PDFとメタデータのリアルタイム同期

PDF and Metadata Real Time Synchronization

マーカス ワーラー[*] Marquis WALLER
マイクル ローツ[*] Michael LOTZ
ジェフリー シッキンク[*] Jeffrey SIKKINK

要　旨

PDF (Portable Document Format) は交換可能なドキュメントレイアウトとして設計された．PDFの使いやすさ，セキュリティ機能，さまざまな環境で作成できる特徴は，PDFの需要を高め，PDFはさまざまな産業分野で活用されている．印刷業界はこの10年にわたりPDFの利用により大きく発展してきた．PDFが印刷される場合にPDFに欠けているのは，PDFファイルをどのように印刷するかを定義する能力である．

JDF (Job Definition Format) が，PDFをどう印刷するかの指示を定義する業界標準となっている．JDFは独立したファイルであるため，印刷ワークフローでPDFファイルが変更された場合にPDFファイルとの同期がとれなくなり得る．本論文では，PDFファイルに対するアクション（ページ移動，ページ削除，ページ回転等）のリストを，他のプログラムによる変更を防ぎつつJDF（およびそのほかのメタデータ）がリアルタイムにデータと同期されるように処理するプロセスを詳述する．

このプロセスは，印刷時の正しい出力を保証するとともに，PDFファイルを操作する際のファイルI/Oを改善する．

ABSTRACT

PDF, Portable Document Format, was designed to be a transferable document layout. PDF’s ease of use, security features, and ability to be created almost anywhere has helped PDF’s popularity grow and made PDF viable in a variety of industries. The printing industry has seen a major growth in the use of PDF over the last decade. A major function that PDF lacks when it comes to printing is the ability to define how the PDF file will be printed. JDF (Job Definition Format) has become an industry standard to define the instructions on how to print a PDF. JDF is still a separate file that can get out of synchronization with the PDF file as a production workflow changes the PDF file. This paper details a process created that can take a list of actions on a PDF file (move pages, remove pages, insert pages, rotate pages, etc.) that ensures the JDF (and other metadata) stays synchronized with the data in real time while preventing any updates by other programs. This process improves file I/O when manipulating PDF files as well as ensuring proper output at print time.

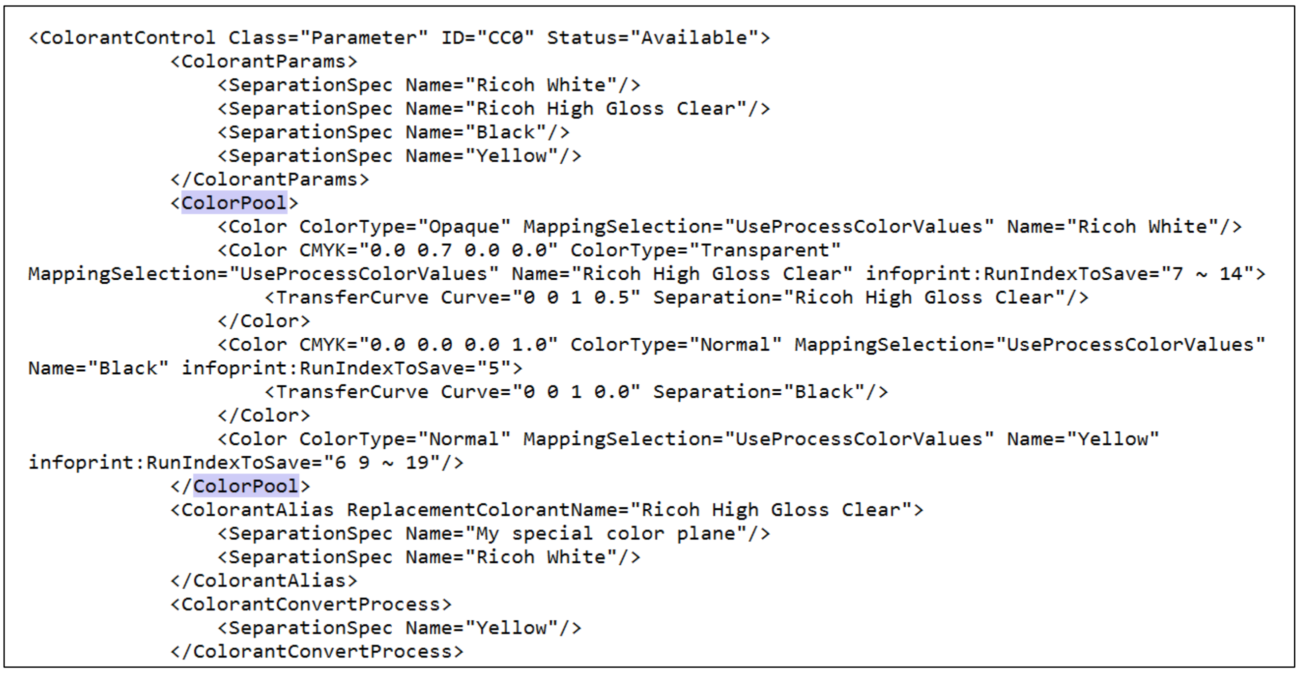
*　リコーUSA
Ricoh USA, INC.

