMLSpy [2] (commercial)
- Morphon [7] (commercial)
- XMLPro [8] (commercial)
- Pollo [9] (open source)
- XML Operator [10] (open source)
- XML Notepad [11] (freeware)

They were evaluated after the following aspects; view, editing concept, schema validation, namespace handling, source view, handle of invalid / not well formed XML, help, programming language, window manager and finally a general overview.

The commercial editors were in general better; they had more functions and were more user friendly. All of them, except XMLSpy [2], had some kind of tree view. XMLSpy [2] was the most complex and the best according to the writer of this thesis. A summary of the evaluation is available in Appendix C.

The evaluation resulted in the following guidelines of the design and features of the CIP4 JDF Editor:

- Tree view with editing functions
- Source view
- View for visualizing the process relations
- Error view
- Search function
- Cut, copy and paste
- Undo and redo
- Drag and drop
- Insert elements and attributes
- Schema validation when opened, saved, closed and edited. Possibility to switch it on and off
- Namespace handling
- Ability to open an invalid file with error messages, mark the errors
- Help – tool tip help and a long help
- Swing as window manager
- Pluggable look and feel
- Internationalization
- Mac and PC compatible
- Easy and simple interface

These general guidelines were to be modified on the way due to the time limit. There was no time to implement source view, schema validation, drag and drop, undo and redo and the namespace handling is limited to the possibility to highlight foreign and private elements and attributes.

3.2.2 Which Information in the JDF file shall be Visualized and How is it Done in the Best Way?

There are two general ways to visualize the information in a JDF file. The first way is the hierarchical tree structure, which corresponds to the hierarchical workflow model. The other way is to see the processes that it describes through resource linking as in the lateral workflow model. The two ways complement each other, and should therefore both be present in the CIP4 JDF Editor. The hierarchical structure of a JDF file can be viewed in any XML editor or in Internet Explorer. It was decided that it should be implemented as a tree, a common way to describe hierarchical structures and is therefore well known to the user. It is the lateral workflow model that will make the application unique and it was decided that it would be best done in two complementary ways. In the first view, the input and output view, only the relationships of one process or resource at a time are displayed in detail. In the other view, the process view, one process with all the processes it includes is displayed.

3.2.3 What Differs Between a JDF Editor and an XML Editor?

As mentioned before there are some features of JDF that an ordinary XML editor cannot handle. JDF can be seen as an ordinary XML file but also as a description of how the processes links to each other. Both these views should be available in the application. The tree view represents XML and the process view displays how the resources and processes described in the file are linked. Also, the input and output view displays these relations in another way. The errors messages are adjusted to JDF since the validation is made programmatically.

3.2.4 The CIP4 JDF Editor in General

To easily run the application, an executable application was created. When the application is initially launched no file is opened and only relevant menu items and buttons are displayed. A file can be opened through a file dialog or through the open recent files menu, where the four most recently opened files are shown. When choosing to open a file, an instance of JDFDocument, a class in the CIP4 JDF Library, is created and the file is parsed through methods in that class. The program checks whether it is a JDF or a JMF file. The file is then sent to a method that creates and displays the tree. The tree is displayed collapsed from start in order to make the opening process less time consuming. The file name is shown in the border of the application. Only one file can be open at a time. If the user chooses to open a completely new file, she/he gets to choose from opening a JDF or JMF file, which gets created through methods in the CIP4 JDF Library.

The automatic validation is switched off by default. The user can switch on and off the automatic validation at any time. If the automatic validation is switched off the validate button is enabled and the user can validate the whole file at any time. When switching it on the whole file is validated. Before opening a file the user can chose to open a file as “read only” by switching off the editing mode.

If an existing file has being modified or a new file has been created and the close or exit button has been pushed the user always gets the question if she/he wants to save the file. The saving of the file is also done through methods in the CIP4 JDF Library.

3.2.5 The Graphical User Interface

It was decided that the CIP4 JDF Editor, like most editor applications, should have a menu bar and a button bar at the top. The menus needed were: File, Edit, Insert, Tools, Validate and Help. Only currently valid menu items and buttons are enabled in order to help the user to make the right choice. The menu bar and the button bar is available in Figure 3.8.

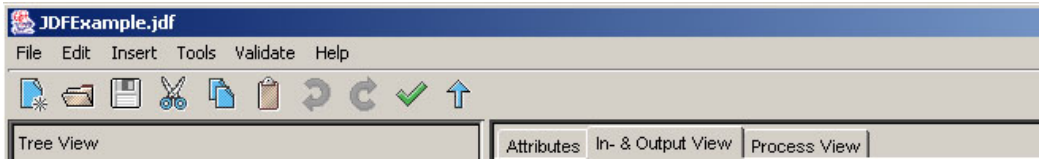

Figure 3.8 The menu and button bars

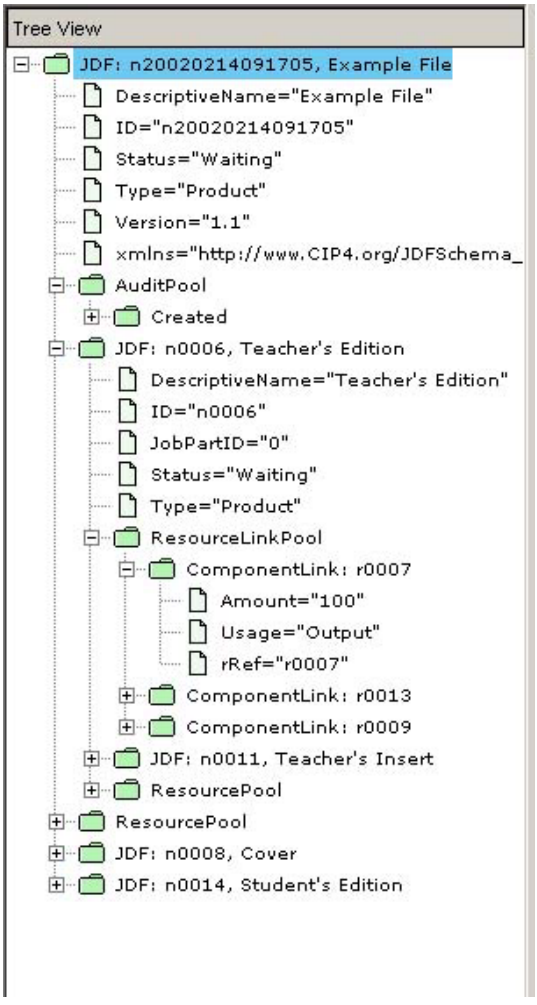

Figure 3.9 The Tree View

Some menu options, such as Open, New, Save, Copy, Cut and Paste are available from the button bar as well. Some button icons were downloaded from the Java Look and Feel Graphics Repository. Every icon needed was not available and were created in Photoshop following the guidelines in the Repository. [12]

In order to have all the views in a well-arranged way, tabbed panes were used. The attribute, input and output and process view all share the same space. To display the desired view, the user only has to select the corresponding tab at the top.

Tree view
The tree view is the main view and it displays the hierarchical structure of the XML file. It also displays the attributes of an element as sub-nodes of the tree node, see Figure 3.9. This will make the attributes easy accessible for modification. To distinct the elements and attributes from each other they are represented with different icons. To achieve this the tree renderer had to be overridden. The tree view has connections to all of the other views in the application in order to make it easy to follow links and still keep track of where in the file the user is currently positioned. Editing is only available from this view and could be done by a right mouse click menu or from the Edit menu.

Error view
When the file is validated the error messages are exposed in the error view, see Figure 3.10. Every error message is connected to the tree node it belongs to in order to make it possible to find where the error occurred and more easily make corrections. The error view is a list that contains list elements that store both the error message, which is a string, and a tree node. The list element is stored in the tree node itself. When a list element is selected a search algorithm using depth first enumeration searches the tree for that same tree node, when it is found it gets selected. When selecting a tree node in the tree view the error messages for that particular JDF element and all its children are shown, the search method searches the tree for invalid nodes and displays the list elements they contain. This means that clicking on the top-level node will reveal all the errors of that file.

