JDF-Integration and Automation

Thomas Hoffmann-Walbeck

The JDF-integration of production steps in a print shop is presented. A production example is analyzed concerning the JDF-interfaces between the different components of a well-known workflow management system. The article shows screen-shots of the GUI of different applications, as well as the JDF-output which the applications are generating. The JDF-code will be explained and with it the structure of this particular XML. In particular, the JDF-terms product intent nodes, process node, process groups, resources and gray boxes are discussed. Different examples of JDF-resources are given, like MediaIntent, TrappingDetails, InZoneProfile, Preview and CuttingParams.

Introduction

In this paper the integration of production steps by cross-linking JDF-data is presented. The installation at the Stuttgart Media University is taken as an example. There, a couple of machines in Prepress, Press and Postpress have been connected via JDF recently. Also, an order management system is part of the JDF-network. But, it goes without saying that the installation is far of being complete.

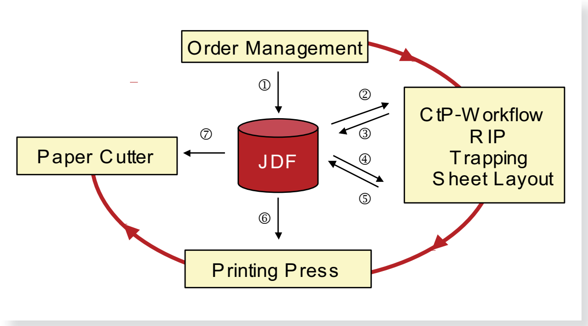
This article shows the GUI of individual applications, which play a role in this JDF-network, as well as the actual JDF code, which these applications are generating. Along the line a short introduction to the JDF data structure will be given.

Besides, experiences, as well as about prerequisites and consequences of the JDF integration will be discussed.

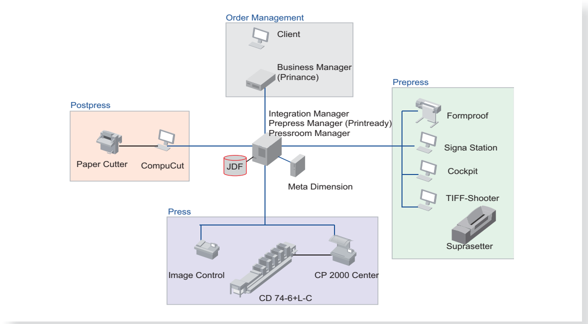
The typical production steps in a commercial print shop are starting with the order management, followed by the CtP-workflow and the printing process. In illustration 1 Postpress is represented by a paper cutter.

The red arrows represent the delivery of physical resources such as printing plates, sheets or folded sheets. In addition, there normally is a paper-based job jacket, which is generated by the order management system and passed on from one department to the other, supplemented by the operators.

Presently, only little data are exchanged electronically between the departments. Most well known example in this respect surely is the presetting of the ink zones with the help of the Print Production Format (PPF), also called CIP3-format. The prepress-RIP provides the fundamental PPF-data for the pre-setting, i.e. images of the sheets, and special software calculates the actual values and passes them on to the offset printing press.

