Application Notes for CGATS.20 (PPML/VDX)

Version 1 — August 2004

Prepared by:

CGATS

The Committee for Graphic Arts Technologies Standards

NPES The Association for Suppliers of Printing, Publishing and Converting Technologies
1899 Preston White Drive
Reston, VA 20191-4367
USA
Tel: 703-264-7200
Fax: 703-620-0994
E-mail: standards@npes.org

Contents

Page

Application Notes for CGATS.20 (PPML/VDX)

1 Introduction

These application notes discuss topics that aid implementers of PPML/VDX workflow tools and demonstrate the various design features of the PPML/VDX file format. Topics also include preflighting PPML/VDX instances. Annex A details the steps of converting a conforming PPML/VDX instance to a scheme more oriented towards a PPML type of consumer interface.

This document is intended to supplement CGATS.20-2002, *Graphic Technology – Variable printing data exchange using PPML and PDF (PPML/VDX)*. It is assumed that a reader of this application note is familiar with the PPML/VDX standard, as well as the PDF, PDF/X, PPML, and JDF data formats.

To anyone who acknowledges that these application notes are provided "AS IS", WITH NO EXPRESSED OR IMPLIED WARRANTY: permission to use, copy and distribute them for any purpose is hereby granted without fee, provided that the contents of these notes are not altered, including the NPES copyright notice tag. Neither NPES nor CGATS makes any claims or representations about the completeness of these application notes.

These application notes are subject to revision and enhancement. The most current version of this document can be found in the NPES standards workroom at <http://www.npes.org/standards/tools.html>.

Some earlier revisions may be maintained at the same site. Comments and suggestions should be sent to the CGATS secretariat at standards@npes.org.

1.1 References

Adobe Portable Document Format Reference Manual -Version 1.3, Dated 11-March-1999, Adobe Systems Incorporated

ANSI CGATS.20-2002, *Graphic Technology – Variable printing data exchange using PPML and PDF (PPML/VDX)*, American National Standards Institute, available from NPES The Association for Suppliers of Printing, Publishing and Converting Technologies, http://www.npes.org/standards/workroom.html

JDF Specification – Release 1.1 Revision A, http://www.cip4.org/ – September 2002

PPML Functional Specification – Version 2.0, Dated 27-March-2002, Print On Demand Initiative (PODi)

1.2 Symbols, notations, and abbreviated terms

PDF operators, PDF keywords, the names of keys in PDF dictionaries, and other predefined names are written in a bold sans serif type font.

Operands of PDF operators or values of dictionary keys are written in an italic sans serif font.

Names of PPML and JDF elements are written in a bold sans serif font.

Attribute names of PPML and JDF elements are written in a bold italic sans serif font.

Values of attributes of PPML and JDF elements are written in an italic sans serif font.

The term *PDF specification* refers to the *Adobe Portable Document Format Reference Manual* (see 1.1).

The term *PPML specification* refers to the *PPML Functional Specification* (see 1.1).

The terms *PPML/VDX*, *CGATS.20*, or *PPML/VDX Standard* refers to ANSI CGATS.20-2002, *Graphic Technology – Variable printing data exchange using PPML and PDF (PPML/VDX)* (see 1.1).

The term *JDF specification* refers to the *JDF Specification* (see 1.1).

1.3 Why PPML/VDX?

ANSI CGATS.20 (PPML/VDX) defines a variable-data printing (VDP) structured document file format.

Unlike the many VDP page description language (PDL) file formats available and in use today, PPML/VDX is designed to convey graphical page content as well as product intent (a description of the finished print products to be manufactured), in a way that is both production-device independent and indifferent to any target print-production workflow. In color reproduction, production-device independence is achieved through use of intermediate reference printing conditions or reference color characterizations.

The PPML/VDX standard was developed to facilitate several different objectives:

- the effective creation of variable-data print jobs without requiring the authoring system to target the jobs to specific equipment;
- the ability to target variable-data print jobs to a variety of output devices and/or printing companies; and
- the integration of all required data to enable color management.

PPML/VDX uniquely satisfies the requirements of guaranteed portability, page independence and device independence, and thus provides a new opportunity in variable digital printing that decouples the complex activities of VDP job authoring from the methods of print production and fulfillment.

The PPML/VDX standard was developed to satisfy the requirements necessary to enable this decoupling in a robust way. It allows the VDP data originator to maintain control over the quality of the VDP content data and print-product definition throughout its distribution into, and within, print production environments. PPML/VDX provides the business and contractual footing for variable jobs that has been expected for traditional commercial print work.

PPML/VDX can be used for exchange of VDP page data between disparate authoring and production workflow environments without the need for supplemental technical communication. This is referred to as "blind exchange."

PPML/VDX can also be used as a working format within a single, loose or tightly integrated authoring and production workflow environment, much the way a significant amount of VDP is done today.

This flexibility allows implementation of advanced job-ticket-based VDP workflows, leading to new business models and practices beyond what can be achieved today within the confines of wholly integrated authoring and production workflows.

In current VDP practices, the content creator and print producer negotiate each aspect of each job and reach contractual agreements that specify use of certain output devices. In the emerging arena of commercial variable-digital printing, creators will expect the freedom to send jobs to any of a number of printers with the necessary fulfillment-verification mechanisms in place.

Using PPML/VDX and the emerging JDF job-ticket standard, the content creator will be able to divide each job among several print producers, and still be assured that output from each will exactly match the color and quality of output from every other one.

1.4 How is this vision to be achieved?

PPML/VDX and JDF standards put the power to create, control and express intent for each part of every job firmly into the hands of the content creator – the artist, ad agency, publisher or whomever.

The development of both PPML/VDX and JDF, together with ICC color management, enable creators to cross the divide between themselves and those who output their creations on paper, and generalize the creation step so that it can be reliably produced on any number of machines in any number of geographically separated plants.

Though PPML/VDX enables all these benefits for blind exchange of variable jobs, it is equally useful in all graphic arts variable-digital printing situations, even those performed entirely within a tightly integrated workflow in which job creation and production occur within the same environment. Divorcing device-control information from the original job content makes it easy to introduce new equipment into the workflow, or even upgrade from one printing press or manufacturer to another – both of which are far from easy today.

1.5 The state of the art today

Most VDP systems require the VDP data authoring and composition system to immediately target the job's digital data to a very specific production workflow and printing device. In most cases, device control information, or metadata, is commingled directly within the page description language data as a monolithic stream that can only be consumed by a raster-image processor (RIP) sequentially from beginning to end.

Such formats also require the page composition system to specify page data as a fixed, streamed sequence of pre-imposed sheet surface layouts. From an historic perspective, monolithic streaming approaches were indeed warranted given the limitations of computing platforms in which memory was expensive, data-transmission bandwidth was limited, and page buffering was to be avoided due to excessive cost.

This is no longer an issue, as high performance computing platforms – with significantly more storage capacity and data transmission bandwidth – enable VDP tools to concentrate more on workflow enablement and content management.

The center of effort for VDP job creation and production can now shift from concern for press productivity and RIP optimization toward enabling and empowering the VDP page producer. The continuing improvement in color digital printing devices and the general availability of off-the-shelf, high-performance computing and data storage platforms brings a practical opportunity for efficient production of high-quality, full-color VDP output.

These increasing capacities lead to substantial demand for new sources of high-quality VDP jobs. Sources of VDP jobs will include creative design agencies, advertising agencies and corporate marketing departments, thus moving well beyond the earlier paradigm in which print providers themselves were effectively forced to become advertising and marketing agencies in order to increase VDP page volume on their digital presses.

1.6 How does PPML/VDX differ from PPML?

PPML/VDX is based on a subset of PPML, as published by PODi, with the following additions:

- a manifest (the content-binding table) that specifies all the components needed to complete the job;
- use of JDF syntax to express the creator's intent within the PPML/VDX job specification;
- color-management through the use of PDF/X for content data.

The restrictions on PPML include:

- the removal of PPML elements that assume certain device characteristics;
- the required use of PDF/X for content (the "relaxed" conformance level does allow use of use of conventional PDF);
- disallowance of in-line content, including text or graphics.

These features and restrictions enable:

- page producers to work independently of the print provider, with no sacrifice of complexity or graphical richness;
- late-stage targeting at one or more printing devices from within the production workflow;
- a reader-order sequence of independent pages, with no implied print production ordering.

Print production information such as imposition layout, sheet marks, page distribution scheme, and printing device parameterization is internal to the print provider, and is usually specified in job-ticket data outside of the PPML/VDX data. In the context of the print-production process, the PPML/VDX page data is thought of as the content resource to the print job.

Alternatively, PPML/VDX can also be used in production workflows in which the VDP page composition system also generates production parameters encoded internally as a job ticket. In this more integrated approach, it becomes the responsibility of a workflow's intelligent composition system to provide some or all of the job ticket's process parameterization.

Despite these differences, a PPML/VDX job can be easily transformed into a form that can be consumed by any PPML-compliant device capable of processing PDF/X content references from **EXTERNAL_DATA_ARRAY** elements.

1.7 Structure promotes efficiency

In VDP applications, the structure of the VDP data format used must be such that it enables consuming applications and devices, such as workflow tools and digital printer raster image processors (RIP), to perform at or above the rated speed of the digital printer. In the past this has been accomplished through the use of proprietary PDLs optimized for use by a particular vendor's printer front-end hardware.

PPML/VDX also incorporates features for enabling RIP efficiency.

The ability to target a PPML/VDX job late in a production workflow places additional requirements on the structure of the PPML/VDX data. These requirements include the need to guarantee page independence and efficiency of access to each PPML page definition and its associated PDL content data.

This is necessary in order to guarantee page re-ordering efficiency in preparation for print production, since the order in which pages are utilized on a sequence of imposed sheets (printer spreads) is almost always different from the reader order in which the PPML/VDX data specifies them.

Other VDP PDLs, which combine layout information with actual content data, do not enforce page independence and usually cannot be efficiently parsed and manipulated as required by an imposition processor. For this reason, embedding graphical content data in the data stream using PPML's **INTERNAL_DATA** element is prohibited by the PPML/VDX standard. For similar reasons, PDF was selected as the only format for graphical content data in PPML/VDX.

Separation of the VDP authoring and print production processes in support of blind exchanges, as enabled by PPML/VDX, is empowered by the standard in the area of job-liability management. This includes the ability of the receiver of the exchanged data to reliably determine completeness of the exchanged job data.

Of course, PPML/VDX also satisfies all of the reliability requirements for use within more integrated, application-specific production workflows, including both high- and low-volume workflows involving any level of verification automation and design complexity.

1.8 Exchange and data reliability

The essence of PPML/VDX is data reliability. Because PPML/VDX utilizes PDF data as its exclusive content-file format, the PPML/VDX interface for prepress tools and printing devices is consistent and well defined.

CGATS SC6 TF2 (the committee that developed CGATS.20) believes that a strict-conforming PPML/VDX job has a greater potential for cross-platform portability and operating-environment interoperability than any other open VDP file format available today. There is also greater potential for improved performance, workflow flexibility, and creative expression – graphical content complexity. The exclusive use of PDF data for page-content data has the important benefit of allowing workflow-tool vendors and users to leverage many existing PDF prepress tools, such as off-the-shelf PDF preflight tools.

During the course of job processing, there will likely be a number of tools or processes through which the job will progress, involving a number of senders and receivers. As an example, a client sends a job to a printing firm for production. In the print production workflow, the PPML/VDX instance remains intact as it proceeds through the various prepress, press, and post-press stages of a print-production workflow.

Interpretation of a PPML/VDX instance is stateless and has no dependence on any other instance. As instances proceed through a given production process, run-state independence ensures that execution of any particular instance by a process or printing device is not affected by the execution of any prior instances through that same process.

The state of a process is affected only by the current instance’s job ticket. For this reason, the use of the Global value for the Scope attribute of the **OCCURRENCE** element in the PPML data is prohibited by the PPML/VDX standard.

Although one of the goals for the standard is to facilitate blind-exchange workflows, PPML/VDX also incorporates the ability to handle numerous other types of exchanges and workflows.

For example, in tightly coupled workflows and trusted exchanges, PPML/VDX-Content files may be reused among multiple PPML/VDX instances supplied by a client. In such a case, neither retransmission nor repeated revision checking of the pre-exchanged PPML/VDX-Content files is required, and the use of the MD5 checksums present in the **Binding** elements of the **ContentBindingTable** ensures such exchanges are reliable.

2 VDP Jobs and Workflows Using PPML/VDX

PPML/VDX is a page content format. The bulk of this application note is focused on PPML/VDX as a page content format. In this section, a JDF workflow context is used as the framework for describing PPML/VDX job processing.

As with traditional static print workflows, there is no single VDP workflow that can be defined to accommodate the production of all possible VDP applications. In fact, many different workflows and devices may be used in the production of a PPML/VDX job resulting in the manufacturing of the exact same VDP products. Figures 1 – 4 provide a graphical overview of possible PPML/VDX workflows.

2.1 Types of VDP workflow

VDP workflows can be differentiated based on the level of integration between the authoring system and the production system:

- ***Highly integrated workflows*** where the VDP content data itself is generated within the same environment as the consuming production workflow, and the composition engine has a-priori knowledge of production processes and all devices. See Figures 1 and 2.

- ***Non-integrated, open exchange workflows*** where the VDP content data is late stage targeted at a printing device(s) and postpress workflow. In this step a manual or automated workflow planning and prepress phase is required. See Figures 3 and 4.

- ***Semi-Integrated partial exchange*** where the VDP content is created in a different environment from the consuming production workflow, yet the content creator takes advantage of resources already at the production site. Note that this workflow is architecturally the same as the non-integrated, open exchange workflow shown in Figures 3 and 4.

PPML/VDX is suitable as a VDP content format for all of these types of VDP workflows. It is important to note that in these workflows the functions of authoring, production planning and production management differ as a function of the workflow architecture. Differences are noted in the following figures.

2.2 VDP jobs

A VDP job involves the production, on behalf of a given sender in a specific timeframe, of a set of print products where each product contains at least one unique component per recipient. As with any kind of print, a PPML/VDX job, in general, involves the steps of content authoring, digital asset transfer, production planning, and managing the activities of prepress, press, and post press phases.

For each individual recipient represented in the VDP job, a complete description is provided for each finished piece as well as the definition of the custom page content of each finished page surface of each piece.

Once a production contract is instantiated, a print production workflow is devised by a production planning system. The workflow plan usually involves the use of one or more RIPs and digital printers, as well as the finishing devices used in a production sequence. The activities of each device are considered production steps, or phases, in the execution of the VDP job. The term Job, however, is often used in the context of an individual device's participation in the workflow. In the context of this document, the term VDP job includes the definition and execution of all steps in the manufacturing process, and its definition includes all required data resources such as page content data.