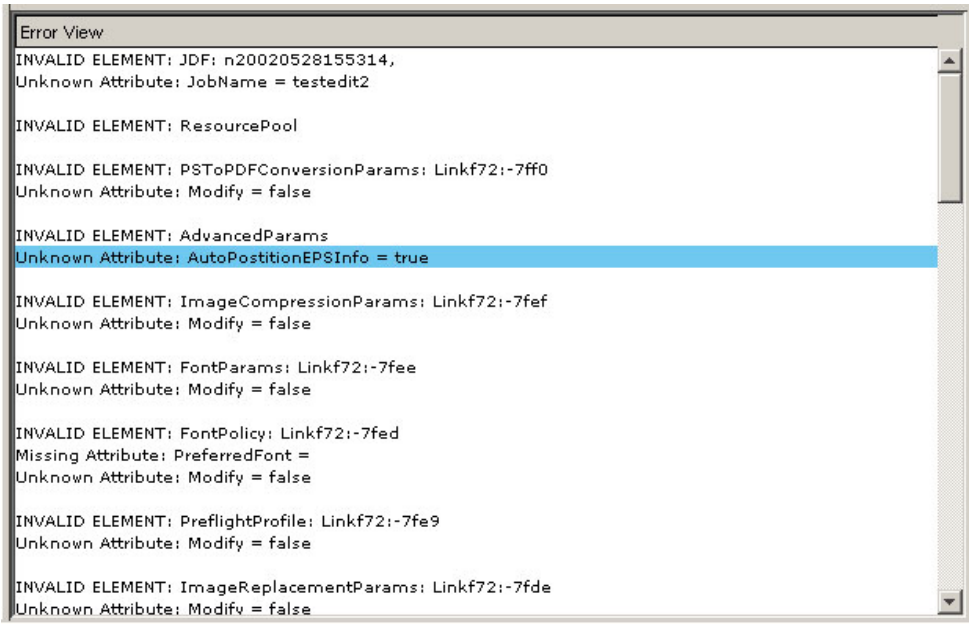

Figure 3.10 The Error View

Attribute view
In the attribute view the attributes of an element are listed. This is the only function of this view. The attribute view remains from a stage in the development where the attributes were to be optionally displayed in the tree view. They are an important part of the file and needs to be easily accessible. Initially the intention was to make it possible to hide the attributes in the tree view to get a better overview of the file and then view them in the attribute view instead. Since the attributes also are Java-tree nodes as well as the elements, it is a quite complex task to hide only some of the children of a tree node, it is non-standard behavior. The decision was taken to concentrate on other parts of the application instead of implementing the optional hiding of attributes in the tree view.

Input & output view
In the input and output view, either a certain resource with its JDF consumer and producer or a JDF node with the resources it inputs and outputs, can be displayed. Clicking on a resource in the tree view will therefore show that particular resource in the center of the view with the JDF node that produces it on the left side and the JDF node that consumes it on the right side. When clicking on a resource link in the tree view, the link is followed to the resource it links to, and that same resource is centered in the view. When clicking on a JDF node, that JDF node will appear in the center, with the input resource(s) on the left and the output resource/resources on the right.

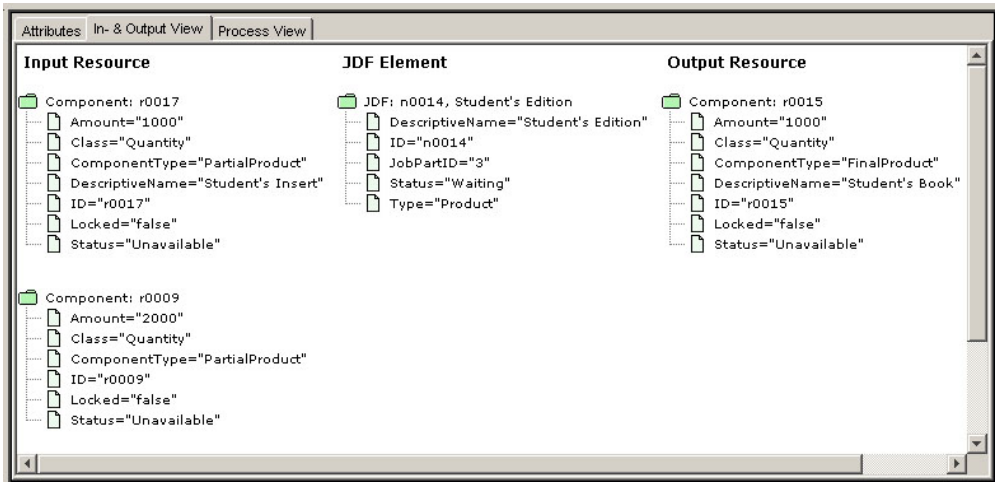

Figure 3.11 The Input and Output View

The resources and JDF nodes displayed in this view are displayed as small trees, with their attributes as children, see Figure 3.11. If an element inherits attributes from its parent the text is bold in order to separate them from each other. A left mouse click (which correspond to CTRL and mouse click on Mac) on an element or an attribute in the input and output view results in that the corresponding element or attribute gets selected in the tree view. Doing a right mouse click on one of the elements displayed on the sides will result in that this element is displayed in the center and the elements it relates to are displayed on the sides. This feature will help the user to follow links and see how the processes relate to each other.

In the CIP4 JDF Library there are methods for finding the target of a resource link. So it is not difficult to find the resources a JDF node consumes and produces, it is just to follow the links stored in the ResourceLinkPool. To find the JDF nodes that consumes or produces a resource, all the ResourceLinkPools in the file has to be searched to see if they contain a link to that particular resource. The most difficult task with the input and output view was to position the components within the panel that stores them. To solve this the layout manager was removed and the components were instead positioned by stating the exact point in the view where they should be located.

Process view
Another way to follow the processes is in the process view, which displays the process or product a JDF node describes and the processes it contains. The resources are displayed as green rectangles and the JDF nodes as blue rectangles with round corners, see Figure 3.12.

Clicking on a JDF node will always activate the input and output view first. To see the process view the user has to select that tab. Well inside the user can navigate the file. By clicking on the “Go up one level in process view” button located in the button bar, the user can go up to the process that is described by the parent JDF node of the current JDF node. By clicking on a process with the left mouse button the user will expand that process view, hence one level down in the hierarchy. If a JDF node is selected in the tree view, the processes of that node are displayed, otherwise the process view for the top node is displayed. Clicking on a resource or a JDF node in the process view will also select the corresponding node in the tree.

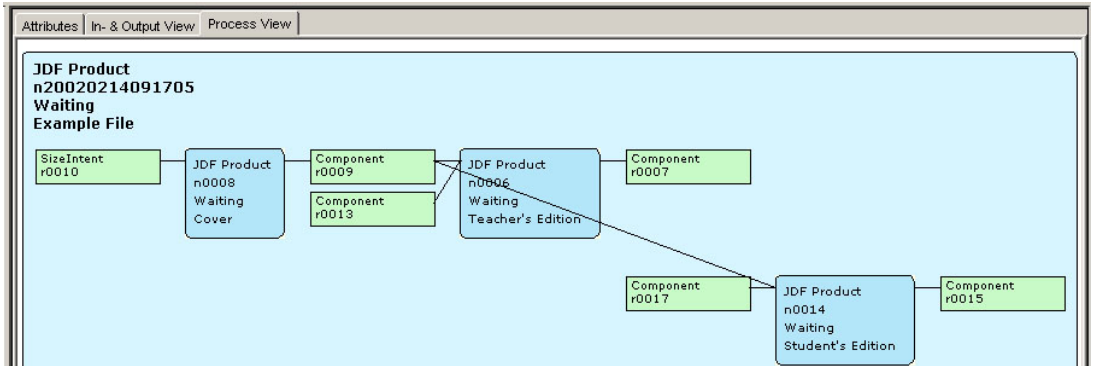

Figure 3.12 The Process View

When displaying the process view the input resources of the current JDF node are drawn, and then the algorithm looks at the JDF nodes inside the current node. First the JDF nodes inside have to be sorted. The sorting algorithm sorts the JDF nodes after the resources it inputs and outputs, so that they could be drawn in the right order, as a chain. JDF nodes that do not share resources with other nodes are put at the end of the vector storing the sorted nodes. When the sorting is done the process chain is drawn from left to right. When the nodes and resources inside are drawn, the parent JDF node is drawn outside so that the processes it contains also are shown inside. Finally the output resources of the current node are drawn. The most complex task was the sorting algorithm and to keep track of the position of where to start draw the next resource or JDF node. This could be especially hard when many different JDF nodes use the same resource.

The connections between the different views
In order to be able to navigate the file the error view, the process view and the input and output view are all connected to the main view, the tree view. Clicking on an element in one of these views will result in that the corresponding tree node gets selected. This also works the other way around. Selecting a node in the tree view will load the current data in all the other views, including the attribute view.

3.2.6 The Validation

The validation is done programmatically and is based on algorithms in an existing CIP4 Java application, CheckJDF. There are methods in the CIP4 JDF Library for finding unlinked resources, dangling links, missing attributes etc.

The first thing the application does when a file is being validated is that it checks the whole file for unlinked resources. All errors found from this check are stored in a vector. The tree is built recursively and when a tree node is to be created from the current element, the element is checked for missing elements and attributes, unknown elements and attributes and missing elements and attributes, dangling resource links, missing links among others. The element is also checked against the elements in the vector containing the result from the first check for unlinked resources. If an error is found a Boolean within the tree node is set to be false, which means that it this node is invalid. The tree node gets a special icon in the renderer telling that this element or attribute is invalid. A list element for the error view is created and also stored within the node. If a JDF element has invalid children it shall be displayed as invalid as well, therefore a check is done after the validation of an element to see if the parent element shall be marked as invalid too.