1. Introduction

Ideally a data stream would contain all of the information about how it is to be produced inside of the data stream. However, PDF, Portable Document Format was designed to be a transferable document layout, not a data stream that was meant to be printed or physically produced in a production environment. The printing industry has seen a major growth in the use of PDF over the last decade. A major function that PDF lacks when it comes to printing and physical production is the ability to define how the PDF file will be produced. PDF's ease of use, security features, and ability to be created almost anywhere has helped PDF's popularity grow and made PDF viable in a variety of industries, but when it comes to printing a PDF file there are many aspects to the output other than the images, barcodes and text in the document[1,2].

The data stream needs to define many characteristics about itself to ensure the physical output is produced correctly. For example, what paper the output should be printed on, if the output should be stapled, punched, should specific pages be handled differently, etc. All of these different variations need to be communicated to the printer and other upstream processes. The PDF data stream itself does not contain a structure to allow these characteristics to be defined in the PDF. When working with the PDF data stream the only way to do this is with metadata defining what the output should look like. JDF (Job Definition Format) has become an industry standard to define the instructions on how to physically produce a PDF. JDF allows for a large number of characteristics to be defined on how the PDF data stream is to physically be created. The XML structure of JDF allows it to be extendible to handle specific device requirements as well as providing a standard for more common needs to physically produce a PDF file.

Fig. 1 is a small example of a JDF structure defining the Color Type to be used for specific pages of a PDF file.



```
<ColorantControl Class="Parameter" ID="CC0" Status="Available">
            <ColorantParams>
                <SeparationSpec Name="Ricoh White"/>
                <SeparationSpec Name="Ricoh High Gloss Clear"/>
                <SeparationSpec Name="Black"/>
                <SeparationSpec Name="Yellow"/>
            </ColorantParams>
            <ColorPool>
                <Color ColorType="Opaque" MappingSelection="UseProcessColorValues" Name="Ricoh White"/>
                <Color CMYK="0.0 0.7 0.0 0.0" ColorType="Transparent"
MappingSelection="UseProcessColorValues" Name="Ricoh High Gloss Clear" infoprint:RunIndexToSave="7 ~ 14">
                    <TransferCurve Curve="0 0 1 0.5" Separation="Ricoh High Gloss Clear"/>
                </Color>
                <Color CMYK="0.0 0.0 0.0 1.0" ColorType="Normal" MappingSelection="UseProcessColorValues"
Name="Black" infoprint:RunIndexToSave="5">
                    <TransferCurve Curve="0 0 1 0.0" Separation="Black"/>
                </Color>
                <Color ColorType="Normal" MappingSelection="UseProcessColorValues" Name="Yellow"
infoprint:RunIndexToSave="6 9 ~ 19"/>
            </ColorPool>
            <ColorantAlias ReplacementColorantName="Ricoh High Gloss Clear">
                <SeparationSpec Name="My special color plane"/>
                <SeparationSpec Name="Ricoh White"/>
            </ColorantAlias>
            <ColorantConvertProcess>
                <SeparationSpec Name="Yellow"/>
            </ColorantConvertProcess>
```

Fig. 1 Example of JDF file.

