The Maturation of the Job Definition Format: Changes in JDF 1.2

James E. Harvey*

Keywords: JDF, CIM, Process, Standards, Workflow

Abstract: There are hundreds of JDF-enabled products coming to market this year, but the ongoing development of the JDF standard continues. Harvey will provide a technical review of the key changes in JDF 1.2 and what some will mean to graphic arts systems in the coming year to two years. Some of the key changes include to JDF 1.2 include significant changes in device capabilities features that can be used to automate the "handshake" between devices; changes in data types, including new and deprecated data types; improvements in JMF — the Job Messaging Format; the addition of some postpress processes; and Changes that effect layout and rendering.

Introduction

The International Cooperation for the Integration of Processes in Prepress, Press and Postpress Organization or "CIP4" is a not-for-profit association. CIP4 is established in Switzerland, it has no regular offices; rather is a global organization with representatives several countries. Currently the CIP4 has about 220 members, but that number is hard to pin down because the membership is constantly growing.

CIP4 was formed in early 2001. Its predecessor, CIP3, was formed by Heidelberg and was managed by the Fraunhofer Institute for Computer Graphics. CIP3 created the Print Production Format or "PPF," which has found some success in ink key pre-setting and postpress operations. The PPF format was written in a proprietary format and CIP3 was considering the move to XML at the time. In the meanwhile, Adobe, Heidelberg, Agfa, and MAN Roland, sometimes referred to as the "gang of four," had put together an XML-based job ticket called The Job Definition Format or "JDF" and they asked CIP3 to take over stewardship of the specification, provided that CIP3 would reorganize as a public not-for-profit entity open to all, which it did as CIP4.

Adobe's Portable Job Ticket Format (PJTF) was another early attempt at creating a method for exchanging print metadata, as was Graphic Communications Association's Industry Architecture Project and IFRA's ifraTrack. Each metadata program had its own unique failings and lessons

*Media4theWorld

learned; hence, JDF is not a "first attempt," but rather builds on the experience gained from all of these efforts:

1. Embedding metadata into production files and/or PostScript (PJTF and ifraTrack) isn't going to work for many reasons: the least being that front-end systems cannot handle the size and volume of production files.
2. Metadata must be typed and structured so that it can move from data store to data store (IAP)
3. The basic language needs to be XML because it is open and also widely used by programming tools (all)
4. The environment that JDF is developed in must be a public and open environment (all)

In fact, PJTF and PPF are "mapped" into JDF — In the JDF document you will find appendices that provide explicit instructions for moving from PJTF or PPF to JDF.

Once the "Gang of Four" transferred the specification to the new CIP4 organization in 2001, JDF 1.0 was published. Everyone agrees that it was not possible to implement JDF 1.0, rather it served as a "straw man" document that members of CIP4 could shape, change, and improve … it was a starting point. JDF 1.1 and 1.1a were published in April and October of 2002. Significant changes were made in the specification and accompanying schema and most of the equipment that you see on the market today is build to JDF 1.1 and 1.1a. JDF 1.2 is just being released for public review and is expected to be formally published in at drupa 2004 on 9 May 2004. JDF 1.2 is constitutes the maturation of the JDF specification and includes several important new features that are the subject of this paper.

Overview of Changes

JDF 1.2 has over 650 specifically identified additions, deletion, deprecations, clarifications, and modifications plus at least six global changes. These changes include:

1. Minor changes that include editorial changes for clarity, organization, consistency of the specification,
2. Moderate changes consisting mostly of improvements identified due to lessons learned by developers,
3. Significant changes that include reorganizations of JDF or JMF elements and attributes, and
4. New functions.