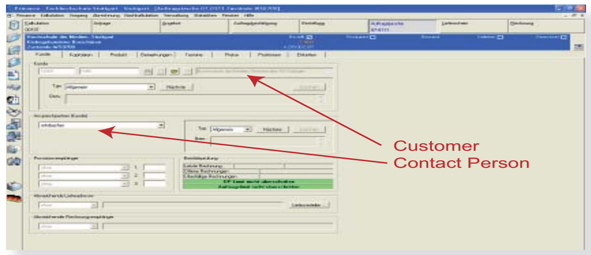


1 - Production steps

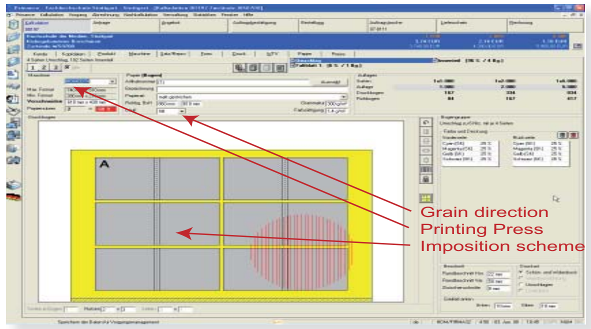


2 - JDF-Integration

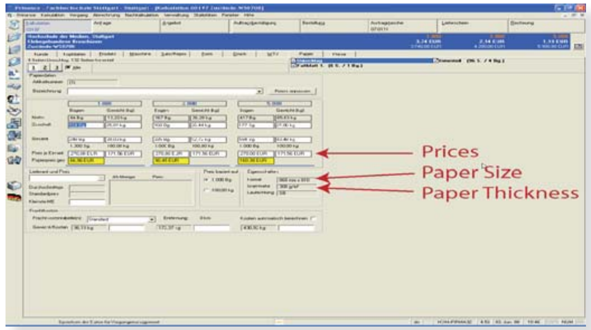
Many other production parameters, which are already entered into the order management software in order to calculate a quote for a customer, must be read from a paper job jacket and entered into other software over and over again. That is, of course, not particularly effective and increases the risk of errors. Only within small ranges of the print production – mainly within a department – there is an electronic information interchange between individual software modules. Here, above all, the CtP Workflow software is to be mentioned, which actually represents the germ cell of the idea of the overall cross-linking.



3 - MIS Customer Information



4 - MIS Machine Details



5 - MIS Paper Details

Workflow Configuration

In picture 2 the structure of the JDF integration at the HdM can be seen. By the product names one can tell now that the Prinect software from Heidelberger Druckmaschinen has been installed. However, not products should be discussed in this paper, but only the intrinsic procedures of the software. Those are rather similar with other manufacturers, for example, the Prinergy Workflow from Kodak or the Scope Workflow from Esko.

In the upper rectangle of the illustration 2 there is the client/server system of the order management software Prinance. The job data are passed on from there to the production server, per job. On this server not only all the JDF-files for the jobs are stored, but also the Prepress Manager (Printready). This software used to control the CtP Workflow only, now, however, it is the headquarter for the entire JDF integration.

Various other software modules have access to this server, for example the imposition software Signa Station, the control systems of the printing press CP2000 and the paper cutter. And, of course, also the client software of the Prepress Manager, the so-called Cockpit.

The communication between the order management system and the production server is in our place, for the time being, still unidirectional. Prinance cannot read JDF-files yet. Thus, recalculation of the jobs using operational data is not yet possible in this configuration. Also production changes are not written back into the order management system. This will change in due time.

But what are the prerequisites for such JDF integration? Obviously one needs a stable local network between all departments and all devices that are involved. Beyond that, one also needs one or several central servers, on which all players have access to. The access should be controlled by user rights, of course.

The most important and perhaps also most difficult point, however, is that the JDF integration presupposes standardized operational rules. This does not mean that the print products have to be standardized or that there is only one manufacturing path. But the net of production paths must be ordered and well defined – JDF-integration does not go along very well with a high level of improvisation. For example, it is not possible right now to change a job in the order management system when the production of the job has been started.

Finally, a JDF-workflow makes a lot of sense if, but may be only if, the involved devices can directly use the new information. That means: The machines must be pre-settable electronically.

In this paper the modules order management, CtP Workflow, printing press and paper cutter are examined exemplarily. Also the JDF-data that is exchanged between those four modules are analyzed. Therefore, the JDF interfaces in picture 1 are marked with numbers between 1 and 7. Throughout this paper these interfaces will be discussed according to these numbers.

Order Management

It starts with the order management software, in which important production details are entered. Some of these values are exported to a JDF file in interface 1 and fi nally used by the following production processes.

As a production example, a student's booklet, which has been produced in 2007, has been chosen.

The screenshots in illustration 3 to 5 are showing different production characteristics like customer, contact person, imposition scheme and so forth, entered by the operator into the order management software.