JDF allows for the communication of “how” the PDF needs to be created to be exchanged between software and devices. JDF has provided continued growth of PDF as a print and physical output data stream however; JDF is still a separate file that can get out of synchronization with the PDF file as production workflows, reprints or other process change the PDF file. From Fig. 1, if the PDF file would have page 8 removed or if only part of the PDF file needed to be produced, the JDF file is no longer correct. The JDF would no longer represent how the PDF file should be produced because the JDF is not in sync with the PDF data stream being sent to the output device. The PDF and JDF data streams are now required to stay in sync if the correct physical output is going to be produced. The need for the PDF and JDF to stay in sync is critical to ensure the physical output is correct. If the PDF and JDF are not kept in sync, a large amount of waste in both physical output and time can be incurred by users. When the JDF does not represent the PDF correctly, physical output is produced incorrectly, causing waste of paper, ink, missed service level agreements and shop floor time. Users need to be ensured through a workflow when the PDF is modified the metadata that represents its output is as well. When this is in place the physical production of PDF is streamlined, efficient, and accurate.

2. PDF and Printing a perfect Storm

2-1 How PDF and Printing don’t mix

Ideally PDF would contain all of the information it needs for physical output inside of the data steam, but PDF does not. PDF was designed for viewing, not printing and, to its credit is device-independent when rendered on a display. JDF and other metadata files are important to ensure the physical output of the PDF is correct and to allow systems to define a wide range of finishing options, but the nature of a PDF file destined to print is rarely static. As a PDF file is processed in a printing workflow, many aspects of the PDF can change. The PDF file may need to have static pages inserted for electronic forms. The PDF file may have to have its page order reversed for a continuous forms printer. Some pages may need to be removed if they contain incorrect data or need to be pulled from printing for a variety of reasons.

When actions like these occur, it affects the end results of the physical output, which means another process has to update the JDF and other metadata related to the PDF file. Performing these updates in separate processes can lead to numerous problems when it comes to printing.

A simple example: A PDF file has a JDF file defining page exceptions for the paper the file should print on, such as pages 1-2 are on 8X11, pages 3 is on A4, pages 4 is 8X11 and so on. If a PDF file has a constant back page added after each page to allow the file to print duplex, but the JDF is not adjusted, then over ½ of the job will be printed on the wrong paper as the page exceptions would reference the wrong data pages (See Fig. 2.)

This leads to a significant cost in time, quality, and supplies for clients. When producing output for printing, as a PDF file flows through a workflow, the PDF and metadata files need to stay synchronized. This should occur in real time each time the PDF file is touched to ensure all of the steps executed in a production workflow are working with the correct data. Not performing the synchronization in real time may result in the workflow working with the wrong information and ultimately producing the wrong physical output. In addition to ensuring that the printed output matches the original intent of the PDF author, other metadata that is used by the workflow steps for such things as conditional processing (If Member level = Gold, print a high-value coupon, if Silver level, print lower value coupon) must be preserved when page number or order changes. Moreover, the ability to update information about the processing of the PDF to indicate when reprints were done or an invalid address was used ensures that any reporting done on the production of the PDF file is accurate.

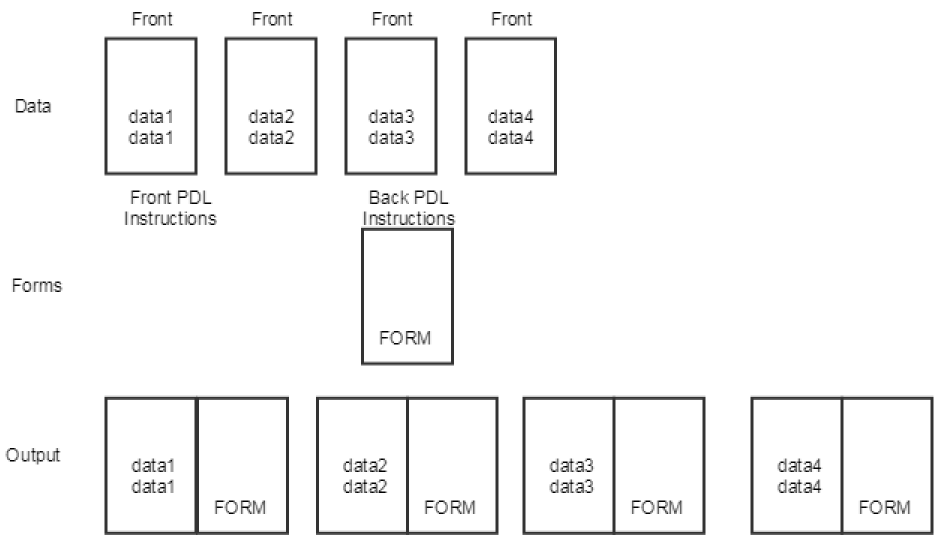


Fig. 2 Example of PDF file having pages inserted for constant backs to allow for duplex printing.