A PPML/VDX job definition initially comprises the PPML/VDX instance data that includes the variable page content and optional JDF product intent data (stored within the **PPMLVDX/ProductIntent** element). Alternatively, a separate referential JDF product job ticket may accompany the PPML/VDX instance data where this JDF job ticket will contain a **RunList** resource that refers to the PPML/VDX-Layout file as the source of page data. Although different from the inferential (**TICKET_REF**) use of JDF as specified in the PPML/VDX Standard, this is another way of specifying a JDF product intent job ticket in accordance with the JDF specification that is similar to how JDF is used to specify product intent for static jobs.

A production workflow that accomplishes the manufacture of the VDP products is then determined. In a JDF job ticket managed production workflow, a JDF process job ticket is created and one or more JDF process nodes that model the workflow and devices utilized in the production workflow are added, as shown in the Production Planning portions in the following figures. It is the resulting JDF job ticket that embodies the complete production definition of the job (the JDF process nodes are usually added to the JDF product intent job ticket in the case where a JDF product intent job ticket already exists).

The *CIP4 Digital Printing Working Group* has developed extensions to the Job Definition Format (JDF) Specification that permits referential linkage of JDF product intent and process job ticket data to structured document formats characterized as MultiSet (**RunList/LayoutElement/@*ElementType***=*MultiSet*) such as those based on PPML data, PODi is finalizing a specification known as Digital Print Ticket (DPT) version 2.0 based on this linkage method developed by the CIP4 Digital Printing Working Group.

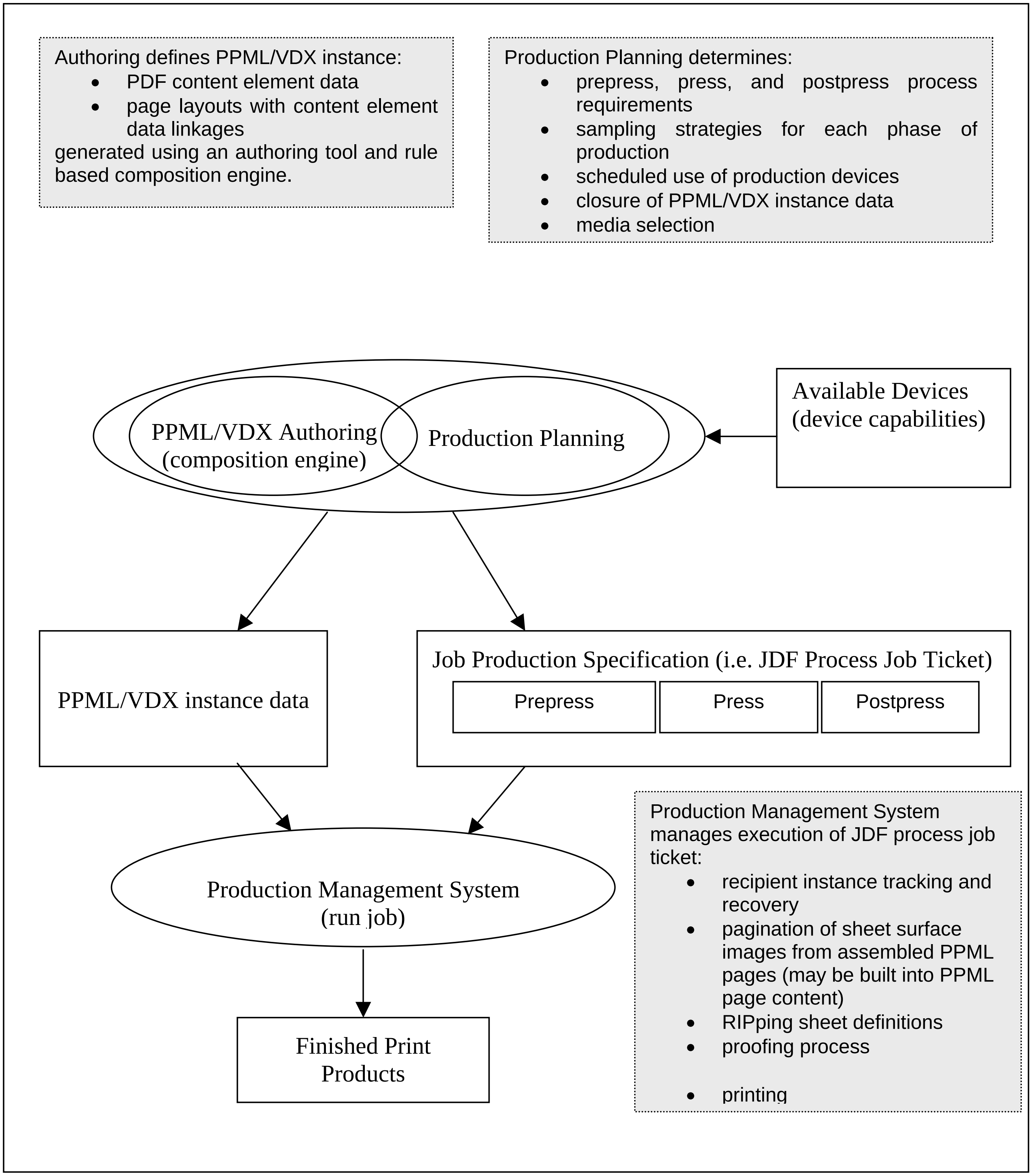
One or more JDF process nodes that describe imposition layout, RIPping, and digital printing, must have at least one input **RunList** resource that refers to the PPML/VDX-Layout file as the source for page content data. This assumes the printing system is receiving reader pages rather than pre-flattened imposed sheet-surface descriptions. Although possible in a workflow, design phase flattening reader pages into sheet surface layouts in most cases is sub-optimal in a job ticketed workflow because sheet marks applied during design are opaque to the production management system and the job cannot be easily retargeted to an alternate production process.

See 7.2.1 for a description of page, panel, and sheet.

Legend for Figures 2 and 4:

- Solid bold lines indicate transfer of PPML/VDX - Layout file.
- Dashed bold lines show transfer of PPML/VDX content files.
- Dotted lines indicate directional references to PPML/VDX -Content files from within the PPML/VDX - Layout file.
- Solid lines indicate access and use of related data.

2.3 Workflow



Authoring defines PPML/VDX instance:

- PDF content element data
- page layouts with content element data linkages

generated using an authoring tool and rule based composition engine.

Production Planning determines:

- prepress, press, and postpress process requirements
- sampling strategies for each phase of production
- scheduled use of production devices
- closure of PPML/VDX instance data
- media selection

Production Management System manages execution of JDF process job ticket:

- recipient instance tracking and recovery
- pagination of sheet surface images from assembled PPML pages (may be built into PPML page content)
- RIPping sheet definitions
- proofing process
- printing

Figure 1 — Highly integrated workflow

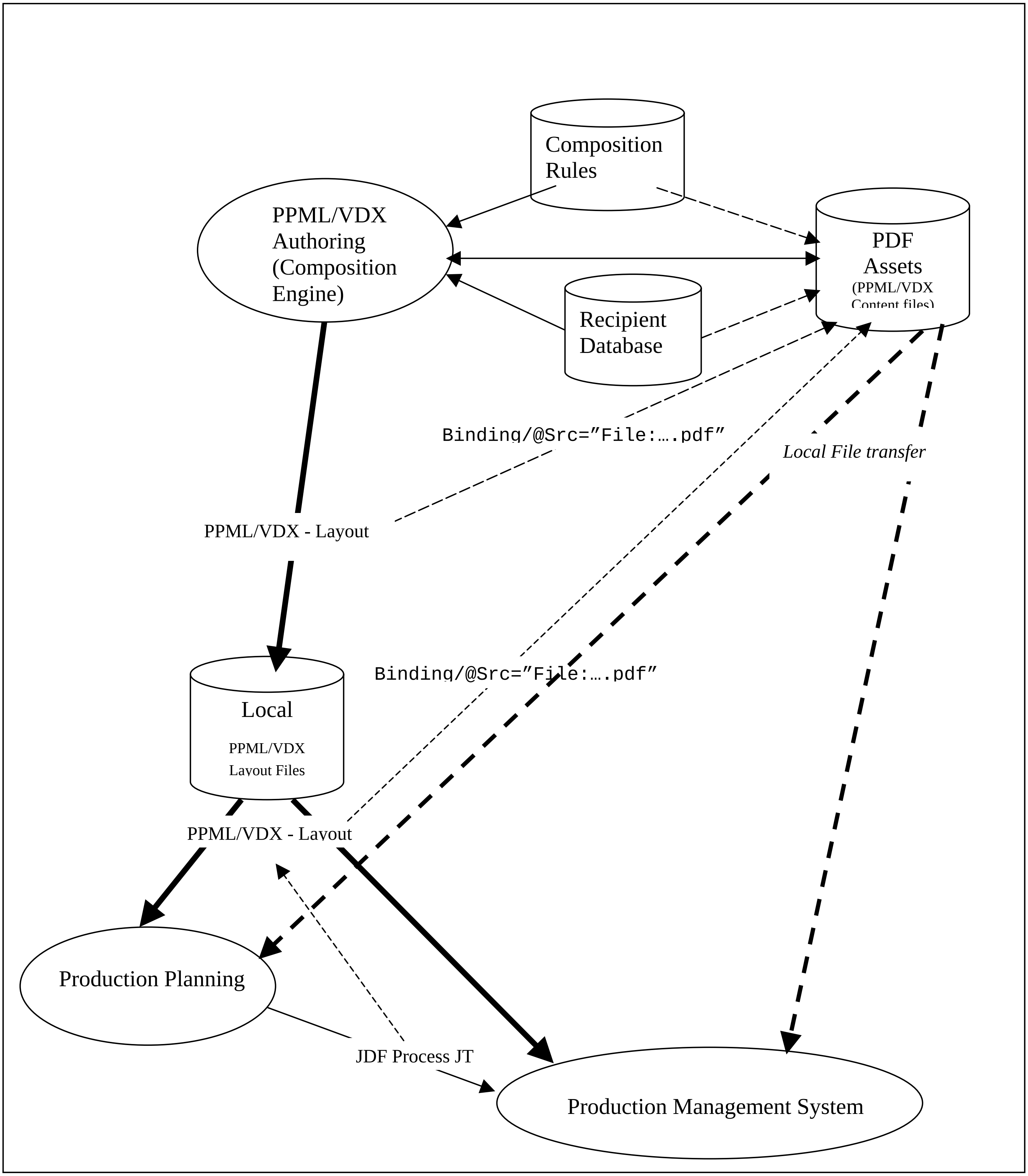


Figure 2 — VDX data asset management in a highly integrated workflow

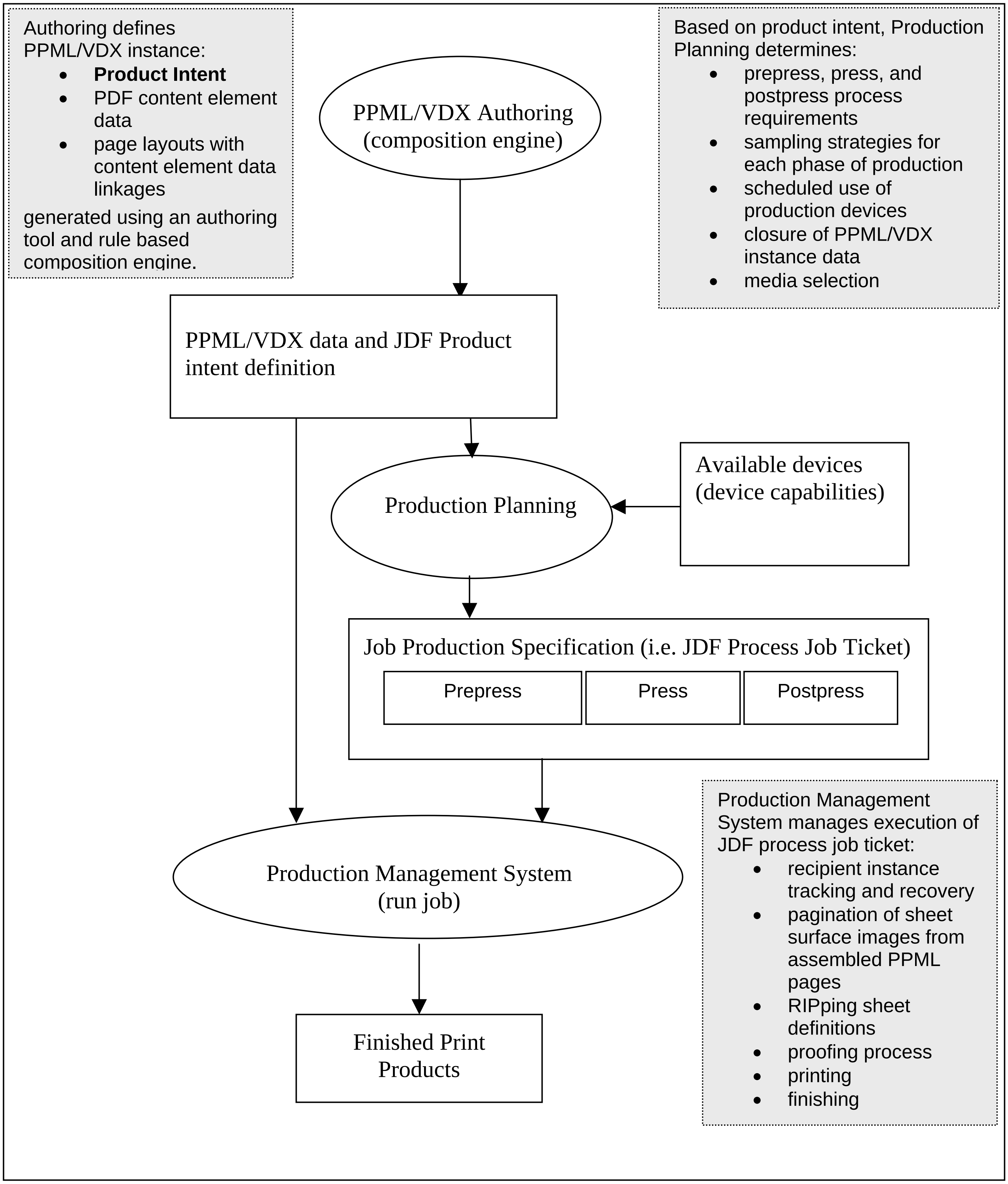
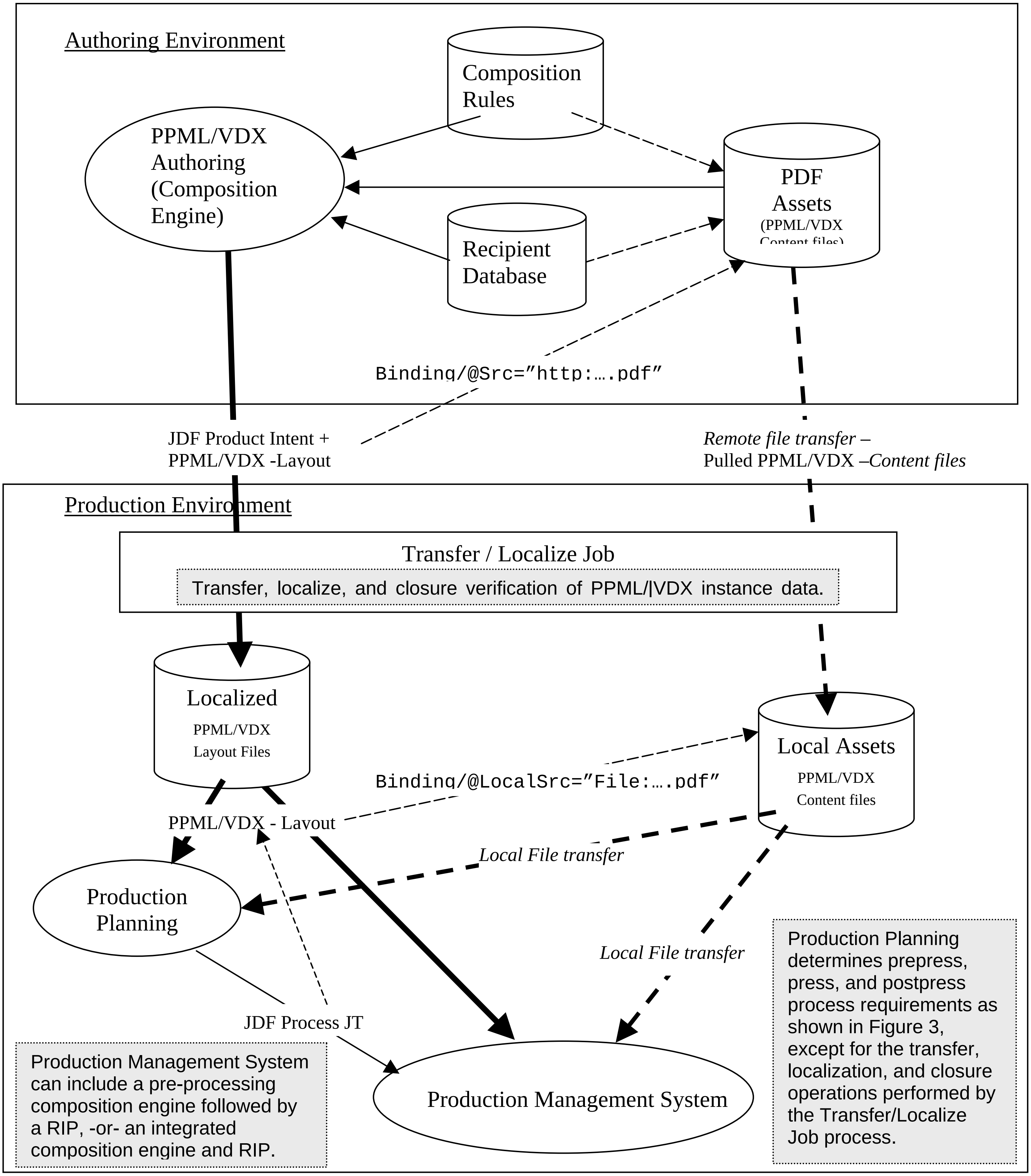