3.2.7 Other Features

Internationalization
The application is available in five different languages: English, French, German, Spanish and Swedish. Only menus, tool tip help, dialogs and titles are internationalized. The result after validation in the error view and the long help are in English. The language can be changed under the tools menu. After choosing another language the user has to restart the application for it to update.

Look & Feel
The user can chose from the look and feels available on the computer. The look and feel can be changed in the Tools menu and is updated during run time.

Search function
Searching can be done in the tree view, the attribute view, the input and output view and the error view. To make the search in the right view it has to be activated through clicking in that view. The searching is not case sensitive and can be done forward or backward. The search starts from the selected position in the view. It is not possible to search in the process view.

The search always starts with finding the selected position to be able to start the search from there. The string at the position is compared to the search string, if it is a match it gets selected. The user then has the possibility to press the Find Next button to go to the next match. If the search ends without finding the string, the message "The string could not be found" is displayed in the search dialog. The search dialog remains opened until the user presses the Cancel button. In the tree a preorder enumeration is used for searching. This algorithm was chosen to get the result in the same order as it is displayed in the tree view, form top to bottom. In the attribute view and the error view the search algorithm just steps through all the list elements. The hardest part was the search in the input and output view, keeping track of the position when, stepping through the different trees. A preorder enumeration is used also here.

Help
In the Help menu there are two menu items, About CIP4 JDF Editor and Help. Choosing the first item will activate a pop-up window with general information about the application. The Help item will activate a frame with help about the application including how to get started, about the different views, the editor, the validation and known bugs. There are also suggestions of further development of the CIP4 JDF Editor.

Open with
The application also has a function for opening files, including the application itself, through the file explorer. If the user does a right mouse click and choose the Open With option, she/he can then chose to open the file with the CIP4 JDF Editor. The application is launched and the file is opened.

3.2.8 Testing

During the development the application was tested both by the CIP4 members and people at Heidelberg, who found it to be a useful tool for displaying JDF files and tracking errors during development.

There were some performance problems that were tracked with JProfiler [5] and were attended to. One of the most time consuming actions was the expanding of all the nodes in the tree that was done after the tree had been created. Therefore it was decided to display the tree collapsed when opened.

The application has been tested on both PC and Mac. The version for Mac has not been tested properly and some features were found to be wrong, these have not been attended to due to time limitations.

3.2.9 What does the CIP4 JDF Editor not Handle?

As mentioned in the beginning of this chapter undo, redo and drag and drop are features that are not supported in the application. Printing of the representations of the file in the different views is not either supported.

The application does not handle foreign and private namespaces. For the viewer this means that elements that belong to foreign or private namespaces are not displayed in the input and output view and the process view. The foreign and private namespaces can be highlighted in the tree view so that they can be easily detected.

Partition display and refElements are not supported by the application. A partitioned resource is a structured resource that represents multiple physical or logical entities, such as separated plates. refElements are used for inter-resource linking when it is necessary to reference resource elements directly from other resources in order to reuse information.

Combined processes cannot be viewed in the process view. A combined process is a group of processes that can be executed by a single device.

3.2.10 Bugs

When the input and output view is created the elements to be displayed there should be validated if the automatic validation is on. This validation is not done correctly.

There are some errors in the version of the CIP4 Java JDF Library used, which leads to errors in the application.

Sometimes when trying to open a file for the first time it cannot be opened. The reason why this happens is not known to the developers and has therefore not been corrected. The user can try to save the file and try opening it again; this is a trick that usually is successful.

3.2.11 User Test

The CIP4 JDF Editor was compared to the three commercial editors because they were, as mentioned before, in general better than the freeware. Testing all six would have been too time consuming. Four persons tested and compared the CIP4 JDF Editor with three XML editors, XMLSpy [2], Morphon [7] and XMLPro [8]. The user test was put together according to guidelines from the Society for Technical Communication (STC). [13] Only general aspects, such as those the CIP4 JDF Editor has in common with the XML editors, were tested. The test was performed so that the test user went through a series of actions, like opening a file, creating a new file, saving, inserting and editing attributes and elements, searching, navigation through the file, validation, finding the errors etc. The same file was tested in all the applications. Detailed interviews were done with the test users.

The test users were observed during the test session to see how she/he acted when performing the different tasks. How long time the test user needed for the different moments were also noted. In order to make the test user familiar with XML editors, she/he first tested one of the freeware applications, XML Notepad [11]. Since the interfaces of the applications to be tested were rather similar, the test user would probably, after a couple of tests, increase the understanding of the standard behaviour.

This could lead to that the test user would find the latest tested application to be the easiest to use. To avoid this possible source of error the applications were tested in different order.

After testing the editors the test user were asked the following questions about the CIP4 JDF Editor:

1. Is the system status always clear to the user?
2. Are the metaphors used recognizable?
3. Can the user easily navigate through the application?
4. Are the pictures used on the icons recognizable to users and do they facilitate the understanding?
5. Any unsupported tasks? (I.e. printing, source view)
6. Can the application be used without documentation/help?
7. Is the feedback time acceptable?
8. General opinion of /comments on the interface?

All test users were well-experienced computer users. They all had a basic knowledge of XML but they had not used an XML editor before. None of them were familiar with JDF.

The general opinions of XMLSpy [2] were that it was too complex for beginners and that it displayed too much information and too many options. One of the test users assumed that it is how experienced users want it. The different views, especially the graphical were appreciated for viewing, not for editing. The validation was easy to perform and the error messages were easy to understand. The user has to choose from a schema or DTD when creating a new file, which caused confusion.

The opinions of XMLPro [8] varied. All the test users agreed on that it was good for beginners to understand the structure of XML but one test user, the most computer experienced, found it simple but very unintuitive; especially inserting elements and attributes. The others found it logical, easy and simple. Validation was unavailable, and the user does not need to be concerned about schema or DTD's.

Morphon [7] was harder to get started for some test users. The opening process takes time because it introduces new notions. Inserting elements and attributes are done in different ways, which causes confusion. The validation is only possible when the source view is active and only one user figured that out after a quite long time. The test users found it hard to understand what the error messages meant and they would have preferred that the errors also were displayed graphically.

All the users found the CIP4 JDF Editor easy to get started with, partly because they did not have to worry about schemas or DTD's. Editing was found easy because it was done in similar ways both for editing elements and attributes. The test users found it hard to understand what the error messages meant but they found it helpful that the errors also were presented graphically. The system status was always clear to the test user, but it is worth mentioning that the file tested was relatively small. The metaphors and icons used in all the tested applications were quite similar and the test users found them easy to understand. One of the test users missed a source view, a search function in the help and undo and redo. The Attribute view was found unnecessary, since the attributes were displayed in the tree.

The interfaces felt familiar since their menus and buttons are designed as frequently used applications, such as MS Word, was a comment from one of the test users. The right mouse menu was frequently used for editing. When a test user was performing a new task, she/he tried to do it in the same way as it was done in the previous tested application. The help was very rarely used. One user preferred XMLSpy [2] because it looked most professional.

3.3 The need for JDF and JMF

3.3.1 State of the Industry Before JDF and JMF

Today's print shops are facing more complex jobs with a tighter time schedule. To be able to handle these new demands standards were developed. But these standards were limited because they only addressed certain aspects of a print job. The standards that existed, such as CIP3's PPF and Adobe's PJTF were limited and could not supply the automation that was needed. PPF was created to make it possible to run between multi-vendor products, so that a single job-ticket could control all those products. PJTF supplied a limited but precise way of controlling processes in prepress and a control over a variety of press processes. Even though they were insufficient they are the predecessors of JDF. [14]

Another issue was the many proprietary interfaces that were used in the products from different vendors. To make these products talk to each other new complex proprietary equipment interfaces had to be implemented.

3.3.2 Why is the Standard Needed and What does it Mean for the Vendor and the Print Shop, respectively?

Workflow automation can decrease the amount of operator interfaces required in the workflow chain. Workflow auditing can be used to analyze the process, discover systematical errors and bottlenecks. [15] Workflow automation is the main benefit of JDF and will help the print shops to deal more efficiently with the various jobs they face today.

Since JDF will make it possible for products from different vendors to communicate it can be a way for small vendors to gain a larger and different market. But this is true even for the larger vendors; if a customer, for example, is interested in a press from Heidelberg but wants to have an Agfa workflow this would not be a problem with JDF. And Heidelberg does not miss the sale of a press. That JDF is a sales argument is something everybody seams to agree on.

An end-user, employed at the print shop, is not supposed to view or interact with JDF files directly. The main benefit for them lies in is the possibility to track and steer the production and the new statistics of the production that can be gained through the stored information in the JDF file or through JMF. One other large benefit is the greater range of products to choose from. This will also cause the vendors to be better since they are exchangeable. The customer will usually need both MIS and production equipment to use JDF in order to get the full leverage from using JDF.