3. Action list to the rescue

3-1 Conventional method to solve the problem

In production environments a data stream can be produced in a variety of ways. PDF was designed for viewing, not printing. In producing output very rarely are two devices alike. Different printers produce the same color differently. One cutter may need a different offset than another. These are just examples of the complexity that needs to be represented to produce the physical output of a PDF file. To try to force fit these configurations into the PDF data stream either through annotations, comments or other means minimizes or reduces the usefulness of the PDF data stream at its core. The PDF file may be printed, but the PDF is also used in other ways such as email or mobile and it is important to keep the integrity of the PDF file intact while being able to represent the way the PDF needs to be produced for physical output. This is where JDF and other metadata files help solve the problem. JDF allows for complex configuration of the output to be defined without impacting the integrity of the PDF data stream. Also JDF provides flexibility in sending the PDF to different devices that may require different settings.

3-2 Why uniformity between PDF and JDF is so important

There is an obvious difference in “waste” when it comes to physically producing a document vs. a digital representation of a document. When producing a physical output many items have to be correct. What type of paper the output is produced on, what ink is used to ensure the proper colors are created, which corners of the document are stapled, etc. If one if these is wrong, even if for only one page, the entire document is thrown away many times and is required to be reproduced. This is a considerable cost to a user in material, time, and environmental impact. When a PDF document is modified the JDF and other metadata representing it must be updated to eliminate this cost. This is where the problem comes in when having the JDF separate from the PDF data stream. Yes the PDF integrity is kept, but the possibility of incorrect physical output is raised. As stated before, the solution is not to insert the “how” to produce the PDF into the PDF document physically, but to use the action list to keep both the PDF document and JDF representation in sync in real time to ensure correct output. Action list keeping the PDF and JDF and other metadata in sync allow for the PDF to be kept pure while ensuring the metadata that is needed to print the PDF correctly is updated for any modifications needed.

3-3 What about other metadata associated with the PDF

Other forms of metadata also allow for different actions to be taken on the PDF based on properties that are not contained inside of the PDF. This provides the ability for production workflows to modify the PDF properly based on outside data. This data could include preferences from a user, requirements on image size or other aspects that affect the output. This metadata is only useful when kept in sync with the PDF document. When the PDF document is modified and the metadata is not updated, it can cause actions farther along in the workflow to be done incorrectly. This hinders the usefulness of automated workflows as manual checks are needed to ensure both the PDF and metadata are correct. Action list allows for keeping this data in sync in real time and avoids the need to add manual checks and allows the workflow to process on the proper data without worry of synchronization issue.

3-4 How action list solve the problem in a better way

Action list reads the PDF file and all associated metadata files that represent the PDF. These are represented as Java objects.

Action lists are built on an extendible architecture. This allows for various developers to create their own filters to take the proper actions needed on a PDF and metadata file. These filters can then be chained together to allow a variety of modifications to occur on a PDF and metadata file. Because of the chain each filter uses the rendered output from the previous filter both in PDF and metadata. This is an important aspect of the design as it allows for many actions to be taken and not require reading and writing of the PDF and metadata over and over again, improving File I/O for the system.

Each filter defines the parameters it needs passed to a function. This gives each filter the flexibility in the additional data it uses to perform the filter's operation.

Filters are than required to have a process method. This method receives what is called a PDFWorkUnit. This PDFWorkUnit contains the representation of the PDF file, JDF file and other metadata files identified for the PDF. This process method performs the needed actions on the PDF, removing pages, rotation images, adding stamps, etc. At the same time the filter modifies the JDF and metadata as needed. When complete the filter "publishes" the PDFWorkUnit to the next filter. When each filter is complete, a close method is called. This allows for each filter to examine the "final" PDFWorkUnit and make any adjustments that filter may need for "final" output. The OutputFilter then writes all files associated with the PDFWorkUnit to disk as defined.