Authoring defines PPML/VDX instance:
Product Intent
PDF content element data
page layouts with content element data linkages
generated using an authoring tool and rule based composition engine.
PPML/VDX Authoring (composition engine)
Based on product intent, Production Planning determines:
prepress, press, and postpress process requirements
sampling strategies for each phase of production
scheduled use of production devices
closure of PPML/VDX instance data
media selection
PPML/VDX data and JDF Product intent definition
Production Planning
Available devices (device capabilities)
Job Production Specification (i.e. JDF Process Job Ticket)
Prepress
Press
Postpress
Production Management System manages execution of JDF process job ticket:
recipient instance tracking and recovery
pagination of sheet surface images from assembled PPML pages
RIPping sheet definitions
proofing process
printing
finishing
Production Management System (run job)
Finished Print Products

Figure 3 — Non- and semi-integrated exchange workflow


Authoring Environment
Composition Rules
PPML/VDX Authoring (Composition Engine)
PDF Assets (PPML/VDX Content files)
Recipient Database
Binding/@Src="http:….pdf"
JDF Product Intent + PPML/VDX -Layout
Remote file transfer – Pulled PPML/VDX –Content files
Production Environment
Transfer / Localize Job
Transfer, localize, and closure verification of PPML/|VDX instance data.
Localized PPML/VDX Layout Files
Local Assets PPML/VDX Content files
Binding/@LocalSrc="File:….pdf"
PPML/VDX - Layout
Local File transfer
Production Planning
Local File transfer
JDF Process JT
Production Management System

Production Management System can include a pre-processing composition engine followed by a RIP, -or- an integrated composition engine and RIP.

Production Planning determines prepress, press, and postpress process requirements as shown in Figure 3, except for the transfer, localization, and closure operations performed by the Transfer/Localize Job process.

Figure 4 — VDX data asset management in a non-integrated workflow

Prepress involves all activities associated with mapping reader pages defined in the PPML/VDX instance to the surfaces of a sequence of printed sheets of varying media styles. As with static production workflows, an imposed sheet may contain various static sheet marks such as slug lines, and color control bars, as well as fold and trim marks. Sheets printed from VDP jobs often include dynamic sheet marks such as barcodes that identify printed sheets and their sequence relationship to a set of sheets that belong to a particular recipient. The specification of the imposed sheet layout, including the composition and placement of all static and dynamic sheet marks, is usually specific to the off-line and/or near-line postpress finishing device(s) utilized in the target production workflow.

Printing of the recipient instances occurs in the press phase of the workflow and results in one or more stacks of sheet sets where each set belongs to a particular recipient. The quantity of printed sheets comprising each set may vary from set to set depending upon the number of variable pages the PPML/VDX data specifies for each recipient. Digital printing systems that operate in job ticketed workflows must therefore have the capability of consuming reader ordered VDP content pages conveyed in a PPML/VDX instance and generating imposed printer order sheets from instructions provided in the job ticket (i.e. JDF job ticket).

In order for a postpress off-line or near-line finishing device to discriminate sheet-set breaks and perform the finishing operation on the set of sheets belonging to the same recipient, the finishing machine must often scan and interpret barcode marks present on the sheets loaded in its feeder. Conceptually, the barcode marks imaged on the sheets links to the information present in the structure of PPML/VDX data that signals the reader page groupings for a given recipient instance. Other dynamic barcode marks may also be present on the printed sheets that are used by finishing machines to signal insertion of pre-printed matter into a given sheet set prior to folding, binding, and/or envelope insertion.

In general, most all sheet marks imaged onto sheet surfaces, other than the page content data, are workflow device dependent. This means that determination of static and dynamic sheet marks can only be accomplished with *a priori* knowledge of the production workflow configuration and control requirements of the finishing machines utilized. It is for this reason that a prepress step (manual or automatic) is required for utilizing PPML/VDX data in a production workflow.

For finishing machines that are JDF-job-ticket controllable and near-line connected to a production management system (i.e. network connected but unattached paper path) for use in more automated workflows, it may be necessary for the production management system or the digital printing system itself to covey in the JDF job ticket a description of the sheet layout. In a JDF-controlled workflow, the JDF job ticket data is used to preset the finishing device's configuration to accept and properly process the input stack of printed sheets. For example, sheet-layout characterization data may be conveyed in the output **Component** resource of the JDF digital printing process node (refer to the JDF Specification for details).

The benefit of a VDP workflow utilizing PPML/VDX with JDF is that the authoring system is able to remain decoupled from the production processes. The author is not required to possess knowledge of the manufacturing processes employed in the production workflow (i.e. knowledge of sheet imposition layout details). As can be seen from the preceding discussion, this approach requires the receiving production environment to have a prepress system that allows for the definition of appropriate process controls based on received intent information. Then the resulting process controls can be embodied in a JDF process job ticket. Alternatively, the prepress system can re-write the PPML data of PPML/VDX instances with the imposition layout for the target processes included.

A common VDP workflow environment assumes a close linkage between VDP authoring and VDP production workflows. PPML/VDX can be applied in this workflow as well. The pages of a PPML/VDX file, for example, can define a print order sequence of imposed impressions. In this case, the reader pages generated by the composition engine are already distributed into printer spreads compatible with particular printing, binding, and finishing processes. PPML data of such a PPML/VDX instance would not contain **TICKET_REF** elements used to indicate intent, because the intent and process requirements are already implicit in the job content. This usually implies the use of a manual means of conveying device control semantics.

Use of JDF is described in various areas of this document due to its rapid adoption by printing and finishing equipment manufacturers, and production workflow management system vendors, but PPML/VDX can function in numerous other workflow environments.

2.4 Workflow design objectives

The choice of workflow used in the production of a PPML/VDX job (or VDP job in general) is influenced by many factors including:

- Reliability and liability management

- Scalability to accommodate one-off or recurring jobs (i.e. same workflow but with different PPML/VDX data) on a routine basis

- Workflow efficiency and cost balance for target print volume and job complexity (run length)

- Color fidelity and graphical complexity management

- Robust digital asset management

2.5 Reliability and liability management

It is very often the case that regardless of VDP job classification, the expectation of the print customer is that exactly one print product is manufactured for each recipient – no more, no less. This requires that all printed components be tracked, commensurate with the business rules of the print contract.

In many cases the print client prefers to maintain absolute control of the page content and appearance of the job's variable page data. This data includes any digital assets in the form of PPML/VDX-Content files exchanged with the print provider as well as access to the database of intended recipients used in the creation of a PPML/VDX instance. Ideally, the database itself should not be exchanged to the print provider, and all necessary data should be included in the PPML/VDX instance's PDF/X files.

A reliable workflow begins with a complete exchange of all data comprising the PPML/VDX instance and the ability to rigorously verify that all required data components are present and under the control of the production system. These processes are known as *PPML/VDX data transfer* and *closure verification* (see section 4.3).

It is important to note that closure verification of a PPML/VDX instance may occur again at various points in the production workflow – even within the digital printing device itself. In many cases, printed component-level verification is performed within near-line or off-line finishing devices as well. This usually requires the use of barcodes printed outside the finished page content area of the printed sheets.

Once PPML/VDX data is exchanged and verified for closure, maintaining the integrity of the graphical data becomes the responsibility of the receiver, who must guarantee that the printed result exactly matches the expectation of the creator. For this to happen, the content data must conform to PPML/VDX-Strict. This level of conformance guarantees the receiver of the job that all data belonging to it is device independent and can be verified for closure.

PPML/VDX-Strict conformance requires all PDF data to conform to the PDF/X-1a or PDF/X-3 standards. This means that all color content data is prepared for a single characterized printing condition. Additionally, positive binding metadata is required where all **Binding** element references in the **ContentBindingTable** shall have both a **UniqueID** and **MD5_Checksum** present, so that the receiver can reliably verify closure of the exchanged data set.

Depending upon the established level of trust between the sender and receiver, it is selectively possible to relax the degree to which all data in a PPML/VDX instance must be fully defined and verifiable (e.g. use of unique IDs and checksums). The PPML/VDX-Relaxed conformance level is available for exactly this purpose. Relaxation will normally be done when sender and receiver have an on-going working relationship.

PPML/VDX-Relaxed conformance permits PDF data that does not conform to PDF/X-1a or PDF/X-3 to be included in a PPML/VDX instance. Use of the **UniqueID** and **MD5_Checksum** attributes of **ContentBindingTable**/**Binding** elements are optional. Relaxing the requirements that **UniqueID** and **MD5_Checksum** attributes be included enables workflows that permit late binding of PPML/VDX-Content data to a PPML/VDX instance. PPML/VDX-Content files can therefore be created and exchanged after the PPML/VDX-Layout file has already been exchanged and the absence of positive binding information will alert the closure verification process of special verification requirements. For example, examination of the **ContentBindingTable** and its sub-elements allows the receiver to determine the disposition of the data in terms of device independence and reliability of binding references. It is possible for a PPML/VDX-Relaxed instance to achieve a level of device independence and reliability equal to that of PPML/VDX-Strict conformance.

2.6 Recurring and one-off (on-demand) VDP workflows

There are essentially two fundamental paradigms for VDP workflow systems:

1. Workflow systems that are somewhat dedicated to accomplish a very specific VDP print application. This is usually the case for most recurring VDP jobs.

2. Workflow systems that are very flexible and accommodate a wide variety of print jobs. In this case, the production workflow may be configured in a quick and efficient manner, on a demand basis. Such completely digital workflows are ideally job ticket controlled (i.e. JDF based).

Many VDP applications demand highly-integrated authoring and production workflows because they are high volume, and require a high degree of reliability. Many transactional printing systems fall into this category. Such specialized VDP authoring and production systems are carefully engineered to accomplish the manufacture of only a very specific print application. As highly controlled and integrated systems, often with dedicated equipment, they usually possess little flexibility for accommodating a mix of VDP job types at any one time. To achieve a sufficient economy of scale for such dedicated VDP workflows so as to overcome large engineering and dedicated equipment costs, these workflows are usually only used in the case of high volume, frequently recurring VDP jobs. In general, recurring jobs vary only in aspects of printed content where the specification for the finished print product is the same for each job.

On the other extreme are non-recurring, on-demand VDP jobs. In this paradigm, the print provider must be able to construct a suitable VDP production workflow on an as needed, per job basis. The print provider is able to accept many more smaller, possibly higher value, graphically rich VDP jobs and produce them without the need for a costly workflow re-engineering procedure each time a completely new VDP job is to be produced. Ideally, such an ad-hoc approach can be made scalable and page volume independent. Success in this paradigm places a great deal of onus on interoperability of VDP content formats, job control, and a unification of device interfaces - thus requiring standardization in many areas of the custom print authoring and manufacturing process.

Until recently, on-demand VDP workflows have been uncommon due to the lack of standardization of various production device interfaces – including VDP content formats. Constructing custom VDP workflows on the fly has been seen as nearly impossible. Enter PPML/VDX and JDF – two complementary technologies which together offer significant opportunity in the area of increased demand for VDP.