3.4 Status of the JDF and JMF Development Today

Most of the major vendors in the printing industry are members of CIP4. One exception is Quark. The reason can be that automation is not so important in the creative phase, where Quark is active; this is probably because it is hard to measure how time consuming the creative work is when a large part is not actually done at the computer. All the major press and prepress vendors are members. Of the prepress companies, which often are smaller than the companies in the other areas, about half of them are members. The MIS companies are also often small and many of them are not members of CIP4.

At both Heidelberg and Agfa all new products that are being developed have JDF support. Rainer Prosi [16] thinks that in general new products to a large extent will be completely based on JDF, while older products, only will get a new interface based on JDF. Using JDF for the internal system can be a good way for a company to standardize their systems and not having to define an interface from scratch. Koen Van De Poel [17] thinks that in the first phase JDF will mostly be used for communications, as an interface. The focus of the JDF development at Agfa is today in the creative phase and in the prepress area. They also have development that touches MIS and e-commerce. In the future one focus will also be on newspaper production. Tim Donahue [21] says that NexPress find adoption of open specifications and standards very important, both for the company and their customers.

Rainer Prosi [16] thinks that the first companies that will use and gain from JDF are conventional printing companies that earlier used PPF. JDF will replace PPF, but the other old interfaces will still be supported until all components in the workflow have been exchanged to components supporting JDF. Koen Van De Poel [17] says that it is likely that larger commercial printing companies will be the first to start using JDF compatible products. He also thinks that the last to adopt JDF are the small print shops that do a lot of different types of printing jobs, because it is hard to automate when the workflow differs widely.

The current version of JDF, JDF 1.1 is primary focused on the areas of prepress, press, postpress and delivery. JDF is not ready to be implemented in a newspaper environment yet. CIP4 is working on the JDF base conformance level Interoperability Conformance Specification (ICS) to define different levels, each specifying which parts of JDF and JMF that are supported. If two products conform to the same level they should work well together. Koen Van De Poel [17] says that the specification needs more management and for JDF to succeed it is more important to concentrate on interoperability and clarifying the standard with examples than support of every different value.

3.4.1 Interoperability Matrix

The interoperability matrix displays which processes a product can produce and consume. It also includes which level of JMF the product supports and if it has support for some special features of JDF, like partitioning and combined processes. If a certain product produce a process and another product consumes it, those two products make a testing pair. The interoperability matrix has been used during the test events in San Jose in May 2003 for matchmaking, in order to find potential test partners. The intention is that the matrix in the future shall be available on the CIP4 website and be updated by the companies themselves so it will be kept up to date also for future test events. All data in the matrix is not needed to draw conclusions about the JDF status but it was included because it is needed for test preparations. The interoperability matrix is available in Appendix D.

The interoperability matrix form was sent out to 72 vendors, all vendors that were members of CIP4 at the current date. 21 of the 72 vendors returned the form for one or more products, 23 said that they for some reason could not participate and 21 did not answer at all. The remaining 8 have either stated that they would send the information to us soon or that they have not decided if they shall participate or not.

From those who answered that they did not want to participate the reason was that they did not have any products ready for interoperability testing yet. They were either in the beginning of the development cycle, they did not develop JDF enabled products at all or they did not want to expose product data so long before release.

The matrix shows clearly that the development has primarily been focused on the prepress area, 12 of the products in the matrix are for prepress and there are 2 press products, 2 postpress products and 3 MIS or Production planning products. Many products fall under several categories. The matrix contains 10 released products and the rest of them were alfa or beta versions, the majority of the released products are prepress products.

3.4.2 Newspaper Production

When the first approach to add newspaper production functionality to the JDF specification was made, the newspaper working group tried to define JDF for all processes and resources used in a newspaper production workflow. This means that all the content should be described in JDF. That this approach is not suitable could be explained by the fact that a single newspaper could have over 1000 ads. If these would be described in JDF there would be as many resource nodes as ads, over 1000. This method would cause too much data to be handled in the tight time schedule that newspaper production has. Further on, if the newspaper has different ads for different regions, which are common in the US, the number of resource nodes would increase linearly. During the CIP4 technical conference in Barcelona in February 2003, the newspaper working group decided, due to the problems to define JDF for all processes and resources, to only concentrate on defining JDF for the interfaces between the planning and mailroom systems and also between the planning and the plate room systems. Doing it this way would mean that JDF would only partly be implemented in a workflow for newspaper production. Frank Theede [18] says that this is an important step to have set this delimitation. The reason why this approach was set is because only ppi Media, Heidelberg, Agfa and Müller Martini were present at the meeting and for these four companies this is the only area where JDF is needed at this stage.

One major difference from other printing forms is that newspaper production already is quite automated due to the tight time schedule and the fact that the workflow always looks pretty much the same. As mentioned before, Ifra has already taken one step towards a more automated workflow in newspaper production with the specification IfraTrack. The primary difference between IfraTrack and JDF is that IfraTrack only is for messaging, and has therefore no process control. In IfraTrack the messaging format is called IMF. One problem with JDF compared to IfraTrack is that the JDF specification is so comprehensive; it is over 500 pages while the IfraTrack specification is only 40 pages. Frank Theede [18] think that this could make it hard for small vendors to implement a JDF interface to their products; due to the lack of resources they have compared to larger companies. The interest in JDF from the vendors is greater than it was for IfraTrack when that standard was as young as JDF. But Frank Theede [18] says that if not JDF had existed there would be more IfraTrack implementations.

Björn Hedin and Fredrik Fällström [15] argues that for IfraTrack users that do not intend to make use of the additional functionality JDF offers, there is no reason to change to JDF. But they also conclude that the increased possibilities of workflow automation that JDF can give can be one reason to start using JDF and that there will be many products that support JDF in the future which will make it easier to integrate

tracking functionality with new systems. For a print shop facing a decision to implement either IfraTrack of JDF, the benefit with IfraTrack is that it is relatively easy to implement and that it is proven in practice. On the other side JDF can do more than just auditing. If JDF will become a success, the extra work will be worth it.

The CIP4 newspaper-working group has cooperation with Ifra to see whether it is possible to translate IfraTrack to JDF. This will not be a problem since there is not much content in IfraTrack. The opposite, translating JDF to IfraTrack, however will cause loss in too much functionality. Converting an existing IfraTrack implementation to JDF would require a lot of work today, but would hopefully be easier if or when CIP4 publish guidelines for such a task. [15] Using JDF instead of IfraTrack would bring great advantages to newspaper workflow systems but would require a completely new architecture for these systems and the adding of additional chapters to the specification. [19]

For ppi Media it was a natural step to use JDF as an interface to systems from other vendors as they were looking for a standard to use for the interface when starting to develop one of their new products. They found JDF useful for this purpose. IfraTrack is used internally in this product while JDF is used for the interface. Frank Theede [18] thinks that IfraTrack is going to disappear and JDF eventually will take its place. The big problem here is the same as for IfraTrack when it came; the print shop is not willing to pay extra for the functionality the new standard offers. [16]

The work with JDF for newspaper production has just started and many vendors are waiting to see what the other ones are doing, nobody wants to take the first step. Frank Theede [18] thinks that JDF puts a demand on the vendors to be better, since their products will become more easily exchangeable and that having JDF compatible products will become a sales argument.

3.4.3 MIS

European Print Managementsystem Association (Euprima) is an industry association of European MIS vendors that has 21 members. The main purpose of Euprima is to promote the data exchange using JDF between customers, vendors, subcontractors and the production in network structures.

JDF is a very important marketing aspect also among the MIS vendors. Among Euprima's members one of them has JDF in an application and three of them develop JDF solutions. Not so many MIS vendors are members of CIP4 though. Eckhard Bölke [20] says that it is a very difficult market situation in Europe for the MIS vendors today, which makes them very careful. The CIP4 open source libraries cannot be used by many of the vendors because they develop in other programming languages than those used in the open source libraries. Since the vendors often are small they do not have enough resources for programming. For all MIS vendors go that they want to sell more systems with the statement JDF enabled.

The problem for the MIS vendors lies within the complexity of the logical structure and the correctness of data input. The complexity of the correctness of data input means that it is the accuracy of the calculation or estimation that is the problem. For example 2.4 copies of a print sheet are correct as economical criteria but for JDF it is unsuitable. The production and the MIS need hence different information.

Efficient MIS has the focal point on process optimization and therefore JDF is a very important tool. JDF is a practical way to network technical and commercial systems in order to increase the efficiency at print shops. [6]

Since September 2000 Euprima discussed the lack of MIS companies involved in CIP4, but they have now started a constructive cooperation with CIP4 [20]. That JMF is not

so widely implemented yet is no longer a big problem for the MIS vendors, because of the cross-linking architecture Euprima developed. They also developed a target suggestion regarding JMF. In the summer 2003 the IRD [20] will start a project, which computes the economy of cross-linking solutions.