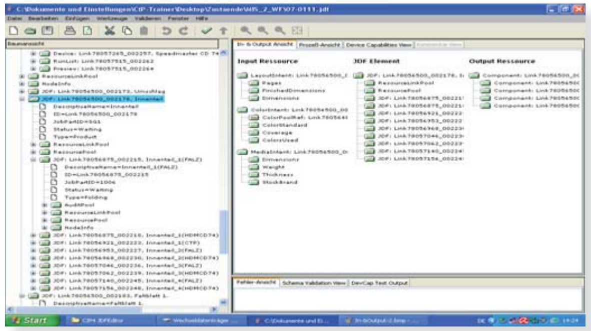
Finally, the software calculates the manufacturing costs and the selling-prices for the different print runs and thus a quote can be printed for the customer. If the customer approves the quote, the quote is transformed to an order by the software and a job jacket is printed This is passed on to the production.

That's it for the order management system in many configurations. In a JDF-environment, however, parts of the production parameters are exported to JDF and passed on to subsequent software modules.

JDF Structure

Before looking at the actual JDF code, please notice the representation of the JDF file as it is shown by the JDF editor, which can be downloaded from the official JDF Website [1]. As once can see in illustration 6, the JDF file consists of JDF elements – also called JDF nodes - and many resources.

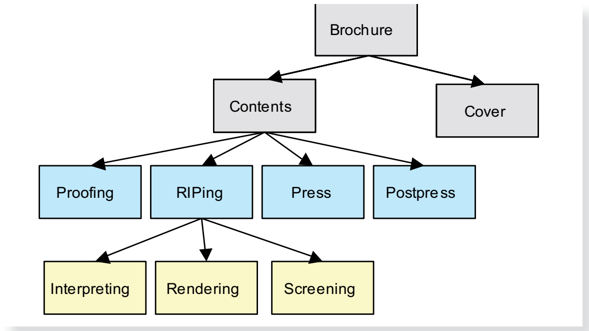
These resources are either input or output re-


6 - JDF Nodes and Resources

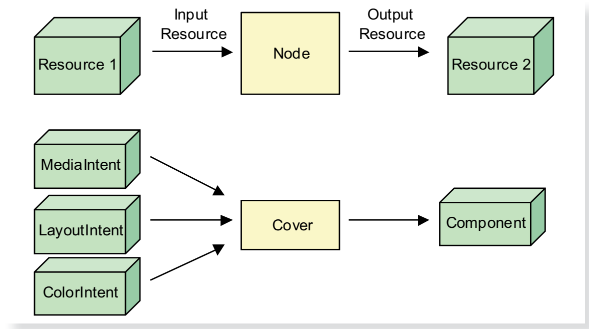
sources for the JDF elements. Not only the print product itself can be described by a network of elements and resources, but also the production of print products with all it's dependences.

Resources are differentiated into physical resources and parameter resources. Physical resources are things such as offset plates, sheets or folded sheets. Parameter resources are non-physical things such as parameter sets or fi les, but also persons.

JDF nodes are divided into 3 classes. There are product intent nodes that can define an intended print product. Then there are process nodes, which define individual steps in the production process. Several processes can be grouped together to process groups. In the example in illustration 7 processes **Interpreting**, **Rendering** and **Screening** are combined to the process group **RIPing**.



7 - JDF-Tree



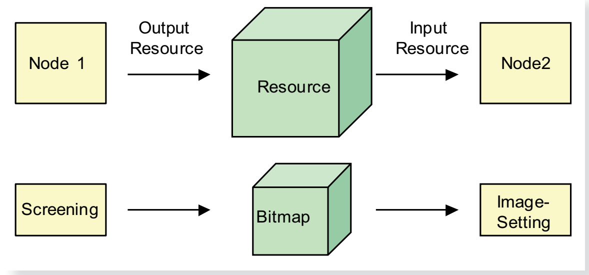
8 - Resources and Product Intent Node

```
<MediaIntent Class="Intent" ID="Link33593482_000019" ProductID="(2)„
 Status="Available">
   <Dimensions Actual="2834.6456692913384 1984.251968503937„
    DataType="XYPairSpan" Preferred="2834.6456692913384
    1984.251968503937"/>
   <Weight Actual="170.0" DataType="NumberSpan" Preferred="170.0"/>
   <Thickness Actual="170.0" DataType="NumberSpan„
    Preferred="170.0"/>
   <StockBrand Actual="" DataType="StringSpan" Preferred=""/>
   <Comment/>
</MediaIntent>
```