2.7 Workflow efficiency and cost balance for target print volume and job complexity

In many cases VDP jobs are considered long run – especially in cases where many thousands of recipients are targeted in a single distribution. Such jobs are characterized as high-volume. In general, the higher the volume of pages output from a job, the greater the need for production workflow automation. In many cases, high-volume transactional jobs are highly automated in order to reduce human interaction. Often, high-volume workflow automation requires auditing at various workflow stages in order to guarantee reliability. Imprinting of barcode encoded audit information is commonly used for tracking printed components and detecting spoilage.

In the case of low-volume or short-run VDP jobs, reliability is as important, however page volume may be low enough that complete automation throughout prepress, press, and postpress is not cost effective or necessary.

Large PPML/VDX jobs need not necessarily be produced in a single print run, and may even be produced in separate production workflows at different times and at different locations. For example, there are cases when it would be reasonable to segment the PPML/VDX instance into a discrete sequence of instances in order to accomplish time shifted production runs or to produce the job in multiple parallel runs (using multiple printers) in order to achieve sufficient output.

Workflow complexity increases when multiple job components are combined at various stages. For example, offset printed shells may be used as the print medium for a VDP print run. Static and multiple VDP printed materials may be assembled at finishing. Complex jobs may involve selective insertion of components from other workflow steps to be included within the envelope that packages a recipient's job instance.

PPML/VDX is intended as a VDP content carrier for any of these scenarios while JDF job ticket manages the workflow.

PPML/VDX instance as part of a larger JDF job context:

A JDF process job ticket may contain one or more digital printing process nodes that refer to PPML/VDX Layout file(s) as page content resources.

An example of the steps involved may include:

1. The PPML/VDX authoring phase generates a PPML/VDX instance with JDF product intent description.

2. The PPML/VDX instance is exchanged into a production environment.

3. In the Production Planning phase of a JDF enabled production workflow system, the PPML/VDX data is analyzed and a production workflow is defined.

4. The workflow process steps are specified by adding JDF process nodes to the JDF product intent job ticket. This completes the JDF job ticket contains one or more JDF digital printing process nodes, each of which contains a **RunList** resource that references one or more PPML/VDX-Layout files.

This JDF process job ticket now represents the VDP job product intent definition and the JDF processing instructions for manufacturing the job. Such use of PPML/VDX with a referential JDF process job ticket requires a particular structuring of the PPML data as specified in section 7.0. This JDF approach is consistent with the use of JDF as defined by the JDF Specification where the digital printing device executing a JDF process node ignores any **TICKET_REF** elements present in the PPML data.

2.8 Color fidelity and graphical complexity management

PPML/VDX provides mechanisms for the integration of all required data to permit effective process control and color management

Unambiguously defined color is a critical aspect of PPML/VDX flexibility. Key mechanisms are available within strictly conforming PPML/VDX, when the Binding element IntendedColor attribute is set True, to provide these capabilities with respect to color:

- Strictly conforming VDX restricts content data to either PDF/X-1a or PDF/X-3.
- PDF/X-1a and PDF/X-3 files contain ICC color management metadata

Care is required in achieving correctly created PDF/X-1a and PDF/X-3. Hence, "provides the mechanisms" should not be read as "PPML/VDX does it for you."

PDF/X-1a files, by definition, have been color-rendered to a particular and well-defined printing condition, and contain only CMYK and spot-color data. PDF/X-3 files, by definition, can contain three-component (e.g. RGB), CMYK and spot-color data.

In both cases, all color PDF/X objects must be associated with a colorimetric source-color definition. These data must also be targeted to a defined printing condition using included ICC color profiles, or pointers to characterization data. Where multiple PDF/X content files are used, it is important to ensure that they all target the same defined printing condition.

In addition to the reference printing conditions currently available on the ICC website, a suite of reference color characterization data sets is being developed by CGATS SC6 TF2 specifically for digital printing machines. These include large, intermediate, and small color gamut characterizations. The large gamut case corresponds to the characterization for high gloss image characteristics; the intermediate gamut case corresponds to the low gloss image characterization, and the small gamut case corresponds to ANSI CGATS TR 001. The two Technical Reports under development are:

- CGATS TR 007 *Graphic technology – Reference color characterization data for non-impact digital printing – Grade 1 substrate (white point) and typically high gloss image characteristics*

- CGATS TR 008 G*raphic technology – Reference color characterization data for non-impact digital printing – Grade 1 substrate (white point) and typically low gloss (satin) image characteristics*

For any job, portability of color among digital printing machines is enabled via color re-targeting from the original PDF/X output intent printing conditions.

A final point is necessary regarding effective color-management systems. Because PDF/X requires the use of ICC-based color metadata, ICC color management mechanisms can be used for proofing and re-targeting.

This is a key distinction for strict PPML/VDX in comparison with other VDP formats. Postscript files and untagged color files do not contain metadata to enable color-accurate proofing or re-targeting, which is a key aspect of PPML/VDX portability.

However, it should be noted that "providing the mechanisms" is not sufficient to obtain controlled color. End-to-end color-managed workflow processes are required to reap the available process-control benefit.

2.9 Robust digital asset management

Robust management of the digital assets comprising a PPML/VDX instance requires careful versioning control and management of all URL references. For example, when a content element of a PPML/VDX instance is changed for a particular use, other uses of that content element that are affected by the change must be controlled in an explicit manner.

PPML/VDX works well in a content managed environment where a PPML/VDX instance can be created and guaranteed complete throughout prepress processing using the attributes of the **ContentBindingTable/Binding** elements. All components identified by the **ContentBindingTable/Binding**/@***Src*** attributes can be gathered into the local environment and verified as the exact version using the various verification attributes of each **ContentBindingTable/Binding** element.

When PPML/VDX Content files are gathered into a local repository and processed for closure, the **ContentBindingTable** element should be updated such that the ***LocalSrc*** attribute of each **ContentBindingTable/Binding** element references the local copy of the PPML/VDX-Content file. The *Src* attribute should always be left as a reference to the original copy of the PPML/VDX-Content file. The **ContentBindingTable/Binding**/@***MD5_Checksum*** and the **ContentBindingTable/Binding**/@***UniqueID*** (required in strict) attributes are available to verify closure when the job content data is transferred.

As a function of workflow design, and knowledge of the expected volatility of referenced digital assets, late-stage content closure process, may re-verify all **ContentBindingTable/Binding**/@***MD5_Checksum*** attributes at any point.

In addition to closure verification, robust management of PPML/VDX digital assets requires persistence of the verified PPML/VDX-Content files from the time that production is started to the time that the job is completely finished and shipped. If any mid-production failure recovery is required, the job content can be ensured by workflow processes to be available and consistent. Business factors may dictate that digital assets of PPML/VDX instances persist indefinitely.

Use of particular content in a particular PPML/VDX instance can also be transitory, and can be managed for robustness using careful coordination between the PPML/VDX data creator and the print provider. For example, in certain VDP workflows a PPML/VDX-Layout file can be created and exchanged prior to the completion of all dependent PPML/VDX-Content files. Such a PPML/VDX-Layout file can be received at a print provider before all of the content located through the **ContentBindingTable** exists. In this case proper ***MD5_Checksum*** attributes of **ContentBindingTable/Binding** elements cannot be specified but may be present and have a dummy checksum values used as placeholders.

When a PPML/VDX instance's **ContentBindingTable/Binding**/@***Src*** URI reference cannot be resolved, or can be resolved but has an invalid **ContentBindingTable/Binding**/@***MD5_Checksum*** attribute value, that job cannot be verified for closure. In this situation, an oversight scheduling process, such as a JDF MIS or prepress management system, must manage the delayed closure verification processing for such a late binding job. Separate notification of content availability (out of band to the exchange of the PPML VDX data) or delivery of the late content to a specified **ContentBindingTable/Binding**/@***LocalSrc*** location (local disk, file server, ftp server, etc.) can trigger the final verification process to determine closure. When closure verification is triggered, the immediate failure of the dummy checksum can trigger retrieval of the external **ContentBindingTable/Binding**/@***Src*** content and a correct ***MD5_Checksum*** attribute value. Alternatively, the separately delivered content itself can include an MD5 checksum value that can be used to update the **CBT/Binding/@*MD5_Checksum*** element. Acceptance of the referenced PPML/VDX-Content file as correct therefore depends on business rules.

3 Structure of a PPML/VDX Instance

A PPML/VDX instance can be a single file, or multiple files. Which scheme of a PPML/VDX instance to use depends entirely upon workflow and requirements of the data exchange.

3.1 Single file PPML/VDX instance

A single file PPML/VDX instance is comprised of a single PDF file known as the PPML/VDX-Layout file. According to the PPML/VDX standard, this file should have a *.vdx* file name extension.

This special PDF file must contain a PDF stream object that contains the XML **PPMLVDX** element and is referenced from an entry in the file's **catalog** dictionary.

At the very least, this **PPMLVDX** element must contain a **ContentBindingTable** sub-element followed by a **Layout** sub-element. It may also contain an optional **ProductIntent** element, which if present, must contain a **JDF** sub-element or a **JDFRef** sub-element that is a reference to an XML file that contains a **JDF** element. This **JDF** element must be a conforming JDF product intent node as defined by the JDF specification and further restricted by the PPML/VDX standard.

The **ContentBindingTable** must contain a **Self** sub-element that has an ***Src*** attribute with a URI value that matches the value of the ***Src*** attribute of all **EXTERNAL_DATA_ARRAY** elements present in the PPML data.

3.2 Multiple file PPML/VDX instance

There are several schemes of multiple file PPML/VDX instances. In general, most multi-file PPML/VDX instances are comprised entirely of PDF files: exactly one PPML/VDX-Layout file, and one or more PPML/VDX-Content (PDF) files.

The minimum requirements for the PPML/VDX-Layout file of a multi-file PPML/VDX instance are the same as for a single file PPML/VDX instance (see the description under the previous heading).

The difference in this case lies in the definition of the **ContentBindingTable** where it must contain a single **Binding** sub-element for each PPML/VDX-Content file included in the PPML/VDX instance. In general, the **ContentBindingTable** is a manifest that exactly identifies the entire set of PPML/VDX-Content files that are members of the PPML/VDX instance's file set.

For each member PPML/VDX-Content file, a separate **Binding** element entry referencing it must be present in the **ContentBindingTable**.

The **ContentBindingTable** in this case is not necessarily required to contain a **Self** sub-element. This is the case where none of the PDF pages of the PPML/VDX-Layout file are used as compound elements referenced from the PPML data.

3.3 Use of the ContentBindingTable

The **ContentBindingTable** is most useful with respect to multi-file PPML/VDX instances and has several functional uses including:

1. Provides a complete manifest of dependent PPML/VDX-Content files. For each PPML/VDX-Content file included in the PPML/VDX instance, a separate **Binding** sub-element must be present. Each **Binding** sub-element must have an ***Src*** attribute present that is a URI that identifies a PPML/VDX-Content file. This URI must be resolvable by a reader in the receiving environment. For each unique

EXTERNAL_DATA_ARRAY/@***Src*** value there must be a corresponding **Binding** element entry in the **ContentBindingTable** that has a matching ***Src*** URI value.

2. Provides optional meta-data used by a reader (receiver) to guarantee a positive binding of the **EXTERNAL_DATA_ARRAY**/@***Src*** to a PPML/VDX-Content file. This is important in the case where the job must be tested for closure in a receiving environment. This requires use of the optional ***UniqueID*** and/or ***MD5_Checksum*** attributes of a **Binding** element. If these attributes are present, a tool responsible for closure determination must use them to verify that the PPML/VDX-Content file resolved from the **Binding**/@***Src*** attribute is indeed the correct file as intended by the sender.

   - In the case of PPML/VDX-Strict conforming instance, both the ***UniqueID*** and ***MD5_Checksum*** attributes must be present in each **Binding** element so that closure of the exchanged PPML/VDX instance's file set can be guaranteed.

   - In the case of PPML/VDX-Relaxed they are optional and may be used as required by the exchange agreement.

     a. If the ***MD5_Checksum*** attribute is present, then the receiver of the data is obligated to verify that the calculated MD5 checksum of the PPML/VDX-Content file resolved matches the attribute's value

        i. If the calculated value doesn't match, then the resolved PPML/VDX-Content file is considered invalid

        ii. Likewise, if the PPML/VDX Content file's **trailer** dictionary doesn't match, the PPML/VDX-Content file is considered invalid

     b. Depending upon the business terms under which the job is being produced, the invalid PPML/VDX content file must either be corrected prior to job execution, or the entire PPML/VDX instance rejected

3. Provides a URI translation for localized PPML/VDX-Content files. For workflows involving inter-environment exchange, the **Binding**/@***Src*** attribute may identify a PPML/VDX-Content file located on a remote server, outside of the control of the production environment. To complete the exchange, it is often desirable to resolve all **Binding**/@***Src*** referenced files and copy them into the local environment if a copy is not already present. Once copied, a **Binding**/@***LocalSrc*** attribute must then be added to each **Binding** element that refers to the local copy. This is the process of PPML/VDX data localization. From a conforming PPML/VDX reader's perspective, if the ***LocalSrc*** attribute is present, it must use that URL to resolve the PPML/VDX-Content file reference, not the URI specified by the ***Src*** attribute. This translation mechanism eliminates the need to update all of the **EXTERNAL_DATA_ARRAY**/@***Src*** attributes present in the PPML data. (See section 5.2.)

4. Provides an indication of the color reproduction expectations of the PPML/VDX data originator. A **Binding** element has an optional ***IntendedColor*** attribute of type Boolean. If set to *true*, the referenced PPML/VDX-Content file must conform to either the PDF/X-1a or PDF/X-3 standards. If the resolved file is not appropriately conforming as indicated by this attribute, the PPML/VDX instance is considered invalid, and depending upon the business terms under which the job is being produced, the invalid file must either be corrected prior to job execution, or rejected.

The **ContentBindingTable** is a feature that is unique to PPML/VDX. PODi's PPML and PPML/GA have no equivalent capability. Therefore, a PPML consumer that also supports the PDF content format (i.e. one that conforms to PPML/GA) has no native provision for remote to local URL translation. This means that it is necessary to revise all **EXTERNAL_DATA_ARRAY**/@***Src*** attributes to refer to the corresponding Binding/LocalSrc prior to submission to such a consumer.

3.4 Point of entry

The PPML/VDX Layout file is the point of entry for a conforming reader.

A PPML/VDX reader uses the PPML/VDX Layout file for direct or indirect access to the PPML layout data as well as the optional JDF product intent data all of which is stored within the **PPMLVDX** element.

A PPML reader uses the **PPML** element as the entry point of the jobs reader order page data, and the **ContentBindingTable** to determine the manifest of required PPML/VDX Content files.

If JDF product intent data is present in the PPML/VDX instance as defined by the PPML/VDX standard, interpretation of the JDF data is with respect to references pointing back at it from the **TICKET_REF** elements within the PPML layout data.