Each action list contains the following.

(1) InputFilter

(2) Set of action filters

(3) Output Filter

(See Fig. 3.)

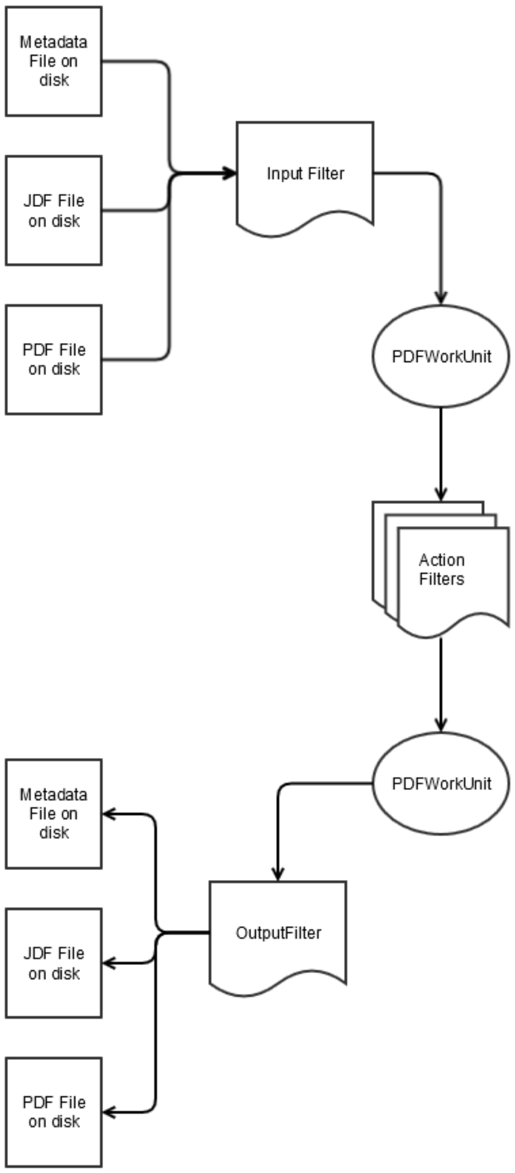
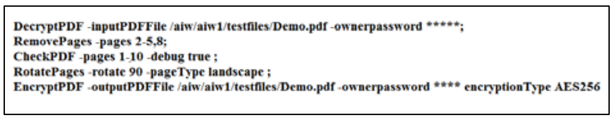


Fig. 3　Flow of filter.

These three types of filters are important in dealing with a large range of PDF data. One example is Encrypted PDF documents. Encrypted PDF documents cannot be modified and in many cases not read in by programming languages. To allow the filter to modify the PDF and metadata and write it out securely again, an InputFilter to decrypt the PDF can be created. This allows the action chain to Decrypt the PDF, take actions on the PDF and metadata files. Then an OutputFilter to reencrypt the PDF can be written to ensure when the PDF document is written to disk and the PDF document is encrypted again.

This is another important aspect of performing the operations in memory. It ensures the PDF file is never on disk in a decrypted mode.

Here is an example of what a filter chain may look like when being called.



```
DecryptPDF -inputPDFFile /aiw/aiw1/testfiles/Demo.pdf -ownerpassword *****;
RemovePages -pages 2-5,8;
CheckPDF -pages 1-10 -debug true ;
RotatePages -rotate 90 -pageType landscape ;
EncryptPDF -outputPDFFile /aiw/aiw1/testfiles/Demo.pdf -ownerpassword **** encryptionType AES256
```

Fig. 4　List of actions.

The design of the filter allows for simple additions of new filters and the ability to expose the filters to the end user without additional work.

Each filter includes the following methods

- additionalFilterInformation()

  This allows for the filter writer to provide additional text about the filter to the end user. This is automatically displayed when looking at available filters.
- process(PDFWorkUnit)

  Where the work is done on the PDF and metadata. The filter designer can take various actions on the files defined in the PDFWorkUnit and pass the updates to the next filter.
- provideParameters()

  Allows for the filter writer to define a set of parameters and the properties of those parameters. This information is converted to text automatically and displayed to the end user when a list of filters and parameters is requested.
- setFilterToClassMap(Map<String, String>)

  Allows for a simple name mapping to allow the filter to be called.