Minor Changes

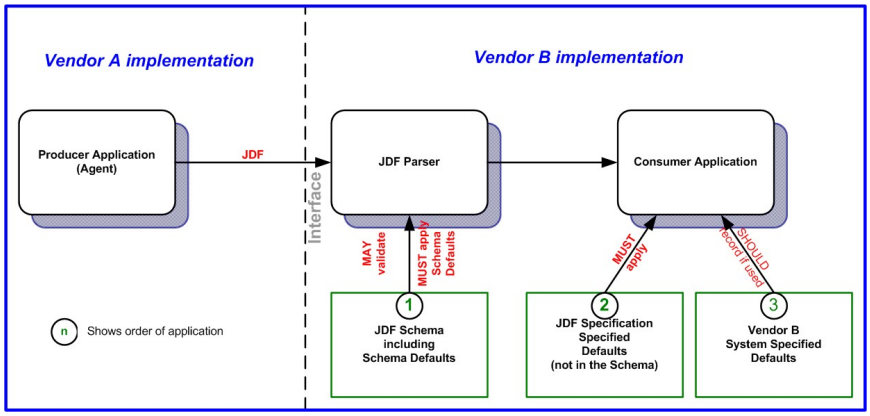
Most minor changes are editorial in nature and are not substantive, but rather provide clarifications. For instance, In JDF 1.1a several different JDF element definitions discussed operations that affect the "page"; however, the use "page" was inconsistent. Did "page" mean numbered pages as in the layout file, did it include blank pages as in

the imposition, was "sheet" a "page", and so on. In JDF 1.2 definitions such as "finished page," "reader page," "folios," "leaf," "sheet," were refined or added and are now consistently used throughout the document. Another example is using modified XPath Notation to indicate location of elements and attributes in the schema (ex. **Element**/Element/@*Attribute*, where the application of bolding, font and italics to terms is unique to JDF and indicates element types of certain categories as defined in the JDF specification such as **Resoures**, ***Process***, Elements, *Attributes*, and *"Attribute_Values"*.) For example:

> **Surface**/MarkObject/DynamicField/@*Format*=*"Replacement Text for %s and %s go in here at %s on %s"* Means the value *"Replacement Text for %s and %s go in here at %s on %"* of the *Format* attribute of the DynamicField subelement of the arkObject element of the **Surface** resource element.

Some other minor changes that were not substantive include the removal of deprecated process, resources, and JMF messaging elements to an appendix to imporove readability of the specification, and the movement of most references to a references appendix due to a dramatic increase in the number of references specifications, standards, and other technical references.

Minor changes that are substantive include the removal of nearly all "Systems_Specified" enumeration and NMTOKEN default values. In JDF 1.2, many attribute values defaulted to "Systems Specified," but this meant that in practice a JDF job could not be sent to a device until the values of attributes that were not specified by the user or operator were defined at "System Specified." This created the need for extra and unnecessary programming, and where JDF "templates" are used that allowed users to only fill in information that is unique to a job the use of "system specified" introduced potential errors.



Handling of Default Values of JDF Attributes

Source: JDF 1.2 Specification

In JDF 1.2, a sending device only needs to send the JDF data it needed to handle its processes or functions in the workflow as validated JDF to a consuming device. Only those defaults specified in the schema that are required by the consuming system are applied by the receiving validator prior to being imported into the consuming system. Since there are no "system specified" defaults, the consuming system is now responsible to adding any default values it requires; including JDF-specific "if-then" defaults where in the specification. The "if-then" defaults, which are not validated by an "off-the-shelf" JDF schema validator, are conditions in the specification where default is not set if there is a value somewhere else in the, but if that value is not set, then there is another default for the current attribute value. For instance, if an output ICC profile is set at both the document (layout) level as well as at an image level, then the document output profile takes precedence.

Another substantive yet minor change was the removal of "Number" data types including "Number", "NumberList", "NumberRange", and "NumberRangeList". Number allowed both Integer and Double data types to be used. In practice this introduced an ambiguity that posed a problem for interchange between systems. Since integer and double data types were also defined in the specification, number was eliminated and only integer and double data types are now allowed.

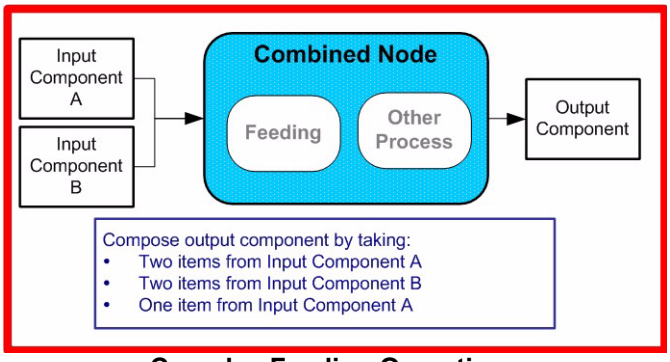
There were hundreds of clarifications made to the definitions of attributes. Most of these were simple done to eliminate ambiguities and provide clarification for programmers and developers. There are too many to cite in this article, but in general, these changes were part of the maturation of the JDF specification in that they resulted from experiences in using JDF-enabled systems and testing to implementing inter-supplier data exchange. In the process many misinterpretations of the specification were found and the various working groups who are responsible for the development of JDF clarified the problematic definition to prevent future misinterpretations.