If a PPML/VDX layout file is a component of a JDF process job ticket, then the JDF process job ticket is the entry point. The PPML/VDX file is then considered a resource of the JDF job ticket. (See section 2.)

4 Preflighting PPML/VDX instances

4.1 Overview

In all workflows, an appropriate level of preflighting and validation of the PPML/VDX instance should occur before it is committed for production. This application note discusses the steps involved in preflight for a PPML/VDX instance – the steps that should be taken by these workflows to validate the instance. The ultimate goal of these preflight steps is to ensure that an instance can be reliably expressed as a set of finished products. This means that all files are present and verified as the intended versions, that color information is suitably specified, and in general that the finished product can be produced to meet expectations for appearance and production schedule.

In the graphic arts, preflight generally refers to the process of ensuring necessary resources are available to assemble a document, as well as examining the document files to ensure that the documents to be generated match, as closely as possible, the appearance expected by a customer or supplier. As the PPML/VDX format is designed to facilitate blind exchange, the receiver of a PPML/VDX instance may assume that, if all required resources for the instance are present, then the layout is as intended by the sender of the instance. Preflighting a PPML/VDX instance is therefore restricted to checking that all required resources are present and can be verified as ready for prepress and production.

A PPML/VDX instance need not be complete to begin preflight on portions of the instance. Files and resources that are known to be present are eligible for validation, and it may be an advantage to preflight components of an instance as early as possible so that, if a component is damaged or not as requested, this information can be communicated back to the sender.

Nevertheless, a complete preflight should be carried out once all components are specified for and available to the recipient. This is recommended to ensure that files have not become inaccessible or been replaced by incorrect versions.

It is strongly recommended that all fonts used in a given content element be embedded (full or subset) in the PDF document containing that content.

The recommended steps for complete preflight of a PPML/VDX instance are:

1) Gather information and verify the PPML/VDX XML data.

2) Verify that all referenced resources are present and correct, known as verifying *closure* for the PPML/VDX instance including the presence of all required fonts.

3) For a PPML/VDX relaxed instance, if the instance is not closed, then closure may have to be managed through interaction between sender and receiver.

Each of these is discussed in detail in the sections that follow.

4.2 Gathering information and verifying the PPML/VDX XML data

The process of preflighting a PPML/VDX instance varies depending on whether the instance is strict or relaxed. Determination of whether the instance is strict or relaxed is therefore the first step in preflight.

To determine whether a PPML/VDX instance is strict or relaxed, retrieve the value of the **GTS_PPMLVDXConformance** key from the PPML/VDX layout file's **Info** dictionary. There are two valid values for this key: *(PPML/VDX-Strict:2002)* and *(PPML/VDX-Relaxed:2002).* The former indicates that the PPML/VDX file set is a strict instance, and the latter indicates that it is a relaxed instance. A different value, including no value, is an indication that the file is not a valid PPML/VDX layout file.

The XML in the **PPMLVDX** element should be validated against a schema to ensure that the **PPMLVDX** element conforms to the requirements for a PPML/VDX instance. The schema for the **PPMLVDX** XML element is given in section 7.

The PPML/VDX layout file contains the **ContentBindingTable** element (also referred to in this application note as the content binding table) for the PPML/VDX instance as a sub-element of the **PPMLVDX** element. Although the content binding table is able to contain validation information for all content files and external XML streams, it cannot itself contain the information required for validating the PPML/VDX layout file, since the process of including a checksum in the layout file would invalidate the checksum written to the file. The PPML/VDX standard therefore does not mandate a scheme for validating this file. It is suggested, however, that a receiver request an MD5 checksum for any PPML/VDX layout file accepted and compare this against the calculated MD5 checksum for the received file. In this manner a recipient can have confidence in the fidelity of the layout file.

Care must be taken during exchange of a PPML/VDX instance to ensure that inadvertent corruption of components does not occur. The most common causes of file corruption during transmission are using ASCII mode in an FTP client, as well as using other transmission agents that may substitute end-of-line (CR/LF) character sequences based on the destination environment, such as network file sharing (NFS) clients.

It is imperative that FTP clients transmit PPML/VDX instance files using binary (image) mode exclusively to prevent file checksum invalidation. NFS or other clients that cannot be instructed to suppress line ending or other character substitution should be avoided. File corruption will cause a PPML/VDX consumer to calculate checksums that do not match those entered in the content binding table by the producer prior to the exchange.

File corruption to PPML/VDX layout or content files is usually not repairable and will often require retransmission.

For XML data, however, corruption due to CR/LF substitution can be repaired if both the PPML/VDX sender and receiver allow it. If replacement of line-ending characters in an XML file yields a checksum that matches that calculated prior to the exchange, then it may be assumed that line ending substitution was the only corruption. If the checksums do not match then additional corruption occurred during the exchange and the files must be retransmitted.

4.3 Verifying the PPML/VDX instance data

A PPML/VDX instance is said to be *closed* when all content and layout data, as well as resources, are verified as correctly received. Once a receiver has access to all resources for an instance, and all components have been verified as undamaged and correct, closure has been verified. Instances that either have not been preflighted, or that fail preflight, are known as *open* PPML/VDX instances.

An open instance will typically have either missing files or resources, which the receiver may request from the sender if the sender has stated that the instance has been fully transferred.

For strict PPML/VDX instances, additional steps are necessary to verify that an instance is closed, such as verifying that all PDF data conforms to the PDF/X-1a or PDF/X-3 standards. A table of disallowed conditions in strict PPML/VDX instances can be found as Annex E of the PPML/VDX standard. In the general case, a process capable of fully parsing and interpreting PDF files will be required to ensure a PDF file meets one of the permitted PDF/X standards.

Some relaxed PPML/VDX instances may not be verifiable as closed without further communication with the sender. If closure cannot be verified, a preflight tool should inspect the PPML/VDX file set to determine what tasks must be carried out to achieve closure and provide this information to the recipient of the instance. For example, an instance may be open due to version mismatches between files, or unidentified color data. The section on managing closure (section 5) describes steps that can be used when insufficient information is present to continue processing the instance.

4.4 Closure assertion and confirmation

The central idea surrounding closure assertion versus closure confirmation is that a strict PPML/VDX instance **asserts** closure, because a strict instance is required to meet all of the PPML/VDX-Strict conformance criteria laid out in the PPML/VDX standard. When inspecting a strict instance, the primary responsibility of a preflight process is to **confirm** that the instance is closed. *No assumption should be made that an instance is closed based on the assertion of the sender*.

All assertions made by the sender of a PPML/VDX instance should be verified, whether a strict or relaxed instance. A recipient should not process any instance containing assertions proven to be false without consulting the sender.

4.5 Overview of the closure process

It is important to understand that a preflight tool must complete the following tasks to confirm closure:

4) The PPML/VDX layout file must be confirmed to be a conforming PPML/VDX layout file (see Clause 5 of CGATS.20-2002).

5) For each entry in the content binding table, the *Src* reference must indicate a PDF file that is accessible, and the values of the ***MD5_Checksum***, ***BaseID*** and ***UniqueID*** attributes of the **Binding** sub-elements in the **ContentBindingTable** entry must be validated, if present.

6) Each file referenced by an **EXTERNAL_DATA_ARRAY** element in the PPML layout data must have a corresponding entry in the content binding table.

7) All other files in the PPML/VDX file set must be confirmed to be in compliance with the PPML/VDX standard and the standards or specifications governing these components.

8) Validation of the ***IntendedColor*** attribute of the **Self** and **Binding** sub-elements in the **ContentBindingTable** must be carried out for entries where this value is *true*.

The presence of all required font data in the layout and content files of a strict PPML/VDX instance must be verified. In the relaxed case, fonts may also be listed as resources in the PPML data stream; for example, when a sender cannot embed these fonts due to license restrictions. These should be verified as present and correct at the receiver as described the next section. It is extremely important that fonts not be substituted without explicit instruction.

The PPML/VDX standard gives specific criteria for verification of the integrity of a strict PPML/VDX instance. The criteria for verification of the integrity of a relaxed instance beyond the requirements of the PPML/VDX standard are implementation dependent.

4.6 Implications of Strict and Relaxed conformance on preflighting

A PPML/VDX instance may be *open* because a complete preflight cycle has not taken place, not all content files or resources have arrived, or the failure of a preflight process.

If an instance is open due to files not being received, preflighting cannot be completed. Once all files have been transmitted, or a sender has sent notice that it is permissible to process a PPML/VDX instance, preflighting should be completed. As previously noted, preflight on resources that have been received is, of course, permissible.

If all files have been received and the instance fails preflight, then the PPML/VDX instance has one or more errors. Strict conforming PPML/VDX instances known to be open for this reason should not be accepted by a receiver in a blind exchange workflow.

Relaxed PPML/VDX instances are typically used in workflows where additional technical communication is expected to occur between clients and the entity producing an instance. Retransmission requests as well as negotiation over acceptable levels of risk for an open PPML/VDX instance mean that an instance that failed validation of closure should likely be maintained, and the portions that prevent closure remedied. Through this sender and recipient can agree on an acceptable level of risk to be assumed during the production of a PPML/VDX instance. This is known as *managing closure*.

5 Validating a PPML/VDX Instance

5.1 Overview

There are a number of tasks involved in determining if a PPML/VDX instance is complete and has closure. For PPML/VDX instances which are not closed, a recipient will wish to determine if the instance is acceptable for processing, and if not, determination of what components will need to be supplied by the sender.

The files comprising the PPML/VDX instance should be brought under the recipient's control before any processing. This means that the recipient has copies of all files in the PPML/VDX instance that cannot be changed during processing by the sender or any third party.

5.2 Resolving references to PPML/VDX-Content files

The ordered steps of a typical implementation are:

- Receive PPML/VDX-Layout file and validate file system consistency.

- Check conformance of PPML/VDX layout file (i.e. PDF structure), including verification that all required keys are present in the /Catalog and have valid values.

- Verify that the XML from the PPMLVDX element is well-formed and valid.

- Perform a consistency check between the values of the **EXTERNAL_DATA_ARRAY::*Src*** attributes in the PPML data and the ***Src*** attribute of the **Binding** element entries in **ContentBindingTable**.

- Traverse the **ContentBindingTable**, and for each **Binding** element entry resolve the URI referencing the PPML/VDX-Content file (PDF file). NOTE: Consumer should look for the /Base key in the layout file's /Catalog object and if present use the value as the base URI for all URI's in the **ContentBindingTable**. The /*Base* key would be used in those cases where all of the assets referenced from the **ContentBindingTable** are located on a system that is different from the location of the layout file.

  The recommended procedure for resolving a URI found in the **ContentBindingTable** would be use the /*Base* entry, if present, concatenated with the URI. If that resolution attempt fails, treat the URI as absolute and re-resolve.

  Application behavior in the failure to resolve a URI depends on the workflow and is not dictated by this note or the PPML/VDX specification itself.

5.2.1 Resolving URIs from Binding element entries

It is recommended that the URI of the **Binding** element entry only use the http, https, ftp and file protocol schemes. When resolving these URIs, the following contexts may be encountered:

1) The *Src* URI is a URL that refers to a PPML/VDX-Content PDF file in the local environment. This is a likely scenario in workflows where the PPML/VDX data producer resides in the same environment as the receiver (e.g. integrated authoring and production workflows). The following is an example URI syntax:

   "<file:///Acme/VDPJobs/assets/promo/ppmlvdxContent/version1.3/logo.pdf>".

2) The *Src* URI is a URL that refers to a PPML/VDX-Content PDF file in a different environment. Note that the PPML/VDX standard requires that the URI be resolvable from within the receiving environment. This is a likely scenario in workflows where the PPML/VDX data producer or content data provider resides in a remote environment relative to the receiver. The following is an example URI syntax:

   "<http://www.acme.com/VDPJobs/assets/promo/ppmlvdxContent/version1.3/logo.pdf>".

All discussions of URI resolution for Binding element entries should also apply to the resolution of **PPMLRef** and **JDFRef** URIs.

For each uniquely identified PPML/VDX-Content file transferred or verified as present, compute the MD5 checksum and compare to the value of the **MD5Checksum** attribute of the associated **Binding** entry. There should be uninterrupted file integrity between the checksum computation and asset use. If this cannot be guaranteed, then the system should re-compute and re-verify the checksum prior to asset usage.

5.2.2 Resolving Remote to Local Binding

There are three technical approaches to maintaining a proper binding between the PPML/VDX-Layout file reference and the transferred PPML/VDX-Content PDF files when managed locally:

1) Update the **LocalSrc** attribute of the **Binding** element of the **ContentBindingTable** to refer directly to the PPML/VDX-Content PDF file. Henceforth, this value must be used as the indirect reference by the workflow when resolving an **EXTERNAL_DATA_ARRAY::*Src*** reference.

2) Use a proxy server or a URI resolving service. This requires thorough integration of all elements to use that proxy interface. This may be more specialized than a standardized off-the-shelf proxy service because persistence management of resolved and transferred resources is required in the case where local control of PPML/VDX-Content files is required. Also, many off-the-shelf services provide caching facilities that may not be appropriate in a given workflow.

3) Edit the PPML data and revise all **EXTERNAL_DATA_ARRAY::*Src*** attributes to take the value of the localized URL. In this scenario it is very much encouraged that the corresponding **ContentBindingTable/Binding/@*Src*** attribute be updated to maintain consistency. It is important to note that doing this means a total loss of context in terms of the original reference to the remote PPML/VDX-Content file. It is then no longer possible to say that a particular URI fails when it is no longer present in the remote environment.

Modification of any part of a PPML/VDX-layout file (such as an embedded **ContentBindingTable** or PPML document) requires a tool with intimate knowledge of the PDF file format so as not to damage it.

5.3 Determining file set completeness and correctness

The first step in determining closure is to ensure that all referenced data has been transferred without error and is accessible to the receiver. Start by confirming the presence of a valid **PPMLVDX** element. If the **PPMLVDX** element is valid, then search the element for the **ContentBindingTable** sub-element. Within this sub-element there is allowed to be a **Self** sub-element, though it is not required in the case when the PPML/VDX-Layout file contains none of the PDF pages used as compound element data.

If the **Self** sub-element is present, it must have a string type attribute named ***Src,*** which must be a non-empty string. The URI referenced by this string may or may not be accessible. It is not necessary to verify the correctness of this URI, it is only useful as a mechanism for determining that the Src URI reference of a EXTERNAL_DATA_ARRAY is referring to a compound element of the containing PPML/VDX-Layout file.