9 - Resource Media Intent

```
<Company Class="Parameter" DescriptiveName="HdM" ID=
  "Link33592745_000004" OrganizationName="Hochschule der Medien„
   Status="Available"/>
<Address City="Stuttgart" Class="Parameter" Country="Deutschland„
   CountryCode="DE" ID="Link33592745_000005" PostalCode="70569„
   Status="Available" Street="Nobelstraße 10"/>
…
<Person AdditionalNames="" Class="Parameter" DescriptiveName=
  "rohrbacher" FamilyName="Rohrbacher" FirstName="Jörg„
   ID="Link33592839_000007" JobTitle="" NamePrefix="Herr„
   NameSuffix="rohrbacher" Status="Available">
   <ComChannel ChannelType="Phone" Locator="+49 (0)711 8923 2236"/>
   <ComChannel ChannelType="Phone" Locator="+49 (0)711 8923 2897"/>
   <AddressRef rRef="Link33592839_000009"/>
</Person>
```

10 - Resource Company



11 - Process Nodes with Resource

The nodes are hierarchically structured. At the root, that is at the top, one finds the abstract description of the product and the further down one gets, the more detailed the information becomes. At the top of the little model in illustration 7 there are the gray colored product intent nodes, one line down the blue marked process group nodes and finally the yellowish process nodes.

The tree only shows the nodes, not the resources. Those will connect the nodes and thus define the dependencies and dynamics between the nodes. For example, a process node can only be executed, if all input resources are available.

Illustration 8 shows an example of the relationship between a Product Intend Node and it's 3 input resources and a single output resource. Input resources define the printing material, the imposition scheme and the colors, while the output resource represents the cover being a component of the final product.

There are, of course, no small rectangles and connecting arrows in the JDF-code. The XML-code of the JDF-tree defines the dependencies simply by embedding sub-elements, as to be seen on code example 9.

Each element is characterized by attributes and their values. The node **MediaIntent** contains a subelement **Thickness**, which has the attribute **Thickness Actual**; the value of the attribute is 170, that is 170 micrometers. Another subelement **Dimension** has the attribute **Actual**, which describes the dimension of the untrimmed sheet. With this attribute the unit is 1/72 Inch, so that, if one transform the values, we get the format 70x100 cm.

In the resource **Company** in 10 the customer name is stored, in the resources address and person the address and the phone numbers of the contact person, as entered into the MISsoftware.

In illustration 11 there are two process nodes: **Screening** and **ImageSetting**. One Resource between the two nodes connect the two process steps. The **Screening**-process produces the resource **Bitmap**, which is consumed by the **Image-Setting**-process when the plates are imaged. With the help of such links complex production procedures can be described.

In 12 there is special process group node, which has been written by the order management system into the JDF file. The group node **Umschlag (CTP)** lists the processes **Imposition**, **RIPing**, **Preview-**

Generation and ImageSetting.

Those processes are not specified in detail. No attributes of these processes are set and also not all the necessary resources. Only a formal container is defined for these processes. The order management system actually cannot supply a lot of details concerning these processes, thus it writes only this formal group node into the JDF-file.

Such kinds of group nodes are called Gray Boxes. Subsequent processes are supposed to fill such a Gray box with „life“, that is, specify are necessary details.

```
<JDF Category="FinalImaging" DescriptiveName="Umschlag(CTP)„
   ID="Link33597446_000044" JobPartID="1004" Status="Waiting„
   Type="ProcessGroup" Types="Imposition RIPing PreviewGeneration
   ImageSetting">
…
</JDF>
```

12 - Resource Node and 2 Process Nodes

Prepress

In the next section we will explain how the CtP Workflow system takes over the JDF data from the order management system.