Moderate Changes

Moderate changes are mostly changes to element definitions that were the result of lessons learned in the field an in testing. Many of these changes included the addition of element and attributes that were found to be missing. For instance Instant Messaging and SMS (cell phone messaging) were added as channel options for customer messaging. Again there are far too many to cover in this article, but some other examples of such changes include:

- Adding reprint options to "Amount Handling"
- Supporting imposition of multiple customer jobs on a single sheet
- Adding Bundling, PrintRolling, Feeding, process for Postpress operations
- Adding Bundling, PrintRolling, Feeding, process for postpress operations
- Adding several new data types

For instance, in the case of the addition of the Feeding process, it was assumed that the gathering process was sufficient in JDF 1.1a. However, that assumption was that sheet or signature gathering was always based on one of each sheet or signature produces. In practice it was found that some gathering operations were not singular (e.g., 1, 1, 1…), but could include combinations of single and multiple signatures or sheets (ex. 1, 2, 1, 3). So in the graphic below, to input component sources are feed to a Gathering or other postpress process where two items are pulled from the component A source, then to items are pulled from the component B source, and then one item is pulled from the component A source.



Complex Feeding Operations

Source: JDF 1.2 Specification

In another example, it was found in implementing JDF 1.1a systems that there were some functions that could not be implemented, or implemented only with great difficult with the data types provided in JDF 1.1a.; hence, several new data types were added. For instance specifying an acceptable trim range or bleed area range had to be completed by defining two rectangles and then comparing the. In JDF 1.2 this function was made easier by adding a RectangeRange data type and XYRelation data types. The new data types include:

- DateTimeRange
- DataTimeRangeList
- DoubleList
- DoubleRange
- DoubleRangeList
- DurationRangeList
- PDFPath
- RectangleRange
- RectangleRangeList
- XPath
- XYRelation

Significant Changes

Significant changes are those changes that do not add new features or functions to JDF, but which include significant changes. Please note that one significant change is how JDF version are handled. A JDF instance may indicate the JDF version it is based upon and JDF-enable systems must be backwards compatible. This is why the definition of deprecated items has not been removed from the specification, but rather they have been removed to an appendix.

Quality control was inadequately defined in JDF 1.1a. For the most part this is due to the fact the many quality control systems and databases are user defined or built and the developers of JDF-enabled systems did not fully understand the printer's quality control needs until they gained experience with early implementations. A QualityControl general process was added to JDF 1.2 with two associated (and new) JDF resources: QualityControlParams and QualityControlResult. The QualityControl process defines the setup and frequency of quality controls for another JDF process. QualityControl is generally performed on Components (i.e., sheets, plates, etc.) produced as intermediate or final output of a JDF process. The QualityControlParams include setup items like duration, frequency of sampling and so on. The QualityControlResult specifies the overall results (e.g., pass or fail) as well as the specific values of measurements taken and the file that they are contained in.

Event though JDF 1.1.a including specifications for file exchange methods, there was no way established to communicate how to exchange JDF instances or content files between a sender and receiver. The new DigitalDelivery process was added to answer this need. It includes parameters such as direction (push or pull), protocol (HTTP, HTTPS, FTP, or SMTP), and method (email, webserver, etc.)

Color management in JDF 1.1a was rather rudimentary form the perspective of color experts. In JDF 1.2 supported color correction and colorspace conversion methods were greatly expanded and detailed. ColorMeasurementConditions, PrintCondition, and an appendix describing how to use the color adjustment attribute were added and significant modifications were maded to Color, ColorCorrectionParams, ColorSpaceConversionParams, ColorantControl, ColorantAlias, and SeparationSpec.

It was realized that proofing is a combination of several different functions, including imposition, separation, rendering, imaging, color conversion, and so on. As single JDF process it was terrible complex and unruly. In JDF 1.2 proofing is no longer a process, but rather a combined process like conventional and digital printing in that it is actually the closely linked application of several specific processes. This change helped make JDF 1.2 systems easier to build and simplifies the automation of print jobs.

In the case of the Job Messaging Format (JMF), many changes have been made, (again too many to cover completely in this article), but these can be covered in three categories: Added additional MIS ↔ Device messages, Added messages for requesting new resources, and Clarified JDF/JMF transport and packaging.