3.4.4 Digital Printing

A Variable Digital Printing (VDP) job comprises the definition of many customized sets of pages where every set can be used to fill a series of sheets in a finished booklet. The page content of one of these booklets is customized to appeal to the targeted end recipient. These customized pieces can for example vary in number of pages, page content and utilization of media. The workflow process requirements are evidently relatively complex where both content and finishing differs. [21]

Tim Donahue [21] explains that there are many VDP oriented Page Description Language (PDL) formats today, of which most are proprietary. The past few years it was realized that the market growth in the area of digital printing was not advancing. The reason was thought to be the lack of portability and interoperability among these formats. Another reason was the difficulty for commercial printers to enter into VDP because the current tools require the printer to be involved in all aspects of a job authoring and also the print production. If these two aspects could be separated, growth could be gained in the digital printing area. Authoring involves both database expertise and graphic design expertise.

The Print On Demand initiative (PODi), a consortium of vendors, attempted to create a non-proprietary VDP PDL, an XML based page layout format, known as Personalized Print Markup Language (PPML). PPML is a layout technology designed to contain a broad range of VDP RIP implementations with features that enable its support of proprietary solutions.

Another standard developed in 2002, PPML/VDX, use chosen subsets of PPML for specifying the page layout, PDF for specifying page content elements and JDF for specifying the product intent. PPML, used in this way, is intended to meet the strictest requirements of portability, interoperability and device independence in order to separate the authoring and the print production. [21]

The PPML data can be equipped with elements called TICKET_REF. These reference elements can refer to device parameter resources or to product intent resources. This means that the DPL data refers to parts of the job ticket instead of the job ticket referring to various documents and pages defined in the PDL data. This leads to relatively complex processing requirements where many combinations of device processing states may be established throughout the processing.

In JDF 1.1 the Digital Printing working group defined ResourceUpdate resources. These provide a mechanism that allows the variable data PDL to refer to individual resources or partitions of resources from within the PDL context. Tim Donahue [21] says that it is more or less a data driven process finite state machine approach. The problem with using this approach, ResourceUpdate for process control in a VDP production workflow, is that it can get very complex. Another problem is that it is not clear how imposition printing can be accomplished.

JDF does not make digital printing easier but it provides a more unified way for producers and consumers to achieve commonality in the expression of product intent and process control. The most complex aspect is the authoring of the VDP job and it requires cooperation between client, designer, IT expert and printer. Tim Donahue [21] says that it is likely that JDF will be extended to define this coordination. JDF will help the users to focus on the requirements of the job instead of learning new vendor proprietary tools.

In JDF the model of a digital printing system usually includes combined processes because the digital printer often handles many processes such as Imposition, Rendering and DigitalPrinting. Some devices even have finishing devices integrated. The current ICS2, Interoperability Conformance Specification level 2, is attempting to define a preferred model for a digital printing system with integrated finishing.

3.4.5 Time Perspective

At the Drupa 2004 tradeshow many products that support JDF and JMF will be presented. There will also be real-time feedback using JMF and different companies will have done more integration efforts. It will take some time before modules can be exchanged between different vendors without loosing some functionality, says Koen Van De Poel [17].

Eckhard Bölke [20] says that at Drupa 2004, JDF will be an important topic, but only a few practical examples of MIS solutions will be presented, and those will show only a low cross linking level. By Drupa 2008 the MIS products will have fully integrated high-level cross-linking solutions.

Another 5 years forward in time, at Drupa 2008, Rainer Prosi [16] expect one of four scenarios

- The worst scenario is that JDF is forgotten because it was too complex.
- Some vendors have implemented it and some have not. JDF is just another format.
- JDF version 1.x is like PDF, everybody use it.
- The last scenario would be that a simpler version of JDF, version 2.x, has been developed, which reduces some of the problems with the complexity. CIP4 will try again with a simpler standard. Like SGML that became XML.

3.4.6 Problems

The flexibility of JDF is a great advantage but it could also be a problem. Defining everything that is needed for every different area of the graphic arts industry is an impossible task. Therefore JDF is extensible. This makes it necessary for systems, which communicate with a system that have certain extensions, to also support them. If a vendor needs to make certain extensions to the JDF schema, those extensions can later on be included in the next version of JDF if they become broadly used. The vendors interviewed for this thesis do not seem to think of this as a big problem but emphasize the difficulties with multi dialects. If a system requires some optional parts of JDF it is possible that it still cannot exchange information with systems that does not support those parts. Problems with multi dialects can be solved with device capabilities. Capabilities mean that a component in a workflow needs to be able to communicate what it is capable of handling. Koen Van De Poel [17] says that one major problem is that there are too many ways to describe product intent and that the specification is very flexible for the producer of the JDF but gets very complex for the JDF consumer, the ICS will help to solve this.

At this point it could be hard for especially small vendors to implement JDF to their products because of the complexity of the standard. It takes time to understand the specification. CIP4 is aware of this problem and therefore has an open source C++ library and a Java library for the members to use as help to make the implementation easier. Larger vendors have more resources for this purpose. Koen Van De Poel [17] at Agfa explains that the difficult task is not to generate JDF and that small MIS vendors already do this. It is more complex though to interpret JDF correctly and provide real time feedback. This is the part, which can be hard for small vendors.

The two main reasons why JMF is not so widely implemented yet is because the lack of MIS companies involved, and because there are not so many devices that can understand http, which is the way the messages are sent. The MIS companies have become more active this year though. Koen Van De Poel [17] says that another reason why the implementation of JMF not has reached so far is because it touches the internals of a product and therefore requires a lot more developing efforts.

There are no rules for what is demanded of a product to be called JDF compatible. A product can be completely based on JDF, which means that it uses both JDF as an internal format and for the interface. But it can also just use JDF for the interface, which actually is all that is needed for it to communicate with other interfaces. The ICS will make it possible for products to conform to different levels of JDF.

3.4.7 JDF Version 1.2

In the next version of JDF, version 1.2, CIP4 has defined JDF for preflight and enhanced the capabilities constraints. The ICS is also new for version 1.2. Another process defined is digital file transfer, enhancements of transport protocol which means that SOAP has been defined as transport layer to send JDF and JMF within the workflow. JDF for newspaper printing will not be defined for this version. CIP4 plan to release version 1.2 in the third quarter of 2003 and at Drupa 2004 1.2 conforming products will be demonstrated.

4 Conclusions

4.1 CIP4 JDF Editor

The result from the user test showed that the interface was easy to grasp and well arranged which imply that the interface of the CIP4 JDF Editor was designed and implemented as intended. The metaphors and procedures used were similar to other applications, which makes it easy to get started. There are still some bugs and unsupported features, like printing and undo and redo, which would make the application better if they were implemented. It is possible that more time should have been reserved for testing, especially for the Mac version.

4.2 Status of JDF and JMF Today

It was very hard to get responses from most of the vendors, even after reminding them repeatedly. There was a lot of detailed data to input, so completing the survey became a quite time consuming task. I think that was discouraging for many of the vendors. That may also be the reason why some them left a lot of open points. Another reason could be that it is very sensitive to expose such detailed information about their products, since it will be exposed to competitors. Therefore we gave them the opportunity to supply the data anonymously, which some of the vendors did, but in general it did not seem to make them give more complete answers. If the composition of the interoperability matrix would not have coincided with the interoperability test events in San Jose I think it is likely that many vendors would not have been participating at all. I think that it would have been more interesting to see what the interoperability matrix would have looked like if the participation among the CIP4 members were mandatory and I also believe that it would have made a better base for making comments on the status of JDF. It is known that some of the companies that chose not to participate have products that are JDF enabled. Therefore it is hard to draw conclusions from the matrix.

The interoperability matrix was used during the test events in San Jose and was found useful for matchmaking. A total of 25 products were represented in the interoperability matrix. This shows that JDF actually is ready to be implemented and that the vendors themselves really stake on JDF. 4 of the products in the matrix have only planned support for JDF. It is important to remember though that the JDF concept is not fully proven in practice yet. It is hard to find negative voices about JDF today; the reason is probably that JDF has not been so widely tested by the end-users yet and all the information about the standard is to be found from the vendors, the developers of the standard. That a test event actually is possible to realize proves that the development has reached far and that JDF can be used in some areas of the printing industry. Everyone seem to agree on that Drupa 2004 is an important event for JDF and that many interoperable products will be presented then.

The companies that supplied data to the matrix were mainly companies developing prepress products and the reason can either be that the development has primarily been focused on the prepress area or that those companies just happened to answer. Only 7 of the 25 products in the matrix support some JMF level. The MIS vendors are becoming more active and therefore it is likely that implementations of JMF will be more common soon. JDF is ready to be used in some areas of the printing industry but the full benefits of the standard will not be seen before JMF is supported in the products. There is also a lot of cooperation between organizations and other standards, like IfraTrack to make them compatible.