Next, all PPML/VDX-Content files that are referenced by the PPML/VDX instance must be verified as accessible. For every unique ***Src*** attribute value of the **EXTERNAL_DATA_ARRAY** elements defined in the PPML data stream or element that refers to a PPML/VDX content file, confirm that there is a corresponding **Self** or **Binding** sub-element in the **ContentBindingTable**. To carry out this procedure, examine the ***Src*** attribute values in each **EXTERNAL_DATA_ARRAY** element and create a list of URI references. For each reference in this list either

— there must be a **Binding** sub-element in the **ContentBindingTable** where the ***Src*** attribute value is an exact match, including case, or

— the ***Src*** value of the Self sub-element in the **ContentBindingTable** is an exact match.

Then, verify the existence of the content file specified by the ***Src*** or ***LocalSrc*** attribute of each **Binding** sub-element in the **ContentBindingTable.** Confirm that the file contents correspond to the value of the ***UniqueID*** or ***BaseID*** attribute, if present, and that its contents also correspond to the value of the ***MD5_Checksum,*** if supplied.

To carry out this procedure, the preflight process must complete the tasks noted below for each **Binding** sub-element in the **ContentBindingTable**.

5.4 Validating Strict PPML/VDX instances

Each **Binding** sub-element in the **ContentBindingTable** must have a ***UniqueID*** value; this ***UniqueID*** value must match the second string element of the ***ID*** array in the trailer dictionary for the file referenced by the ***Src*** value in that **Binding** sub-element. The **Binding** sub-element in the **ContentBindingTable** must also have an ***MD5_Checksum*** value. The calculated MD5 checksum for the contents of the file referenced by that **Binding** sub-element's ***Src*** value must correspond to the ***MD5_Checksum*** value in the **Binding** sub-element. See CGATS.20-2002, 6.12, for more detail.

5.5 Validating Relaxed PPML/VDX instances

Each **Binding** sub-element in the **ContentBindingTable** may have either a ***UniqueID*** value or a ***BaseID*** value. A **Binding** sub-element must not have both a ***UniqueID*** value and a ***BaseID*** value. Use of the ***BaseID*** entry indicates that, though the latest version of a content file is preferred, an earlier revision of the content file is allowed. If the ***BaseID*** attribute is used, it is assumed that communication will take place between sender and receiver to determine the correct file revision before the instance is produced.

If a **Binding** sub-element has a ***UniqueID*** value, the preflight process is obligated to confirm that this ***UniqueID*** value matches the changing unique identifier (the second string element) in the ***ID*** array in the **trailer** dictionary for the file referenced by the **Binding** sub-element. If the **Binding** sub-element has a ***BaseID*** value, the preflight process is obligated to confirm that this ***BaseID*** value matches the value of the permanent unique identifier (the first string element) of the ***ID*** array in the **trailer** dictionary for the file referenced by the **Binding** sub-element.

Each **Binding** sub-element may have an ***MD5_Checksum*** value. If the **Binding** sub-element has an ***MD5_Checksum*** value, the preflight process must calculate the MD5 checksum for the contents of the file referenced by the sub-element and confirm that it matches the supplied ***MD5_Checksum*** value.

A relaxed PPML/VDX instance can be a valid PPML/VDX file set even if one or more of its **Binding** sub-elements has neither a ***UniqueID*** value nor a ***BaseID*** value and even if one or more of its **Binding** sub-elements does not have an ***MD5_Checksum*** value. Nevertheless, it is recommended that all producers of relaxed PPML/VDX instances include suitable values wherever possible.

These values are subject to change during the course of processing a PPML/VDX instance; for example, if a revision of a content file is supplied. Therefore, the final preflight before producing a PPML/VDX file set should not rely on cached values or information from prior preflight processes.

The PPML specification has a provision for the explicit identification of occurrences (known as global occurrences) intended to persist in the RIP's raster cache, allowing the reuse of already cached raster data present in the RIP across PPML/VDX instances as well as within an instance. In PPML/VDX, however, there is no way to explicitly identify occurrences intended to persist in the raster cache across job instances. Specific RIP implementations may implicitly re-use rasterized sources across PPML/VDX instances, thus not requiring the authoring system to explicitly manage the RIP's raster cache.

Requiring the accessibility of all source components of a given PPML/VDX instance supports the independence from device related states. This obviates the possibility of obsolete resource data being used.

5.6 Validation of XML data

The XML data for the **PPMLVDX** element should be examined next. If the **PPMLVDX** element contains a **PPMLRef** element, then the referenced PPML file must be verified as accessible and conforming to the PPML specification and the restrictions given in CGATS.20-2002. Otherwise, the **PPMLVDX** element must contain a **PPML** element, which must be verified in the same manner as the **PPMLRef** element would be. Likewise, if the **PPMLVDX** element contains a **JDFRef** element, then the referenced JDF data stream must be verified as accessible and in conformance to the JDF specification and the restrictions given by the PPML/VDX standard. If the **PPMLVDX** element contains a JDF element, then this must be verified as conforming.

5.7 Determining color correctness

All content and layout files in a strict PPML/VDX instance must have content binding table entries that assert the **IntendedColor** attribute as *true*; files in a relaxed PPML/VDX instance may have entries that also assert this attribute as true but are not required to do so. A preflight tool could validate color in a strict PPML/VDX instance and choose not to in a relaxed instance. Not validating the color introduces the risk of not being able to properly produce the relaxed PPML/VDX instance.

For each entry in the **ContentBindingTable**, if the ***IntendedColor*** attribute is set to *true*, the PDF content or layout file referenced must be checked and verified to be in compliance with the PDF/X-1a or PDF/X-3 standard. The process for verifying that they are PDF/X-1a or PDF/X-3 conforming files can be done using software for this purpose.

6 Cropping and trimming

According to the PDF specification, the value of the **/Page** dictionary's ***/MediaBox*** key specifies the maximum rectangular extent of content that may contain visible marks if a **/CropBox** key is not present. Content marks that fall outside of this region shall be clipped. If a **/CropBox** is present, then content within the ***MediaBox*** rectangle is to be clipped to the common rectangular region formed by the ***/CropBox*** and ***/MediaBox*** rectangles.

In PPML/VDX, the content contained within the common rectangular region defined by the **/CropBox** and ***/MediaBox*** rectangles may be further manipulated and clipped by PPML **TRANSFORM** and **CLIP_RECT** elements.

Regardless of the size and location of the ***/CropBox***, the lower left corner of the ***/MediaBox*** is always the location referenced by the PPML **OBJECT** element's ***Position*** attribute.

The PPML/VDX reader must ignore the ***/BleedBox***, ***/TrimBox***, and ***/ArtBox*** if present in a PDF **/Page** dictionary and thus not treat them as clipping boxes.

The rectangle defined by the ***/TrimBox*** specifies the rectangular extent of the finished page and therefore its content.

Typically, a PDF page designed for print may have additional process oriented content marks (e.g. bleeds, printer's marks, etc.). Such marks may appear in the area between the rectangles defined by the PDF **/Page** dictionary's ***/TrimBox*** and ***/CropBox***, and are physically trimmed away in a post-press operation.

When PDF pages are used as compound elements on a PPML page, any content marks located between the ***/TrimBox*** and ***/CropBox*** will be visible as PPML page content. Including these marks as content on a PPML page is very likely not the intent of the designer.

To prevent the rendering of content marks defined outside of the ***/TrimBox*** rectangle, it is the responsibility of the PPML/VDX generation application to specify an appropriate PPML **CLIP_RECT** to prevent such marks from being imaged. This **CLIP_RECT** should be derived from the PDF page's ***/TrimBox.***

When bleed content marks are defined outside the PDF **/Page** dictionary's ***/TrimBox*** rectangle, and the object occurs on the PPML page as defined by the **PageDesign::*TrimBox*** attribute such that the bleed is still required on one or more edges, then the PPML **CLIP_RECT** should be extended to include the bleed area; i.e. the **CLIP_RECT** should be derived from the ***/BleedBox*** rather than the ***/TrimBox*** on those edges.

It is important to note that if not specified, the ***/TrimBox*** defaults to the ***/CropBox***, and if ***/CropBox*** is not present, it defaults to ***/MediaBox***.

The PPML/VDX standard requires the use of the ***TrimBox*** attribute of the PPML **PAGE_DESIGN** to specify the rectangular dimensions of PPML pages. It is important to note that this PPML ***TrimBox*** attribute is not the same as the PDF **/Page** dictionary's ***/TrimBox***.

7 Use of JDF product intent

In a JDF managed workflow, a print product description can be represented as a hierarchy of JDF product nodes. For example, the highest level, or root, **JDF** product node, may provide details as to how the printed components defined in subordinate **JDF** product sub-nodes are assembled as a finished product. Each leaf node of the tree specifies the details of a structure component of the finished product. Examples of such structure components include the front cover, book block, and back cover structure components of a bound book as depicted in the following illustration:

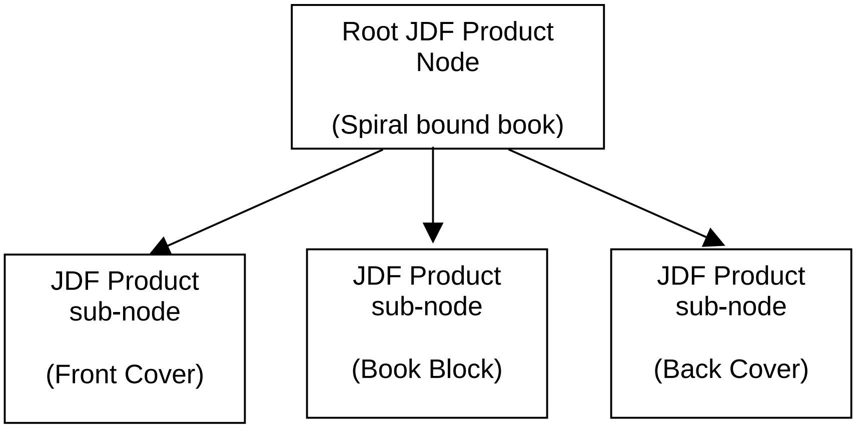


Figure 5 — Example of JDF product notes

Each of the leaf nodes of a JDF product intent hierarchy usually contains a JDF resource element that refers to the graphical page data to be rendered on the finished page(s) it describes. Each such node may also specify the finished page size, media type, and whether or not content is printed on one or both sides of finished sheets. JDF product intent nodes always describe print product components in terms of their physical, finished characteristics in a way that is independent of the process used to manufacture them.

Unlike static print product descriptions, PPML/VDX is able to use JDF product intent data to specify many diverse print products in a single job definition. Each print product of a job may have variations in characteristics such as media, binding style, page layout, and folding requirements. These variations can be specified by the PPML/VDX authoring system, which uses database driven rules to specify those characteristics. These rules are usually applied simultaneously with the application of the database rules used for guiding the composition of customized page content.

The PML/VDX standard permits the use of the PPML **TICKET_REF** element to reference individual JDF product intent resources or partitions of such resources. In this way print product intent characteristics can be logically embedded inline within the PPML hierarchy and thus characterize the product intent context of page definitions. When this is done, the placement of the **TICKET_REF** element in the PPML hierarchy defines the scope of applicability of the product intent resource or partition that it references.

It is important to note that the JDF element for carrying product intent information is optional in PPML/VDX. If, however, a JDF element is present or identified in the **PPMLVDX/ProductIntent** element and no **TICKET_REF** elements are present in the PPML data, the JDF data must be ignored by the reader.

The use of **TICKET_REF** in the PPML data is illustrated in the examples of this section. Note the use of the terms ***page***, ***panel***, and ***sheet*** in the context of these examples have the following meanings:

- A ***page*** is a one sided section of a folded piece defined by the crease of the fold.
- A ***panel*** is a two-sided section of a final folded piece.
- A ***sheet*** is a piece of paper bound into the finished book.

The illustrations below clarify the meanings of these terms. Note that although the illustrations depict a booklet (i.e., a saddle-stitched book), the concepts carry through to other kinds of books, as well (for example, perfect-bound books).

The illustrations below depict an eight-page booklet. The booklet is made up of two ***sheets***. There are two ***panels*** on each sheet, and there are two ***pages*** on each panel (one page on each surface of the panel).