Several new messages and message attributes have been added to improve communication between the workflow or MIS system that governs the production of JDF jobs and the devices that perform the requested processes. For instance, the new QueueEntryID provides a number by which the MIS system can request a change in Queue order or make status inquires of specific jobs in a queue. JobID and JobPartID relate messages to devices to their Jobs. NewJDF allows the MIS system to submit a late change to a JDF Job even while it is the device queue. Shutdown and WakeUp add addition direct controls of the MIS over each device on the shopfloor.

Recognizing that sometimes work has to be re-worked due to bad materials (ex. cracked plate) or other conditions, JDF 1.2 now provides FlushResources and PullResource that to clean out the inputs of job and rerun it. So for instance, if the quality control feedback to a CTP device reports that the last plate was cracked, the CTP device could flush out the "plate" in its memory and request a new plate via "PullResources" to associate to the job and rerun the job.

Considerable work has been done to be more specific in JDF 1.2 on how MIME messaging and MIME packaging of JDF and content files is done. Furthermore, where JDF 1.1a allowed Acknowledgements, JDF 1.2 clarifies the use of acknowledgements by providing a new ResponseURL and AcknowledgeURL that may be different from the sending URL and providing for hot folder use, AcknowledgeFormat. (Note: a response is a status like "completed" from a device where an Acknowledgement is a reply to the sending system acknowledging receipt of message before it is acted upon.)

The preflighting processes and resources in JDF 1.1a were only temporary and there were cautions that they would be fully defined in JDF 1.2. Good thing too, as the resources used by Preflight (e.g., PreflightAnalysis, PreflightProfile, and PreflightInventory) were completely depreciated. Prelighting now the new PreflightParams, PreflightReport, and PreflightReportRulePool resources. A preflight utility may export a JDF PrefiltReport that is structure according to the rules established in PreflightReportRule Pool. A JDF-enabled system may also use a preflighting tool or "engine" to structure specific preflight checks of files. This is enabled by the new JDF Device Capabilities features. Device Capabilities is used to structure tests with Boolean logic and various evaluation tests. A "TestPool" contains the various tests and an ActionPool schedules the execution of those tests. Frankly, device capabilities is extremely complex and is only of use to programmers and developers and may never be directly

Finally, FileSpec is a resource that is used to describe a file and what may be done with that file. FileSpec is used in digital asset exchange with the customer or for internal exchange of files between systems and devices. Many improvements have been made including:

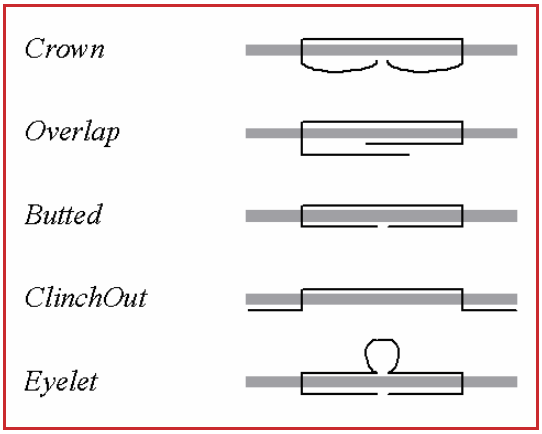
- Several modifications that allow for more precision in identifying the file's source operating system (OS), OS version, and MIME type used.
- A Container element that allows files to be contained in .ZIP, MIME Multipart/Related, or other compression formats.
- A Disposition element (previously a single attribute) that informs the device what to do with the file after it is done and allows for actions such as scheduling a duration for the archiving of a file.
- A UID which is, for instance, used to reference the ID field of the PDF file's trailer.
- OverwritePolicy which can be used to create a new name or version of a file after it has been processed.

New Functions

JDF 1.2 also introduces the concept of "Interoperability Conformance Specifications" (ICS). No single device (i.e., printer, press, imagesetter, etc.) is likely to implement all that the JDF specification provides for. For instance, if you are in the digital printing business, you may not care to facilitate data on hard case binding. A RIP need not facilitate JDF preflighting. A Stitcher probably doesn't need to handle image rendering data. To specify exactly what individual classes of devices need to do with JDF, CIP4 members are developing ICS that will provide the standard for individual classes of devices. ICS document will later (probably beginning in late 2004 or early 2005) be used as the basis for certification testing. CIP4 has signed on GATF as the first certification testing facility and others will later be added in Europe and Asia. Once the certification program begins, you will start seeing products that are marked as "JDF Certified" and this will be to a specific ICS document. The ICS documents are all currently in draft form and only in circulation among members of CIP4, but once published, like the standard, they will be freely available to the public and we expect that they will become part of your buying practices.