In the CtP-workflow there are several modules involved, in particular the job administration, a RIP, a trapping engine and a sheet layout software. Hence, within the CtP Workflow JDF-data are read and written several times by different modules. Here, we simply want to analyze 2 reading and writing actions, that is 2 to 5 in illustration 1.

With the help of the JDF file that the order management software produced (interface 2), an appropriate job is defined automatically also in the Prinect production environment. Many values, e.g. those concerning the sheets, the printing press or the customer, are taken over from the order management system. In the Prepress Manager software production sequences can be defined, which might cover all production steps from preflighting PDF-pages to exposing plates. Also, the computation of ink pre-setting values for the offset press and other pre-setting values for some Postpress device are initiated in these sequences.

By the JDF-information the software can provide the correct sequences to the operator automatically. Please observe in screen shot 13 that the sequences for content and cover of the booklet and that the sequences show the correct printing presses HDMCD 74.

The operator can now change individual parameters of the sequences for this particular job or he or she can execute the sequence without modification.

The CtP-workflow system on the other hand also generates many JDF elements (interface 3). When the operator executes a production sequence, the Prepress Manager writes some of the parameters that are coupled with the sequences into a JDF-file. This way, new nodes and resources are generated. In particular, the Gray Box can be dissolved and replaced by proper processes or so-called combined process groups.

For example, it can be seen clearly that some trapping parameters are written into the resource TrappingDetails in 14.

One can also see that some attributes begin with the prefi x HDM. Those are proprietary attributes from the manufacturer and they are not part of the JDF-specification. This is quite typical for JDF data, but also for other job ticket formats like PPF [2] or PJTF [3]. The developers of the JDF-specifi cation cannot foresee all possible eventualities of print production. So, in order to transport the desired functionality via JDF, the manufacturer is forced to cook up his own attributes, his own nodes and his own resources.

This does not do any harm, as long as this kind of JDF is an internal interface of a manufacturer-specifi c system solution. As a consequence we learn that users are not supposed to assemble RIP software, trapping engines and job management systems of different manufacturers to an individual CtP Workflow solution (at least not within Prinect).

The workflow software also calculates the ink zone pre-setting for the offset press and shows the results in a diagram in picture 15. This information, again, is written into JDF, so that the printing press has access to it.

The zone widths and the number of zones are defined in the resource **InkZoneCalculationParams**. In the **InkZoneProfile**-resource there are the actual pre-setting values for each zone. Please observe 16.

The name of the preview-file and the path in the fi le system is specified in the resource **Preview** in 17. The actual graphic data are not stored in the JDF structure.

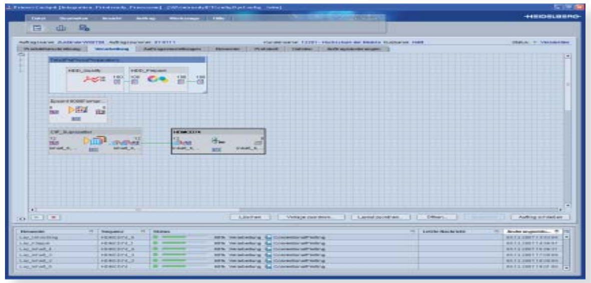
Reading JDF data (action 4 in picture 1) can give rise to powerful automation. This can be noticed especially when looking into the imposition software. Here, the operator must actually only confirm the pre-setting of the software, though he or she still can modify the imposition sheet if desired.

Picture 18 shows the GUI of the sheet layout software. This screen shot was made after importing the JDF data without further modification by any operator.

Please observe that the plate size and the paper parameters are taken out of the JDF-file. In case of a saddle-stitched brochure even the shingling values, for examples, are calculated automatically from the paper thickness, which has been entered in the calculation software a few process steps before.