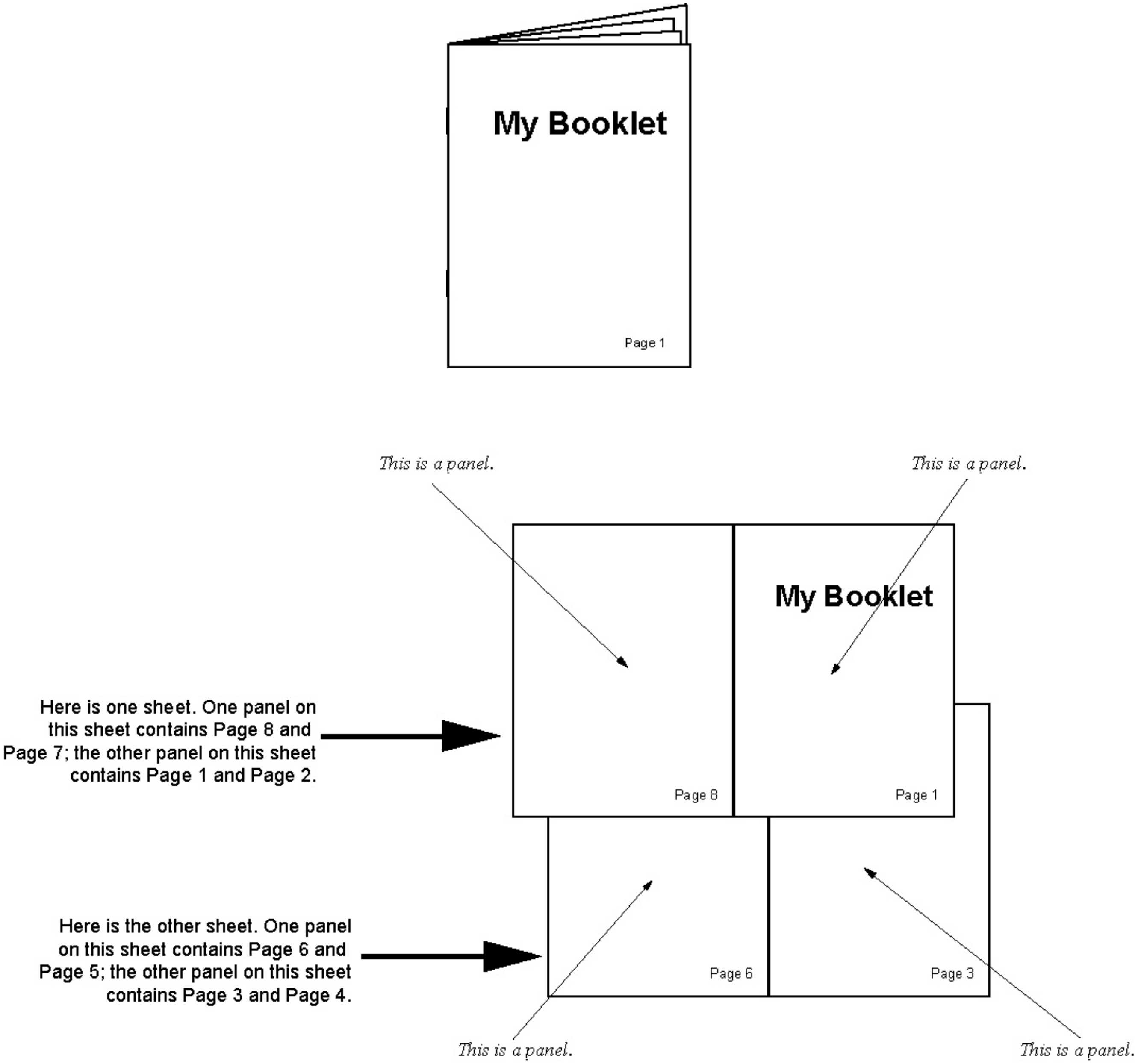
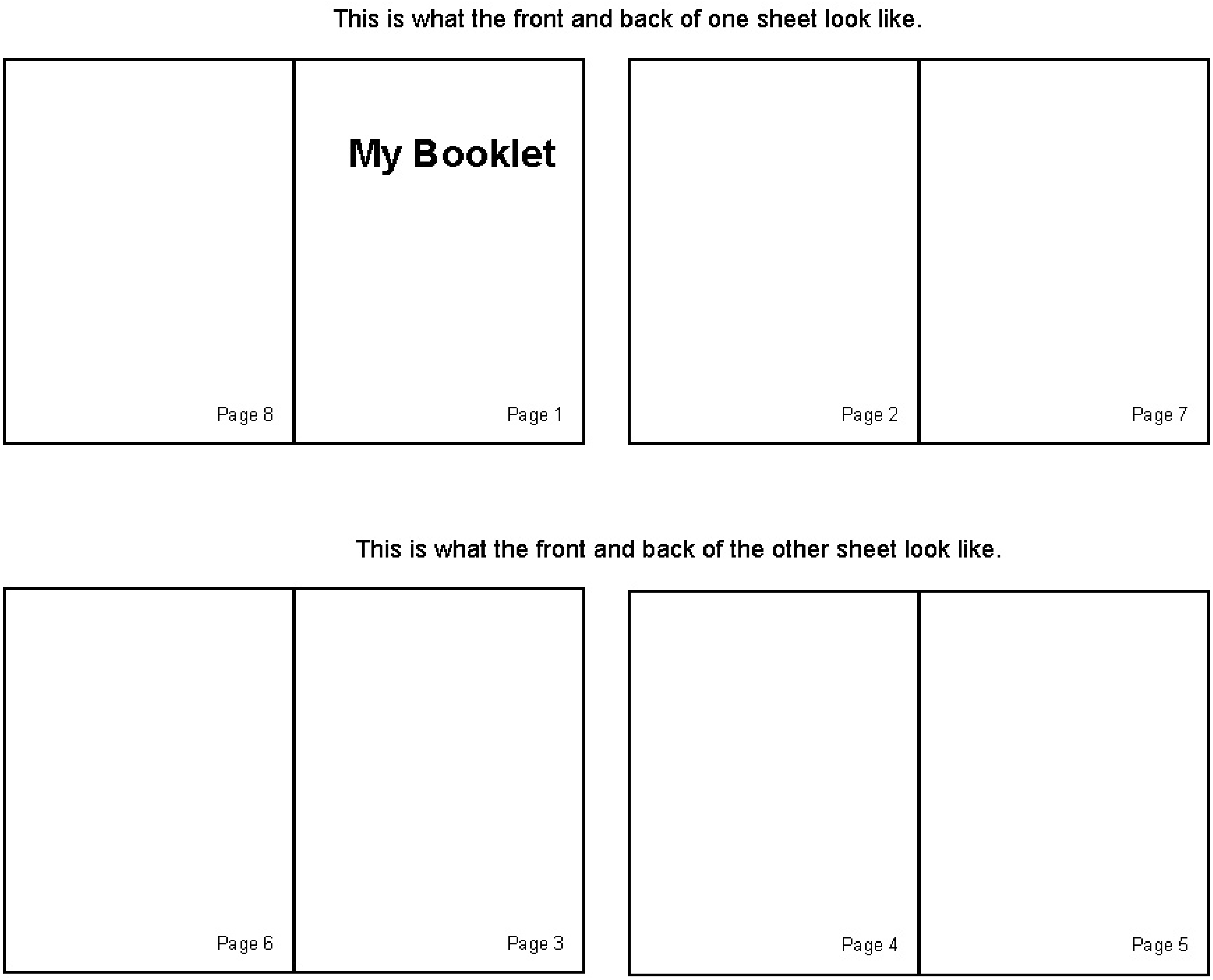

My Booklet
Page 1
This is a panel.
This is a panel.
My Booklet
Here is one sheet. One panel on this sheet contains Page 8 and Page 7; the other panel on this sheet contains Page 1 and Page 2.
Page 8
Page 1
Here is the other sheet. One panel on this sheet contains Page 6 and Page 5; the other panel on this sheet contains Page 3 and Page 4.
Page 6
Page 3
This is a panel.
This is a panel.

Figure 6 — Example of the sheets of a finished booklet


This is what the front and back of one sheet look like.

This is what the front and back of the other sheet look like.

Figure 7 — Examples of front and back sheets

In JDF, the product intent description of a print product has no concept of a *sheet* as defined above. Instead, the description of the print product can be thought of as a bound stack of panels where the sides (pages) of each panel are described in the context of structure components. A content page defined by the PPML **PAGE** element is understood to be the page content that is mapped to a side of a panel.

It is important to note that the concept of a “finished page” as defined in the JDF Specification corresponds to the definition of a “panel” as used in this document.

7.1 Linking PPML data with JDF Product Intent data

There are two ways in PPML/VDX to specify JDF product intent data using JDF and are referred to here as *Inline JDF Intent* and *Referential JDF Intent*.

Inline JDF Intent – refers to the use of the **TICKET_REF** sub-element within the PPML data structure as allowed by CGATS.20. The JDF intent data for this approach shall conform to the JDF Specification and is that which is stored within the **ProductIntent** sub-element of the **PPMLVDX** element as defined by the PPML/VDX standard. This approach is considered ”inline” because **TICKET_REF** elements in the PPML data refer directly to the JDF intent data. Logically this inserts the JDF intent semantics inline, co-mingled with the PPML page data. The actual JDF intent data is a separate resource.

Referential JDF Intent – refers to the use of a separate JDF job ticket instance that is **not** directly linked from within the PPML/VDX structured document page data. This refers to the use of JDF for VDP applications as allowed by the JDF Specification where the JDF node's **RunList** resource refers to the PPML/VDX data, rather than the PPML data's **TICKET_REF** entries referring to the JDF data. This approach involves the use of JDF's ***RunTags*** partition key in the partitioning of product intent resources. This JDF job ticket data is structured differently from the JDF data that may be stored within or referenced from the **ProductIntent** sub-element of the **PPMLVDX** element as defined by the PPML/VDX standard (described here as *Inline JDF Intent*). *Referential JDF Intent* is a separate JDF intent data file that utilizes the JDF **RunList** resource to refer to the **JOB**, and **DOCUMENT** elements present in the PPML data either by name (i.e. **JOB/@*Label***, **DOCUMENT/@*Label***) or by index. This method has the added benefit of being able to describe more complex print products such as the ability to specify the relationship of multiple finished print products defined per recipient instance. In this case, the JDF data structure is represented as a hierarchy of JDF nodes and the intent resource's ***UpdateID*** attributes are not required.

It is important to note that from a JDF process definition perspective it is strongly recommended the PPML/VDX data be linked to the JDF process node (that defines the process that consumes the PPML/VDX data) by a **RunList** resource. This means that in a JDF process context the JDF job ticket that utilize PPML/VDX data uses a referential method where an inline **TICKET_REF** based method in is strongly discouraged. If inline intent is used, it is recommended that the PPML data structuring as described in 7.2.1 be used.

It is important to note that the use of JDF as described in this section emphasizes the *Inline JDF Intent* approach. Additional work describing how to properly structure PPML data in terms of the use of the PPML hierarchy to enable smooth crossover to the *Referential JDF Intent* approach is underway and briefly described in 7.2.1.

From the standpoint of using JDF to specify product intent, when a *Referential JDF Intent* job ticket is present in a PPML/VDX job context, it shall take precedence over any **TICKET_REF** elements and *Inline JDF Intent* data present in the **PPMLVDX** element.

7.2 Inline JDF Intent - Referring to JDF product intent resource data from PPML data

CGATS.20 defines the semantics and contextual relationship between PPML page descriptions and the product intent information it references (refer to 6.10 and Annex D of ANSI CGATS.20). It allows the use of PPML **TICKET_REF** elements to reference JDF intent resources, or partitions of JDF intent resources, to provide complete descriptions of finished print product components.

According to CGATS.20, **TICKET_REF** elements may occur at various levels within the PPML data structure hierarchy where their semantic interpretation depends on a simple inheritance scheme. The lowest level of the hierarchy that **TICKET_REF** elements are permitted to occur is the DOCUMENT element. They cannot occur within **PAGE** elements. If they occur between **PAGE** elements it means that only the **PAGE** elements within the **DOCUMENT** element that follow the **TICKET_REF** element are in scope of the inline product intent semantic it specifies. Some of the examples in 7.2.2 show the use **TICKET_REF** elements at the **DOCUMENT** element level and between **PAGE** elements.

7.2.1 Structured Use of TICKET_REF elements in the PPML data

To simplify conveying product intent semantics using the Inline JDF intent specification, the PPML data can be organized respective of the structure components of the print product described. For example, if a book has front cover, body, and back cover structure components as in figure 5 and there are distinct variations in their product intent descriptions (i.e. media style differs), a separate **DOCUMENT** element within a **JOB** element is defined that specifies the set of pages for each respective structure component. The **JOB** element in this case is used to specify the structure components of print products that belong to a particular recipient (recipient instance). In this structured approach, **TICKET_REF** elements will never occur between **PAGE** elements. Instead, **TICKET_REF** elements that characterize the product intent description of the structure component will occur within the **DOCUMENT** element and before the first **PAGE** element. Since **DOCUMENT** elements then convey the definition of a structure component, if the structure component is that of a bound print product comprising multiple structure components (as shown in figure 5), it is then necessary to specify the **BindingIntent** using a **TICKET_REF** that occurs in the **JOB** element ahead of the first **DOCUMENT** element that defines the first component to be bound.

Another benefit of this approach is that both the **JOB** and **DOCUMENT** elements may be named using their respective ***Label*** attributes. This is an important feature of PPML where in the Referential JDF Intent and process approach (to be described in detail in a document currently being prepared by the CGATS committee) to linking PPML data to JDF, the value of the **DOCUMENT**/@***Label*** attribute corresponds to the value of the JDF **RunList**/@***DocNames*** attribute and also that of the **RunTags** resource partition key. As well, the value of the **JOB**/@***Label*** attribute corresponds to that of the **RunList**/@***SetNames*** attribute.

The following incomplete example JDF and PPML syntax shows the basic structured organization of PPML data for specifying a simple wire comb bound customized print product per recipient instance. The PPML hierarchy of the example shows how separate **DOCUMENT** elements are used to specify the sets of pages particular to a structure component as well as the use of **TICKET_REF** elements in specifying their product intent characterization. For two sided covers, two pages are shown for each the FrontCover and BackCover components, and the finished pages of the Body components are printed two sided in different media than the cover components. The relationship of each structure component definition (**DOCUMENT** element) within each recipient instance (**JOB** element) is specified in terms of **BindingIntent** by the **TICKET_REF** element present in the **JOB** element.

Example:

```
<JDF xmlns=”http://www.CIP4.org/JDFSchema_1” ID=”JT0000” Type=”Product” Status=”Ready”
DescriptiveName=”JDF for PPML/VDX examples”>
…
<ResourcePool>
    <MediaIntent ID=”Medias” Class=”Intent” PartIDKeys=”Option” Status=”Available”>
        <Texture DataType=”EnumerationSpan” Actual=”Smooth”/>
        <StockType DataType=”EnumerationSpan” Actual=”Bond”/>
        <MediaColor DataType=”EnumerationSpan” Actual=”White”/>
        <Grade DataType=”IntegerSpan” Range=”3 4 5” Actual=”5”/>
        <MediaIntent Option=”M1” UpdateID=”120GSMCoatedWhite” Class=”Intent”>
            <FrontCoatings DataType=”EnumerationSpan” Actual=”Glossy”/>
            <BackCoatings DataType=”EnumerationSpan” Actual=”Glossy”/>
            <Weight DataType=”IntegerSpan” Actual=”120”/>
        </MediaIntent>
        <MediaIntent Option=”M2” UpdateID=”100GSMUncoatedWhite” Class=”Intent”/>
            <Weight DataType=”IntegerSpan” Actual=”100”/>
        </MediaIntent>
    </MediaIntent>

    <LayoutIntent ID=”Layout” Class=”Intent” UpdateID=”TwoSidedHeadToHead”
        Sides=”TwoSidedHeadToHead” Status=”Available” />

    <BindingIntent ID=”Binding” Class=”Intent” UpdateID=”WireComb” Status=”Available”/>
        <BindingType DataType=”EnumerationSpan” Actual=”WireCombBinding”/>
        <BindingSide DataType=”EnumerationSpan” Actual=”Left”/>
         <WireCombBinding>
            <WireCombMaterial DataType=”EnumerationSpan” Actual=”Steel-Silver”/>
            <WireCombShape DataType=”EnumerationSpan” Actual=”Twin”/>
         </WireCombBinding>
    </BindingIntent>
```

```
</ResourcePool>
<ResourceLinkPool>
    <MediaIntentLink rRef=”Medias” Usage=”Input”/>
    <LayoutIntentLink rRef=”Layout” Usage=”Input”/>
    <BindingIntentLink rRef=”Binding” Usage=”Input”/>
</ResourceLinkPool>
</JDF>


<PPML …>
    …
    <TICKET_REF ExtIDRef=”TwoSidedHeadToHead”/>
    <JOB Label=”Recipient0001”>
        <TICKET_REF ExtIDRef=”WireComb”/>
        <DOCUMENT Label=”FrontCover”>
            <TICKET_REF ExtIDRef=”120GSMCoatedWhite”/>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
        </DOCUMENT>
        <DOCUMENT Label=”Body”>
            <TICKET_REF ExtIDRef=”100GSMUncoatedWhite “/>
            <PAGE …> … </PAGE>
            …
            <PAGE …> … </PAGE>
        </DOCUMENT>
        <DOCUMENT Label=”BackCover”>
            <TICKET_REF ExtIDRef=”120GSMCoatedWhite”/>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
        </DOCUMENT>
    </JOB>
    …
    <JOB Label=”Recipient2384”>
        <TICKET_REF ExtIDRef=”WireComb”/>
        <DOCUMENT Label=”FrontCover”>
            <TICKET_REF ExtIDRef=”120GSMCoatedWhitec/>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
        </DOCUMENT>
        <DOCUMENT Label=”Body”>
            <TICKET_REF ExtIDRef=”100GSMUncoatedWhite”/>
            <PAGE …> … </PAGE>
            …
            <PAGE …> … </PAGE>
        </DOCUMENT>
        <DOCUMENT Label=”BackCover”>
            <TICKET_REF ExtIDRef=”120GSMCoatedWhite”/>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
        </DOCUMENT>
    </JOB>
</PPML>
```

7.2.2 Generic Use TICKET_REF Elements in the PPML data

As allowed by CGATS.20, **TICKET_REF** elements may occur in the **PPML**, **JOB**, and **DOCUMENT** element only. This section illustrates the semantics of the use of **TICKET_REF** and JDF through the use of a series of examples.