The development of JDF has been going on for four years. CIP4 themselves seem to think that the standard has reached far and has been more adopted than expected in that time. Compared to IfraTrack many more companies are implementing JDF support in their products than was done with IfraTrack when that standard was in the same stage of development as JDF is today.

Almost every important actor in the graphic arts industry supports the development of JDF, which makes it likely that it will be the standard format for workflow automation in the future. But many voices say that the complexity is a problem, a problem that CIP4 is aware of. To make this complexity easier to handle, especially for the small developers, CIP4 should publish more examples, sample code and guidelines for how to implement a JDF supporting system. Every interviewed vendor agrees on that JDF is a sales argument; it is a buzzword in the printing industry.

5 Recommendations

The following features have not been implemented in the CIP4 JDF Editor and are suggested for the continued development:

- Schema validation
- Foreign and private namespaces
- refElements
- Partition display
- Undo and redo
- Drag and drop
- Source view
- Hide and show attributes in tree view

There have also been some problems with the Macintosh version that needs to be taken care of. We did not have the time to test the Macintosh version properly.

One other feature that could be interesting in an application like the CIP4 JDF Editor is the ability to execute the JDF process associated with the JDF file displayed in the editor. This could be done with the return URL setup to notify the editor so that it could be automatically reloaded after process execution. This would allow the user to see how process execution modifies the JDF and allow a quick means for testing processes. There are methods in the CIP4 Java JDF Library for submitting the JDF to process execution.

Another suggestion is to give the CIP4 JDF Editor the ability to extract a table from the content of a JDF file a certain product has generated. The table will contain information about all the possible processes, resources, elements and attributes in the JDF file. This table will be useful for interoperability purposes, a more automated way of putting together information than the interoperability matrix.

Many users prefer to view the file in the process view. Therefore it would be helpful to show the errors in that view as well. This could be done with a red border instead of a black around the invalid resource or JDF node. The error message could appear in the tool tip when dragging the cursor over that resource or JDF node.

The new version of the CIP4 Java JDF Library was not ready when our project ended, but using that instead would make the application faster. It also contains improvements to the validation, which will make it more correct.

The interoperability matrix will need to be updated, and it is intended that the CIP4 members shall do that themselves at the CIP4 website.

6 List of References

[1] http://www.cip4.org, JDF Specification Version 1.1 Revision A, 2003-04-07

[2] http://www.xmlspy.com/, 2003-06-11

[3] http://www.eclipse.org, 2003-05-27

[4] http://www.rational.com/, 2003-06-11

[5] http://www.ej-technologies.com/, 2003-06-11

[6] The Euprima Story board summary, Version 1.1, http://www.euprima.org

[7] http://www.morphon.com/, 2003-06-11

[8] http://www.vervet.com/xmlpro.html, 2003-06-11

[9] http://sourceforge.net/projects/pollo/, 2003-06-11

[10] http://www.xmloperator.net/, 2003-06-11

[11] http://www.webattack.com/get/xmlnotepad.shtml, 2003-06-11

[12] http://developer.java.sun.com/developer/techDocs/hi/repository/, The Java Look and Feel Repository, 2003-05-06

[13] http://www.stcsig.org/usability/, 2003-05-22

[14] http://www.cip4.org, JDF White Paper, 2003-05-07

[15] Hedin B., Fällström F., IfraTrack functionality using JDF methods and syntax, http://www.tu.se, 2003

[19] Müller T., IfraTrack production using the Job Definition Format JDF, http://www.ifra.com, January 2003

6.1 Interviews

[16] Rainer Prosi, Chief technical officer CIP4, Heidelberger Druckmaschinen AG, 2003-04-09

[17] Koen Van de Poel, chairman Prepress working group, Agfa, 2003-04-29

[18] Frank Theede, chairman Newspaper working group, ppi Media, 2003-03-05

[20] Eckhard Bölke, Euprima, IRD, 2003-05-08

[21] Tim Donahue, Chairman Digital Printing working group, NexPress Solutions, 2003-05-09

7 Appendixes

7.1 Appendix A – A JDF Sample File

```
<?xml version="1.0" encoding="UTF-8"?>
<JDF DescriptiveName="Example File" ID="id0123456789" Status="Waiting" Type="Product" Version="1.1" xmlns="http://www.CIP4.org/JDFSchema_1">
	<AuditPool>
		<Created Author="CIP4 JDFWriter" TimeStamp="2003-04-14T15: 13:10+01:00"/>
	</AuditPool>
	<JDF DescriptiveName="Teacher's Edition" ID="id1" JobPartID="0" Status="Waiting" Type="Product">
		<ResourceLinkPool>
			<ComponentLink Amount="100" Usage="Input" rRef="link1"/>
			<ComponentLink Amount="100" Usage="Input" rRef="link2"/>
			<ComponentLink Amount="100" ProcessUsage="Cover" Usage="Input" rRef="link3"/>
		</ResourceLinkPool>
		<JDF DescriptiveName="Teacher's Insert" ID="id2" JobPartID="2" Status="Waiting" Type="Product">
			<ResourcePool>
				<SizeIntent Class="Intent" ID="link4" Locked="false" Status="Unavailable">
					<Dimensions DataType="XYPairSpan" Preferred="612 792" Range="576 756~648 828"/>
					<Pages DataType="IntegerSpan" Preferred="240"/>
				</SizeIntent>
			</ResourcePool>
			<ResourceLinkPool>
				<SizeIntentLink Usage="Input" rRef="link4"/>
				<ComponentLink Amount="100" Usage="Output" rRef="link2"/>
			</ResourceLinkPool>
		</JDF>
		<ResourcePool>
			<Component Amount="100" Class="Quantity" ComponentType="PartialProduct" DescriptiveName="Insert" ID="link3" Locked="false" Status="Unavailable"/>
		</ResourcePool>
	</JDF>
	<ResourcePool>
		<Component Amount="100" Class="Quantity" ComponentType="FinalProduct" DescriptiveName="Teacher's Edition Component" ID="link1" Locked="false" Status="Unavailable"/>
		<Component Amount="2000" Class="Quantity" ComponentType="PartialProduct" ID="link3" Locked="false" Status="Unavailable"/>
		<Component Amount="1000" Class="Quantity" ComponentType="FinalProduct" DescriptiveName="Student's Book" ID="link5" Locked="false" Status="Unavailable"/>
	</ResourcePool>
	<JDF DescriptiveName="Cover" ID="id3" JobPartID="1" Status="Waiting" Type="Product">
		<ResourceLinkPool>
			<ComponentLink Amount="2000" Usage="Output" rRef="link3"/>
			<SizeIntentLink Usage="Input" rRef="link6"/>
		</ResourceLinkPool>
		<ResourcePool>
```

```
                        <SizeIntent Class="Intent" ID="link6" Locked="false" Status="Unavailable">
                                <Dimensions DataType="XYPairSpan" Preferred="612 792" Range="576 756~648 828"/>
                        </SizeIntent>
                </ResourcePool>
        </JDF>
        <JDF DescriptiveName="Student's Edition" ID="id4" JobPartID="3" Status="Waiting" Type="Product">
                <ResourceLinkPool>
                        <ComponentLink Amount="1000" Usage="Output" rRef="link5"/>
                        <ComponentLink Amount="1000" Usage="Input" rRef="link7"/>
                        <ComponentLink Amount="1000" ProcessUsage="Cover" Usage="Input" rRef="link3"/>
                </ResourceLinkPool>
                <JDF DescriptiveName="Student's Insert" ID="id5" JobPartID="4" Status="Waiting" Type="Product">
                        <ResourceLinkPool>
                                <ComponentLink Amount="1000" Usage="Output" rRef="link7"/>
                                <SizeIntentLink Usage="Input" rRef="link8"/>
                        </ResourceLinkPool>
                        <ResourcePool>
                                <SizeIntent Class="Intent" ID="link8" Locked="false" Status="Unavailable">
                                        <Dimensions DataType="XYPairSpan" Preferred="612 792" Range="576 756~648 828"/>
                                        <Pages DataType="IntegerSpan" Preferred="198"/>
                                </SizeIntent>
                        </ResourcePool>
                </JDF>
                <ResourcePool>
                        <Component Amount="1000" Class="Quantity" ComponentType="PartialProduct" DescriptiveName="Student's Insert" ID="link7" Locked="false" Status="Unavailable"/>
                </ResourcePool>
        </JDF>
</JDF>
```