Also, the imposition scheme can be read from the JDF-file. Thus, the imposition sheet for the print product can be generated automatically. In the sheet layout software cut marks and folding marks are set. That is, the positions where the sheets must be cut and folded are specified in this process step. These values are registered in interface 5 into the JDFfile, so that JDF compatible software controlling the paper cutter and the folding machine can actually use these values for computing cutting or folding programs.

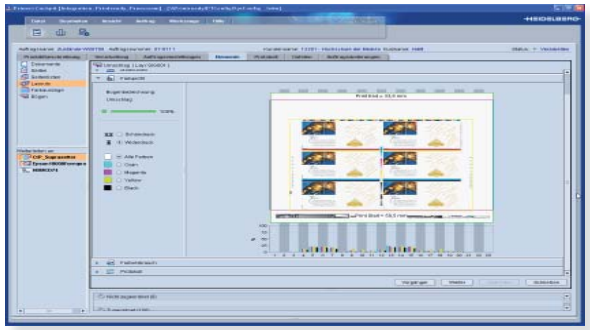
In addition, the sheet layout software produces a PDF-file, in which all marks are stored in their graphical characteristics and in the correct position for each sheet layout. So it is possible to assign the PDF-pages to the imposition sheet including all the marks in a later imposition process. In the JDF-file in the code block 19 only so-called **CutBlocks** are defined, that is, only the positions of the cuts, but not the cutting order. The cut sequence is computed by



13 - Production Sequences

```
<TrappingDetails Class="Parameter" DefaultTrapping="true„
…
<TrappingParams BlackColorLimit="0.95" BlackDensityLimit="1.6„
 BlackWidth="1.0" Enabled="true" HDM:BlackOverprintFonts="true„
 HDM:BlackOverprintFontsLimit="12.0„
 ...
 HDM:ThinLineWidthScalingLimit="0.0" ImageMaskTrapping="true„
 ImageToImageTrapping="false" ImageToObjectTrapping="true„
 ImageTrapPlacement="Normal" SlidingTrapLimit="1.0" StepLimit="0.25„
 TrapColorScaling="1.0" TrapEndStyle="Overlap" TrapJoinStyle="Bevel„
 TrapWidth="0.25" TrapWidthY="0.25"/>
</TrappingDetails>
```

14 - Trapping Details



15 - Ink Zone Presetting

```
<InkZoneCalculationParams Class="Parameter" ID="Link33641623_021578„
 Status="Available" ZoneHeight="1451.338583" ZoneWidth="92.125984"
Zones="23" ZonesY="1"/>
<InkZoneProfile ...
 Separation="Cyan" Status="Available" ZoneHeight="1451.338583"
ZoneSettingsX="2.552319004555964E-4 0.23347108314479864
0.39557497878959436 0.3980868212669701 0.4201399297699865
0.4057804840686292 0.00965709841629265 3.1086244689504383E-15
0.0016204751131252747 0.00177077441553854733.1086244689504383E-15
0.050594421662898834 0.20640135982277744 0.20118931466817727
0.20425757682881085 0.21600602139894665 0.07153433729261217
0.0070316035067904134 0.0022279176093545324 0.002633979072401287
0.005992352469837172 0.0017822633861267828 0.008642180429867335"
ZoneSettingsY="0.12368044369713041" ZoneWidth="92.125984"/>
```

16 - JDF Code concerning Ink Zone Presetting

```
<Preview Class="Parameter" Directory="file://PRINTREADY/PTJobs/Jobs
 /2007-02-15/Studienführer%20SS2007(07 0005)/SheetViewables/HDMCD74„
 HDM:Disposition="Retain" ID="Link33644156_021681„
 PartIDKeys="SignatureName SheetName Side" PreviewType="Viewable„
 Status="Unavailable">
...
</Preview>
```

17 - JDF-Code concerning Previews

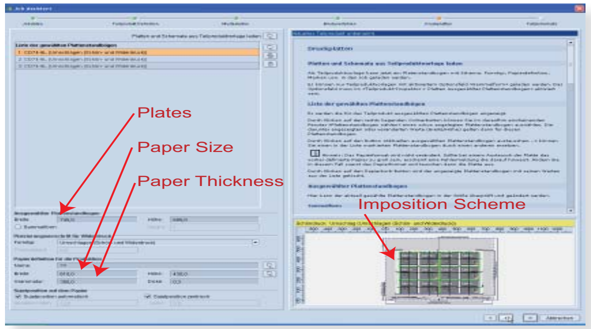
software related to the paper cutter using these cut blocks. The computation of the cutting program, however, is not part of the JDF specification, but implemented by proprietary solutions.