With this design as new filters are added to a system, what they do and what they require are easily exposed to the end user without the need for separate documentation or other sources.

4. Results of new method

A key technology interface was developed to allow actions done to a PDF file to be chained together. This interface takes a PDF file as input to be modified as well as its JDF job ticket. It can also receive other metadata files associated with the PDF such as CSV file containing additional information about specific pages or sets of pages in the file (See Fig. 5.)

As the PDF file goes through the chain of actions, the PDF file is kept in memory. This is important as the PDF file is never written back to disk until all of the actions are completed. This improves performance and also ensures a partially modified PDF file cannot be accessed by other processes. To handle larger PDF files a data model representing the PDF itself was created. This data model allows for the locking of the actual PDF file, and loading pointers of the pages in memory without having to load all of the contents of the PDF in memory as well.

The locking of the file is automatically done on the files the action list is processing. This is a file lock that prevents any reading or writing of the file and allows the action list to perform all operations in memory. This adds to look up time but allows the larger PDF files to be handled (See Fig. 6.)

As each action is performed on the PDF file, the same actions are updating the JDF file, and other supported metadata files in memory as well. All updates to the PDF file and metadata are done in memory and done at the same point. A single action in the list does not complete until both the PDF and metadata files are updated in memory. When an action completes, the action passes the modified PDF and metadata files in memory to the next action in the chain (See Fig. 7.) By not writing to disk between actions, synchronization of the PDF and metadata files is ensured for the full chain of actions. Synchronization is ensured as the PDF File and meta data files are being operated on at the same time and always in