7.2 Appendix B – A JMF Sample File

```
<?xml version="1.0" encoding="UTF-8"?>
<JMF SenderID="Controller1" TimeStamp="2003-05-06T19: 17:31+00:00" Version="1_1" xmlns="http://www.CIP4.org/JDFSchema_1_1">
        <Response Class="Parameter" ID="Link0123456789" ReturnCode="0" Type="QueueStatus" refID="0815" xmlns="http://www.CIP4.org/JDFSchema_1_1">
                <Queue DeviceID="Device1" PT:QueueEntryIDCnt="12" Status="Waiting" xmlns:PT="www.privateSchema.com">
                        <QueueEntry JobID="ID0123456789_1" PT:filename="File1.jdf" Priority="90" QueueEntryID="Device1_1" Status="Running" SubmissionTime="2003-02-17T09:18:50+00:00"/>
                        <QueueEntry JobID="ID0123456789_2" PT:filename="File2.jdf" Priority="0" QueueEntryID="Device1_2" Status="Held" SubmissionTime="2003-02-17T09:18:48+00:00"/>
                </Queue>
        </Response>
</JMF>
```

7.3 Appendix C – Evaluation of XML Editors

	Morphon (Commercial)	**XMLSpy** (Commercial)	**XMLPro** (Commercial)	**Pollo** (Open source)	**XML operator** (Open source)	**XMLNotepad** (Freeware)	**CIP4 JDF Editor**
Views	Tree Source (badly arranged)	Source Browser Authentic Schema Grid	Tree Source (in pop-up window)	Tree Source	Tree Source	Tree Source	Tree Source? Process Error
Editing concepts	Undo, redo Cut, copy, paste Drag & drop Edit namespaces	Undo, redo Cut, copy, paste Insert	Undo (only 1 step) Cut, copy, paste Drag & drop Right mouse menu	Undo Cut, copy, paste Drag & Drop Insert tree node	Undo Cut, copy, paste Move, replace, remove	Undo Cut, copy, paste Add, delete, duplicate Move, demote, promote Drag & drop	Undo, redo Cut, copy, paste Move, add, duplicate Drag & Drop Editing from right mouse click menu and main menu
Schema validation (When?)	Only DTD validation	Automatically when open, save and change view. A validate button. Re-validate button after change.	Only DTD validation	Only in tree view when the validate button is pressed W.f check only in text view	Only RELAX NG schemas and DTD validation.	Should be validated when the file is opened but we have not been able to get the validation to work	Yes When opened, saved, closed and choose from menu. When entering a new value?
Namespace handling	Yes (for DTDs)	Yes	No	Yes but does not work completely, problem with xsi.	RELAX NG supports namespaces.	Yes according to the spec. Have not tested because no validation.	Yes
Handle of invalid/not well formed (w.f.) XML?	No validation: does nothing abut invalid files Non w.f files cannot be open, gives error message	Checks if w.f. when opened and highlights the errors. Cant validate if not w.f Invalid files can be saved, but gives a warning.	Error message when not w.f files are opened, shows the error clearly	Non w.f files opens in text view, error message shows when trying to switch to tree view or press check w.f. button. W.f errors are highlighted. Gives good error message if invalid after validation.	Error message when non w.f. file is opened. Only text view can be opened then but is not editable	Cant open a not w.f file. Gives error message specifying the error.	Should be able to open an invalid file but give a warning and an error message, highlight the error. Edit non w.f files from source view?
Help	Very good help	Yes	Yes	No	No	Yes	Tool-tip help Long help

	Morphon (Commercial)	**XMLSpy** (Commercial)	**XMLPro** (Commercial)	**Pollo** (Open source)	**XML operator** (Open source)	**XMLNotepad** (Freeware)	**CIP4 JDF Editor**
Language	Java	Java	Java	Java	Java	?	Java
Source code	No	No	No	Yes	Yes	No, only code samples	
Window Manager	Swing	?	?	Swing	Swing	?	Swing
Comments	Pluggable Look & feel. Chose layout. 3 languages. Well arranged.	No tree view. W.f. check button. Very clear error messages. Spell check. Code generator. Takes time to learn all features to fully use XMLSpy. Well arranged. Can show the schema.	Cannot edit in the text window. Only 1 step undo. Find.	Badly arranged, no clear overview. W.f check button. Error message appears not in the warning box, hard to detect.	Pluggable look & feel. Can't see file name of open file. Special move menu. Hard to understand.	Very simple to understand & get started with. Different icons for elements. Cant open if invalid. Can write schemas but not DTD. Good error message when opening an invalid file. Find.	Look & feel Internationalization Unlimited undo, redo. Cross platform Search

7.4 Appendix D – The Interoperability Matrix

R = Restricted
C = Consume
P = Produce
* = Combine
PL = Planned

Type of Product: 1: Prepress output management system
2: Production planning and control system
3: MIS
4: Prepress JDF consuming Device (Trapper, Color Matcher, RIP…)
5: Press JDF Consuming device (Digital Press, Press Control System…)
6: Postpress JDF Consuming device (Cutting, Binding, Folding…)

JMF messaging levels 0-4 corresponds to the levels No messaging, Notification, Query support, Command support and Submission support

Product	ApogeeSeries3	ApogeeX	JDF Manager	MarkzNet	anonymous1	anonymous2	Normalizer	FastLane	Shinohara Machinery	ElecRoc	PrintSapiens	DALiM FiCELLE v3.0	DALiM TWiST v5.0	PrintNet	Vio Digital Workflow Application Suite / 4.2	anonymous3	JDF-Link	anonymous4	Hagen OA 7.1 with Press Connector	anonymous5	Best Colorproof / Screenproof 4.6.3	OS ABSYS	Best Remoteproof 1.2	MetaDimension	Prinect PrintReady System
JDF Features				PL	PL				PL		PL														
Partitioning	C, R	C, R	C, R				P, R	C				P, R	C, R					C				P,R		C	P/C
Combined Processes	C, R	C, R	C			C, R	C, R			P/C,R		P, R	C, R					C						C	P/C
ProcessGroup Nodes			C			C, R		C		P/C,R								C	P, R			P, R			P/C
Capabilities												C, R	P, R												
Product Intent Resources						C, R		C				C, R						P/C			P/C	P, R			C, R
Process Resources and Processes	C, R	C, R	C			C, R	P/C,R	C		P/C,R		P, R	P/C,R					P/C	P			P, R		C	P/C
Audits			P				P, R	P		C		C, R	P, R	P/C				P/C				C, R		P	P/C
Support for SOAP																									

Product	ApogeeSeries3	ApogeeX	JDF Manager	MarkzNet	anonymous1	anonymous2	Normalizer	FastLane	Shinohara Machinery	ElecRoc	PrintSapiens	DALiM FiCELLE v3.0	DALiM TWiST v5.0	PrintNet	Vio Digital Workflow Application Suite / 4.2	anonymous3	JDF-Link	anonymous4	Hagen OA 7.1 with Press Connector	anonymous5	Best Colorproof / Screenproof 4.6.3	OS ABSYS	Best Remoteproof 1.2	MetaDimension	Prinect PrintReady System
Intents																									Most, C
ArtDeliveryIntent														P/C	C, R										
BindingIntent								C, R						P/C								P, R			
ColorIntent								C						P/C								P, R			
DeliveryIntent														P/C								P, R			
EmbossingIntent														P/C											
FoldingIntent														P/C											
HoleMakingIntent														P/C											
InsertingIntent														P/C											
LaminatingIntent														P/C											
LayoutIntent								C						P/C								P, R			
MediaIntent								C,R						P/C								P, R			
NumberingIntent														P/C											
PackagingIntent														P/C											
ProductionIntent														P/C											
ProofingIntent														P/C											
ShapeCuttingIntent														P/C											
SizeIntent														P/C											
Process																									
Approval									P, R																
ColorSpaceConversion							P/C,R		P, R														C	P/C	
ColorantZoneDetails																									
Combined RIPing																								P/C	
ConventionalPrinting														P/C		P/C, R		P				P			
DigitalDelivery															P/C, R										
DigitalPrinting																			P/C						
IDPrinting																			P/C						

Process	Product	ApogeeSeries3	ApogeeX	JDF Manager	MarkzNet	anonymous1	anonymous2	Normalizer	FastLane	Shinohara Machinery	ElecRoc	PrintSapiens	DALiM FiCELLE v3.0	DALiM TWiST v5.0	PrintNet	Vio Digital Workflow Application Suite / 4.2	anonymous3	JDF-Link	anonymous4	Hagen OA 7.1 with Press Connector	anonymous5	Best Colorproof / Screenproof 4.6.3	OS ABSYS	Best Remoteproof 1.2	MetaDimension	Prinect PrintReady System
ImageSetting		C	C																						C	P
Imposition		C	C																		P/C, R				C	P
InkZoneCalculation											P, R															
Interpreting		C	C																		P/C, R					
LayoutPreparation																					P/C, R					
Preflight											P, R															P
Proofing											P, R															
PSToPDFConversion								P/C,R			P, R															
Rendering		C	C								P, R										P/C, R					
RIP'ing		C	C								P, R														C	
Separation		C	C								P, R															
Screening		C	C								P,R															
Trapping		C	C																		P/C, R					P/C
ResourceDefinition								P/C,R																		
Group				C																						
Combined				C																					C	
Collection				C																						
CoverApplication				C																						
EndSheetGlueing				C																						
Folding																					P/C, R		P, R			
Gathering				C																						
HoleMaking																					P/C, R					
Inserting				C																						
Stitching				C																	P/C, R					
Trimming				C																	P/C, R					
BoxPacking																							P, R			
Cutting																							P, R			