As mentioned above, JDF 1.2 device capabilities (defined in the DeviceCap resource) are used to support preflighting functions in JDF 1.2 enabled systems. Device capabilities can be used to collect information about all devices available on the floor or known to a MIS or workflow system so that it can compare a job's specifications and content files to "look forward" and pick the best path for the job through production. In the future this may be applied to picking a path through several plants or even networks of printers.

There may be a fourth primary function of JDF, which is new to JDF 1.2 and that is automating the handshake, which is accomplished with the new device capabilities functionality. For instance, in JDF a there are five staple folds that a stitcher may use. If a new stitcher is added to your JDF workflow, the governing workflow or MIS system must know which of those five folds the new stitcher sup-ports. Communicating the set of JDF elements and attributes supported by a device to the MIS system or workflow system is creating the "handshake."



Five Staple Folds in JDF 1.2
Source: JDF 1.2

Currently, printers must make this reconciliation or "handshake" for themselves or with the help of their vendors and/or consultants. Some groups, such as NGP or Print City, are constructing the handshakes between devices of participating companies so that come drupa, if you buy from a the companies in one of these groups, you'll have some assurance that the handshakes have been established and the devices among the partners have been pre-integrated.

Device capabilities in JDF 1.2 allows for 1.2 capable devices to be automatically queried for the details of what aspects of JDF they can and cannot manage. This is an important step towards total "plug-n-play" interoperability, but it may be a year or more before there are enough JDF 1.2 products on the market for buyers to specify and rely on this automated handshake functionality.

Finally, in addition to automating the handshake, device capabilities allow a system to query a device in order to establish how device features can be presented in a graphic user interface. This is not the design aspect of the interface, but rather the fields of information. This is similar to how Print Manager on a desktop computer running a Microsoft operating system displays the job set-up instructions for different printers. Each printer has a unique set of

set-up and print options. Like wise device capabilities can be used by workflow systems to discover the unique set-up, production options and commands for a specific device so that it can generate graphic user interface window that presents those unique set-up, production options and commands to the user.

Conclusion

By establishing the basic language for metadata interchange, JDF has already delivered a great benefit to the industry, as it makes open integration and data exchange between systems possible. This is perhaps 70% of the total effort and many changes in JDF involve modifications and clarifications to this basic language as a result of interoperability testing and real world implementations. JDF 1.2's major contribution is the addition of device capabilities which perhaps is 10% of the total effort. Device capabilities will go a long way to making integration much easier, but it will probably take 18 months to two years for there to be enough JDF 1.2-enabled devices in the market to make an impact on the printing community.

JDF is not plug-and-play, but that is the goal. In the future we will see expansion of JDF to include advertising insert data exchange, distribution metadata, and niche metadata for areas of printing such as packaging. In my estimation this will account for another 10% of the overall effort with the final 10% being the effort to get to true plug-and-play functionality. This may include an XML package for JDF interchanges that specify the content, the transaction type, the nature of the request, and the expected response, acknowledgement and results.

However, JDF 1.2 should not be marginalized. There will be 150+ JDF-enabled products and services on the market by drupa 2004 and I estimate that number will grow to 300-400 by Print '05. There are a few dozen completed customer implementations in the market and the reported realized benefits are impressive (but that would be the subject of another paper!). By the end of 2005 the number of users may be 1,000 or more. JDF is maturing and definitely coming of age.

References

JDF Overview, by James E. Harvey, JDF Expert Certification Program, International Prepress Association, 2004.

JDF Specification 1.2, James E. Harvey Editor, "Copyright © 2000-2004, International Cooperation for Integration of Processes in Prepress, Press and Postpress, hereinafter referred to as CIP4. Reproduced with permission of CIP4 (www.cip4.org). All Rights Reserved. (Note: From pre-published draft at the time of writing this article.)

JDF 1.2 Enhancements, by Dr. Rainer Prosi, Heidelberger Druckmaschinen AG, unpublished PowerPoint presentation, 2004.