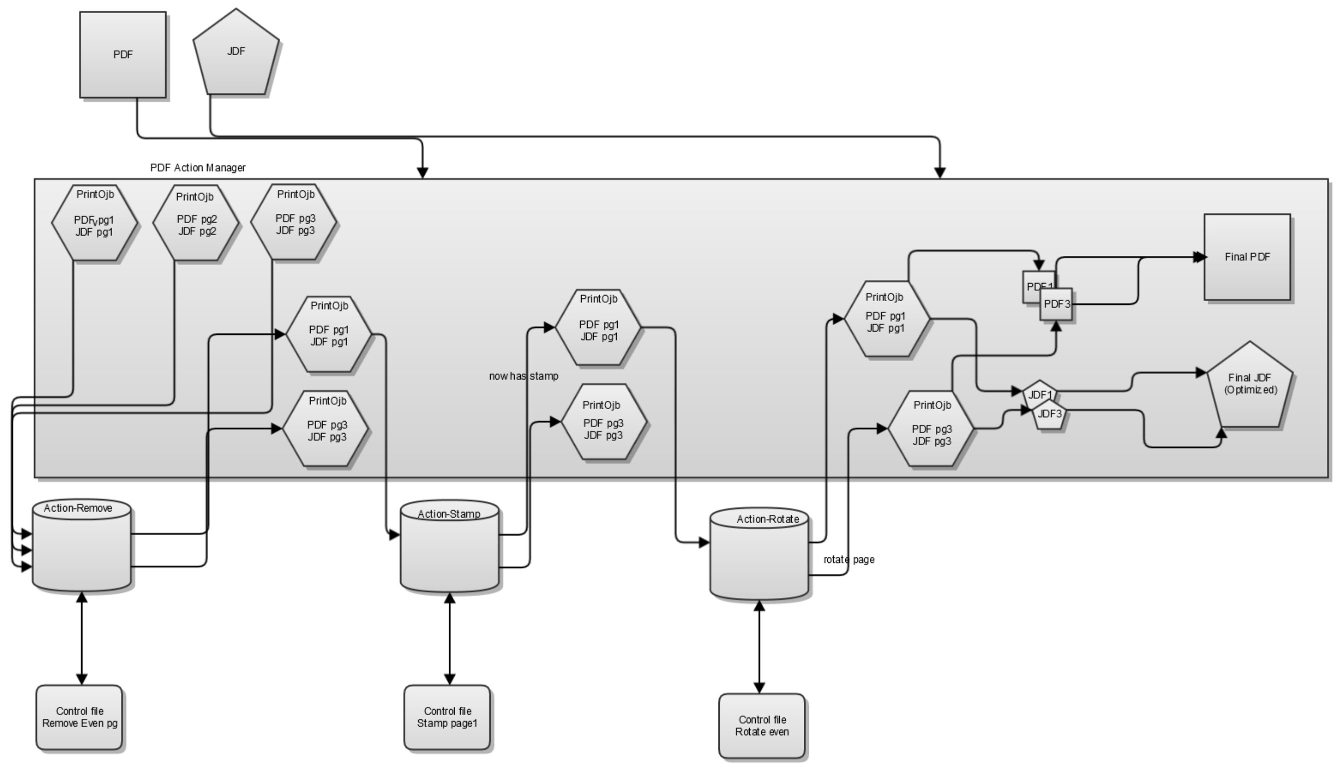


Fig. 5 Flow of action list with PDF file.

memory. Because no other access to the files is allowed and all meta data must be updated for each filter to proceed this action list process ensure PDF and meta data are in sync. This also prevents other outside process from modifying or even accessing the PDF and metadata files during this processing.

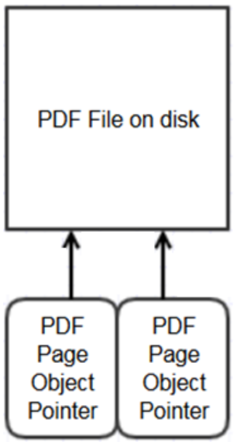


Fig. 6 PDF pages of a PDF File in memory.

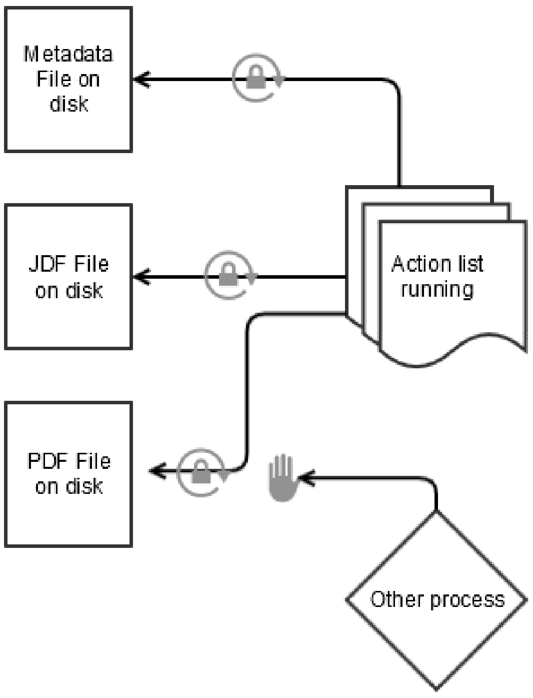


Fig. 7 PDF and JDF Files locked by action list process.

This is important because the next step in the workflow can start processing the PDF and metadata files without having to verify that they match. Before this method multiple checks would have to be performed throughout the workflow to ensure that the PDF and metadata files were kept in sync. Additions or modification to the workflow become hard to do and require a significant amount of verification because of the possibility the PDF and metadata are no longer matching. Chaining actions together allows a workflow to make several modifications to a PDF file and its metadata with low overhead and improved performance.

5. Conclusion

An innovative method has been developed to ensure the PDF data stream and the metadata files that represent the PDF and how it is to be produced are kept in sync with each other in real time. The technology described has been implemented in high-speed production workflow software. Positive feedback has been received from users using these solutions. The solution opens up many opportunities for users to bring in many different kinds of new production work without the concern or fear of how to keep the PDF and metadata in sync and being able to deliver on the committed Service Level Agreements.

References

1) Enfocus: Enfocus Pitstop action manual, http://www.enfocus.com/manuals/Extra/Actions/12/pdf/Actions.pdf (accessed 2017-09-25).
2) Adobe: Acrobat Pro action wizard, Acrobat user guide, https://helpx.adobe.com/acrobat/using/action-wizardacrobat-pro.html (accessed 2017-09-25).