Product	ApogeeSeries3	ApogeeX	JDF Manager	MarkzNet	anonymous1	anonymous2	Normalizer	FastLane	Shinohara Machinery	ElecRoc	PrintSapiens	DALiM FiCELLE v3.0	DALiM TWiST v5.0	PrintNet	Vio Digital Workflow Application Suite / 4.2	anonymous3	JDF-Link	anonymous4	Hagen OA 7.1 with Press Connector	anonymous5	Best Colorproof / Screenproof 4.6.3	OS ABSYS	Best Remoteproof 1.2	MetaDimension	Prinect PrintReady System
JDF Process Resources	All C																								
CuttingParams																						P, R			
ByteMap																									P/C
ColorantZoneDetails	C	C																							
ColorantControl	C	C					P/C			P, R							C, R			P/C, R	P/C, R	P, R	P/C, R	P/C	P/C
Color																								P/C	P/C
ColorantParams	C	C																							
ColorantOrder	C	C																							
ColorPool	C	C																						P/C	P/C
ColorSpaceConversionParams							C			P, R										P/C, R		P/C, R	C	P/C
ConventionalPrintingParams																	C, R		P						
Component							P/C,R										P/C, R		P	P,C		P, R			P/C
Device																			P	P/C, R		P, R			
DigitalDeliveryParams															P/C, R										
DigitalPrintingParams							P/C,R													P/C					
ExposedMedia	C	C															C, R				P/C, R	P, R	P/C, R	P/C	P/C
FileSpec Layout	C	C																						P/C	P/C
FitPolicy																								C	P/C
FoldingParams																				P/C, R					
FontParams							C			P, R												P, R			
FontPolicy										P, R															
HoleMakingParams																				P/C, R					
IDPrintingParams																				P/C					
ImageCompressionParams							C																		
ImageSetterParams							P/C, R			P, R														P/C	P/C
Ink																								C	P/C
InkZoneCalculationParams										P, R															
InterpretingParams		C																						C	P/C
LayoutPreparationParams							P/C, R																	C	
Layout		C						C		C, R														C	P/C
LayoutElement	C	C																						P/C	P/C
Media	C	C								P, R							C, R		P	P/C	P/C, R	P, R	P/C, R	P/C	P/C

Product	ApogeeSeries3	ApogeeX	JDF Manager	MarkzNet	anonymous1	anonymous2	Normalizer	FastLane	Shinohara Machinery	ElecRoc	PrintSapiens	DALiM FiCELLE v3.0	DALiM TWiST v5.0	PrintNet	Vio Digital Workflow Application Suite / 4.2	anonymous3	JDF-Link	anonymous4	Hagen OA 7.1 with Press Connector	anonymous5	Best Colorproof / Screenproof 4.6.3	OS ABSYS	Best Remoteproof 1.2	MetaDimension	Prinect PrintReady System
JDF Process Resources	All C																								
ObjectResolution	C	C																						C	P/C
PDFInterpretingParams	C	C																							
PDFToPSCoversionParams																								C	P/C
PreflightProfile										P, R															
ProofingParams										P, R										P/C, R			P/C, R		
PSToPDFConversionParams							C			P, R															
RenderingParams	C	C					P/C, R			P, R														C	P/C
RunList	C	C					P/C			P, R									P/C	P/C, R			P/C, R	P/C	P/C
ScreeningParams	C	C								P, R														P/C	
ScreenSelector	C	C																						P	
SeparationControlParams	C	C								P, R															
SeparationSpec																								P	
Sheet																					P, R			C	P/C
Signature	C	C																							
Surface																								C	P/C
StitchingParams																			P/C, R						
TransferCurvePool										P, R															
TrapRegion	C	C																							P/C
TrappingDetails																									P/C
TrappingParams	C	C					P/C, R																		P/C
TrimmingParams																			P/C, R						
Partitioned Resource and Keys																									
Separation																P/C, R									
SheetName																C, R									
Side																P/C, R									
AuditPool update in supplied JDF file by receiver																									
Notification																P, R									

Product	ApogeeSeries3	ApogeeX	JDF Manager	MarkzNet	anonymous1	anonymous2	Normalizer	FastLane	Shinohara Machinery	ElecRoc	PrintSapiens	DALiM FiCELLE v3.0	DALiM TWiST v5.0	PrintNet	Vio Digital Workflow Application Suite / 4.2	anonymous3	JDF-Link	anonymous4	Hagen OA 7.1 with Press Connector	anonymous5	Best Colorproof / Screenproof 4.6.3	OS ABSYS	Best Remoteproof 1.2	MetaDimension	Prinect PrintReady System
Act on Scheduling attributes supplied in NodeInfo		Yes																							
Start																C, R									
End																C, R									
JMF: Responses to NodeInfo queries in the supplied JDF file																				P/C, PL		P/C, PL			
StatusQuery														C, R											
JMF: Capabilities report in response to a KnownDevices Query											All P/C	All P/C													
JMF: Other Query & Response with Subscription/Signal																									
Events																C, R									
KnownMessages														C, R											
KnownJDFServices														C,R											
Status																C, R									
Signal-Response											P/C,R	P/C,R													
JMF Queue Support Commands											All C	All C											Yes, C	Yes, P/C	
SubmitQueueEntry											C, R	C, R													
QueueStatus																C, R									
Execution Model																									
Serial Processing													P/C,R												
Parallel Processing													P/C,R												
Test Running																									
Support for test runs													P/C, R												
Others																									
output resource location etc. update in supplied JDF file													P/C												

Product	Company	JDF Version		JMF Version and Levels				Product Status	Type of Product
				JMF 1.1/JDF 1.1A		JMF 1.2			
		Produce	Consume	Produce	Consume	Produce	Consume		
Apogee Series3	Agfa-Gevaert N.V.		1.0					Released	1 & 4
:ApogeeX	Agfa-Gevaert N.V.		1.0,1.1					Released	1 & 4
JDF Manager	Müller Martini		1.1, 1.1A, 1.2	0	0	0	0	Alpha/Beta	6
MarkzNet, Current version is 1.6	Markzware Software	1.1, 1.1A	1.1, 1.1A					Released	4
anonymous1	anonymous	1.1, 1.1A, 1.2(P)	1.1, 1.1A, 1.2(P)	1	0	P	P	Alpha/Beta	5
anonymous2	anonymous2		1.1, 1.1A					Alpha/Beta	6
Normalizer	Adobe Systems Incorporated	1.1 / 1.1A	1.1, 1.1A	0	0	0	0	Released	4
FastLane	Esko-Graphics		1.0, 1.1, 1.1A	0	0	0	0	Alpha/Beta	1 & 4
	SHINOHARA MACHINERY. CO., LTD.								
ElecRoc Workflow Management System, (2.0 Beta Version)	Beijing Founder Electronics Co., Ltd.	1.0	1.0 (Internally)	N	N	To be determined		Alpha/Beta	1 & 4
PrintSapiens	Olive Inc.							Released	
DALiM FiCELLE v3.0	Dalim Software GmbH	1.2	1.2			4(R)	4(R)	Alpha/Beta	1, 2 & 4
DALiM TWiST v5.0	Dalim Software GmbH	1.2	1.2			4(R)	4(R)	Alpha/Beta	4
PrintNet	ppi Media GmbH							Alpha	
Vio Digital Workflow Application Suite / 4.2	Vio Worldwide Limited								
anonymous3	anonymous3	1.1, 1.1A						Released	2 & 3
JDF-Link	MAN Roland Druckmaschinen AG								
anonymous4	anonymous4		1.0, 1.1, 1.1A, 1.2	2(3)	2(3)	2(3)	2(3)		5
Hagen OA 7.1 with Press Connector	Printcafe Software, Inc.	1.1, 1.1A	1.1, 1.1A	4(R)	1			Alpha/Beta	3
anonymous5	anonymous5								
Best Colorproof / Screenproof 4.6.3	Best GmbH, A division of EFI	1.0, 1.1, 1.1A	1.0, 1.1, 1.1A	0	0	0	0	Released	4
Best Remoteproof 1.2	Best GmbH, A division of EFI	1.0, 1.1, 1.1A	1.0, 1.1, 1.1A	0	0	0	0	Released	4
MetaDimension	Heidelberger Druckmaschinen AG		1.1, 1.1A		4 (R)			Alpha/Beta	1 & 4
Prinect PrintReady System	Heidelberger Druckmaschinen AG	1.1, 1.1A	1.1, 1.1A	4(Internally)	4(Internally)			Alpha/Beta	1 & 4
OS ABSYS	Orga Soft HmbH	1.1, 1.1A						Released	2 & 3