Press and Postprtess

What kind of information can the control device of the printing press get out of the JDF, that is interface 6 in illustration 1?

First of all, things such as a job number and the preview of the job. Also, the paper handling inside the press can be adjusted automatically with the help of the JDF-data concerning paper size and paper thickness. Finally, the pre-setting values of the ink zones are imported from the JDF data. The latter looks on the control device of the printing press quite similar to the one, which sees the prepress operator when generating the data (picture 15).

The JDF-interface 7 concerns the paper cutter. As previously mentioned, the JDF-file only contains cut blocks from which some special software such as Compucut can compute a cutting sequence. The operator can still modify his or her cut program, before it is sent to the paper cutter in a proprietary format via LAN. Next, the operator can call and execute the program at the cutter, while he or she will see all the necessary paper handling actions on the cutter's display.



18 - GUI of Layout Software

```
CuttingParams SignatureName="SIG001">
   <CuttingParams HDM:DummyParams="false" SheetName="Umschlag">
      <CutBlock BlockElementType="CutElement" BlockName="Umschlag_B_1_1,
       BlockSize="952.4399448818897 651.967539370079" BlockTrf="1.0 0.0
       0.0 1.0 39.68503937007881 0.0" HDM:CIP3BlockTrf="1.0 0.0 0.0 1.0
       103.46456692913398 140.31496062992125"/>
   <CutBlock BlockElementType="CutElement" BlockName="Umschlag_B_1_2"
    ... />
   <CutBlock BlockElementType="CutElement" BlockName="Umschlag_B_1_3"
    .../>
   <CutBlock BlockElementType="CutElement" BlockName="Umschlag_B_1_4"
    .../>
   </CuttingParams>
</CuttingParams>
```

19 - JDF-Code for cutting Blocks

Conclusion

The example has shown the JDF-potential with respect to automation. Not all production parameters, which actually can be passed on, could be presented. Also, it should be clear, that JDF integration is by all means not yet complete. For example, let us recall the unidirectional communication between the order management system and production management software for example. JDF-integration is not a product, which one can buy off the shelves, but a project, which demands a lot of time during the implementation phase.

We also saw that the JDF code can contain proprietary extensions, which could lead to incompatibilities. Or, saying it differently: Not all JDF-interfaces are suitably for connecting independent software modules of different manufacturers.

Many JDF-interfaces are only designed for internal use inside a specifi c workfl ow software package.

The Job Messaging Format (JMF) has been ignored in this article and with it machine data and operational data logging. This, however, would be an interesting topic by is own.

Finally a few remarks concerning teaching JDF/JMF to students in the graphical industry. This seems to the author quite important, simply because it's an amazing model of the entire printing process.

But every workflow needs system administrators at the production site. And thus the students got to understand the principles of JDF and JMF. Designing and testing production confi gurations is demanding. Not everything run as smoothly as expected, especially if software modules from different vendors are involved. The students have to get prepared so that they can meet these challenges.

But teaching JDF is very abstract and there is no much literature which can be used besides the specification. But there is some light at the horizon, because the CIP4 organisation prepares a textbook about JFD right now. Moreover, they want to publish a standard curriculum for teaching JDF.

References
[1] http://www.cip4.org/
[2] CIP3 PPF Specifi cation Version 3.0: http://www.cip4.org/
[3] Portable Job Ticket Format: Adobe Developer Support, Technical Note #5620, Version 1.1, 2. April 1999

Prof. Dr. Thomas Hoffmann-Walbeck

Stuttgart Media University (HdM), Nobelstrasse 10, D-70569 Stuttgart, Germany; hoffmann@hdm-stuttgart.de