Example 1:

This example shows a PPML element with no inline product intent information (no **TICKET_REF** elements are present in the **PPML** element). This means that the relationship of the **PAGE**, **DOCUMENT**, and **JOB** sub-elements to finished print products is undefined. The interpretation of these sub-elements must be conveyed by information separate from the PPML/VDX instance, as in the *Referential JDF Intent* approach.

Such a PPML/VDX instance is similar in concept to a set of pages defined by a typical PDF file where, by itself, no information is provided as to how the pages are applied in the production of a print product.

```
<PPML …>
    …
    <JOB …>
        <DOCUMENT …>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
        </DOCUMENT>
    </JOB>
    <JOB …>
        <DOCUMENT …>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
        </DOCUMENT>
    </JOB>
</PPML>
```

In the following examples, the use of the PPML **TICKET_REF** element is introduced. In example 2, JDF intent data defines the product intent resources referred to by the **TICKET_REF** elements in the PPML data in examples 3 through 8 that follow.

Example 2:

```
<JDF xmlns="http://www.CIP4.org/JDFSchema_1" ID="JT0001" Type="Product" Status="Ready"
DescriptiveName="JDF for PPML/VDX examples">
<ResourcePool>
    <MediaIntent ID="Medias" Class="Intent" PartIDKeys="Option" Status="Available">
        <Texture DataType="EnumerationSpan" Preferred="Smooth"/>
        <StockType DataType="EnumerationSpan" Preferred="Bond"/>
        <MediaColor DataType="EnumerationSpan" Preferred="White"/>
        <Grade DataType="IntegerSpan" Range="3 4 5" Preferred="5"/>
        <MediaIntent Option="M1" UpdateID="120GSMCoatedWhite" Class="Intent">
            <FrontCoatings DataType="EnumerationSpan" Preferred="Glossy"/>
            <BackCoatings DataType="EnumerationSpan" Preferred="Glossy"/>
            <Weight DataType="IntegerSpan" Preferred="120"/>
        </MediaIntent>
        <MediaIntent Option="M2" UpdateID="100GSMUncoatedWhite" Class="Intent"/>
            <Weight DataType="IntegerSpan" Preferred="100"/>
        </MediaIntent>
    </MediaIntent>

    <LayoutIntent ID="Layouts" Class="Intent" UpdateID="TwoSidedHeadToHead"
        Sides="TwoSidedHeadToHead" Status="Available" />

    <BindingIntent ID="Bindings" Class="Intent" PartIDKeys="Option" Status="Available"/>
        <BindingIntent UpdateID="SaddleStitchLeft">
            <BindingType DataType="EnumerationSpan" Preferred="SaddleStitch"/>
            <BindingSide DataType="EnumerationSpan" Preferred="Left"/>
         <SaddleStitching>
               <StitchNumber DataType="IntegerSpan" Options="2 3" Preferred="3"/>
         </SaddleStitching>
        </BindingIntent>
        <BindingIntent UpdateID="SaddleStitchTop">
            <BindingType DataType="EnumerationSpan" Preferred="SaddleStitch"/>
            <BindingSide DataType="EnumerationSpan" Preferred="Top"/>
         <SaddleStitching>
                <StitchNumber DataType="IntegerSpan" Options="2 3" Preferred="3"/>
        </SaddleStitching>
        </BindingIntent>
    </BindingIntent>
</ResourcePool>
<ResourceLinkPool>
    <MediaIntentLink rRef="Medias" Usage="Input"/>
    <LayoutIntentLink rRef="Layouts" Usage="Input"/>
    <BindingIntentLink rRef="Bindings" Usage="Input"/>
</ResourceLinkPool>
</JDF>
```

Example 3:

In this example, **TICKET_REF** elements have been added after each **DOCUMENT** element start tag. These **TICKET_REF** elements each refer to the same partition of a JDF **BindingIntent** resource that specifies saddle stitching and also specifies that it is the left edge side that is to be bound. In this example, the set of **PAGE** elements defined within each **DOCUMENT** element are to be saddle stitched. Note that exactly how the pages are to be mapped to the surfaces of the finished pages is not specified so the page content may therefore be printed single or two sided according to production system settings.

```
<PPML …>
    …
    <JOB …>
        <DOCUMENT …>
            <TICKET_REF ExtIDRef="SaddleStitchLeft"/>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
            …
        </DOCUMENT>
    </JOB>
    <JOB …>
        <DOCUMENT …>
            <TICKET_REF ExtIDRef="SaddleStitchLeft"/>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
            <PAGE …> … </PAGE>
            …
        </DOCUMENT>
    </JOB>
</PPML>
```

Example 4:

In this example, a **TICKET_REF** element has been added to indicate that the pages of each **DOCUMENT** element to be saddle stitched and are to be printed on both sides beginning with the contents of the first **PAGE** being mapped to the front side of the first finished sheet. This **TICKET_REF** element refers to a JDF **LayoutIntent** element that specifies two sided head to head printing. The resulting finished print products for this example will be two 8 page saddle stitched booklets each with 4 two-sided finished sheets. Each sheet has two panels, and each panel has two pages.

```
<PPML …>
    …
    <TICKET_REF ref="TwoSidedHeadToHead"/>
    <JOB …>
        <DOCUMENT …>
            <TICKET_REF ExtIDRef="SaddleStitchLeft"/>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
    </JOB>
    <JOB …>
        <DOCUMENT …>
            <TICKET_REF ExtIDRef="SaddleStitchLeft"/>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
    </JOB>
</PPML>
```

Example 5:

In this example, a **TICKET_REF** was added which refers to a partition of the JDF **MediaIntent** resource that specifies 100 grams per square meter (gsm) white paper coated on two sides. In this example, all of the pages of the two saddle stitched, two sided print products are to be printed on 100 gsm white coated stock.

```
<PPML …>
    …
    <TICKET_REF ref="100GSMCoatedWhite"/>
    <TICKET_REF ref="TwoSidedHeadToHead"/>
    <JOB …>
        <DOCUMENT …>
            <TICKET_REF ExtIDRef="SaddleStitchLeft"/>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
    </JOB>
    <JOB …>
        <DOCUMENT …>
            <TICKET_REF ExtIDRef="SaddleStitchLeft"/>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
    </JOB>
</PPML>
```

Example 6:

In this example, **TICKET_REF** elements have been added that refer to JDF **MediaIntent** partitions that specify 120 gsm coated white paper. This example specifies three bound booklets:

1. The finished booklet described by the first **JOB** element is a 12-page (six finished panels) saddle-stitched booklet with a first page of 120 GSM coated white stock. The front surface of the first panel is imaged with the content of the first **PAGE** element and the backside of the first panel is left blank due to the presence of the **TICKET_REF** following the first **PAGE** sub-element (refer to clause **6.10** of CGATS.20). The second through fourth leaves are printed on 100 GSM uncoated white paper as specified by the **TICKET_REF** element, and are printed on both sides. The fifth panel is also printed on 100 GSM uncoated white paper, but only printed on its front side due to the presence of the **TICKET_REF** following the eighth **PAGE** element. The sixth panel, like the first, is 120 GSM coated white paper. It has a blank front due to the presence of a blank **<PAGE/>** element, and its back is imaged (it is logically the back cover of the booklet).

2. The finished booklet described in the second **JOB** element is a 10-page (five-panel) saddle-stitched booklet. It is important to note that the number of panels is not an even multiple of two; this presents special finishing requirements.

3. Faithfully meeting the exact requirements of such an intent description must be weighed against the feasibility of producing this saddle-stitched booklet by simply adding an extra panel that is blank on both sides. Simply adding an additional panel not represented in the PPML/VDX data violates the product intent description and should not be done without prior agreement with the print specifier.

4. The finished booklet described in the third **JOB** element is almost the same as the one described in the second **JOB** element; the difference is that two blank pages have been added in the third **JOB** element. It represents a 12-page (six-panel) saddle-stitched booklet. This example has additional pad pages added with a common media type so that it is possible to saddle stitch. In this example, the finished booklet has a fifth panel with both front and back pages blank.

```
<PPML …>
    …
    <TICKET_REF ref="120GSMCoatedWhite"/>
    <TICKET_REF ref="TwoSidedHeadToHead"/>
    <JOB …>
        <DOCUMENT …>
            <TICKET_REF ExtIDRef="SaddleStitchLeft"/>
            <PAGE> … </PAGE>
            <TICKET_REF ExtIDRef="100GSMUncoatedWhite"/>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <TICKET_REF ExtIDRef="120GSMCoatedWhite"/>
            <PAGE/>
            <PAGE> … </PAGE>
        </DOCUMENT>
    </JOB>
    <JOB …>
        <DOCUMENT …>
            <TICKET_REF ExtIDRef="SaddleStitchLeft"/>
            <PAGE> … </PAGE>
            <TICKET_REF ExtIDRef="100GSMUncoatedWhite"/>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <TICKET_REF ExtIDRef="120GSMCoatedWhite"/>
            <PAGE/>
            <PAGE> … </PAGE>
        </DOCUMENT>
    </JOB>
    <JOB …>
        <DOCUMENT …>
```

```
            <TICKET_REF ExtIDRef="SaddleStitchLeft"/>
            <PAGE> … </PAGE>
            <TICKET_REF ExtIDRef="100GSMUncoatedWhite"/>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE/>
            <PAGE/>
            <TICKET_REF ExtIDRef="120GSMCoatedWhite"/>
            <PAGE/>
            <PAGE> … </PAGE>
        </DOCUMENT>
    </JOB>
</PPML>
```

Example 7:

In this example, the sets of pages defined by the two **DOCUMENT** elements are all intended to be saddle stitched together into a single 16-page, eight-panel print product. This example demonstrates how **PAGE**, **DOCUMENT**, and **JOB** sub-elements of the PPML hierarchy carries no semantics regarding which pages are to be bound together as finished components; it only expresses a reader-order set of related pages. Therefore, in PPML/VDX, the **DOCUMENT** and **JOB** elements simply represents a convenient partitioning of sets of **PAGE** elements as a stack oriented scope mechanism.

```
<PPML …>
    <TICKET_REF ExtIDRef="120GSMCoatedWhite"/>
    <TICKET_REF ExtIDRef="SaddleStitchTop"/>
    <TICKET_REF ExtIDRef="TwoSidedHeadToHead"/>
    <JOB …>
        <DOCUMENT …>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
    </JOB>
    <JOB …>
        <DOCUMENT …>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
    </JOB>
</PPML>
```

7.3 Recommended use of the PPML hierarchy in PPML/VDX for complex jobs

As can be seen in the examples of the previous section, that in order to specify how the PPML pages are related to the finished print product in the context of Inline JDF Intent, it is necessary to overlay a simple product intent semantic model on top of the PPML content model using the PPML **TICKET_REF** element. Using this model, it is unambiguous as to exactly which panels, or finished sheets are to be bound together into a set of bound panels, and on which side of a panel the content defined by a **PAGE** element is to be imaged. As was seen in *example 7*, it is made very clear that because a set of **PAGE** elements occur within a **DOCUMENT** element doesn't mean that only those pages are to be bound together.

The structured approach described in 7.2.1 utilizes the **DOCUMENT** element as the way of specifying related sets of pages that have a common product intent characterization as a structured component. Innthis case **TICKET_REF** elements do not occur between **PAGE** elements and the **TICKET_REF** that specifies the binding style occurs in the JOB element before or between **DOCUMENT** elements. This structured approach is more intuitive because it better organizes the semantic model and correlates more naturally to the hierarchical JDF node hierarchy in the case of the Referential JDF Intent or process model.

Given the nature of VDP jobs that specify multiple variable or versioned print components per recipient instance, it is necessary to provide a mechanism that correlates each of the components intended for a given recipient. Hence it is important that each component be identified to its targeted recipient. For example, if a particular recipient is to receive an envelope containing three (3) variable finished components, such as a personalized cover letter, personalized return post card, and a custom color brochure, then several options exist for representing them in a VDP job that uses PPML/VDX:

1. Define each component in separate PPML/VDX instances for a total of three PPML/VDX instances.

2. Aggregate all three components into a single PPML/VDX instance.

3. Aggregate two of the components in a first PPML/VDX instance, and represent the third component in a second PPML/VDX instance.

The first option suggests a unique PPML/VDX instance for each component. This is a relatively straightforward approach; however, it is necessary to guarantee that the print components represented by each can be reliably correlated. Therefore, the recipient ordering of components for each PPML/VDX instance should align across the PPML/VDX instances.

The second option suggests that all recipient-instance components are defined in a single PPML/VDX instance. As with the first option, it is necessary to guarantee that the print components belonging to each recipient can be reliably correlated.

The third option is simply a combination of the first and second options.

7.4 Aggregating multiple recipient components in a single PPML/VDX instance

In the PPML data, there may be three sets of pages, each intended as the content for the pages of each independently finished component. Of course, given the **PPML**/**JOB**/ **DOCUMENT** /**PAGE** hierarchy of PPML, and the permitted use of **TICKET_REF,** there are numerous ways in which these multiple print components may be defined. A preferred way would be to use the PPML **JOB** element as the container within which the finished components for a given recipient are defined. For example, every three successive **DOCUMENT** sub-elements of the single **JOB** element would define the set of finished components for a recipient.

Example 8:

This example shows the use of the PPML element hierarchy where each **JOB** element contains the definition of the variable components for a given recipient. The **JOB** element's ***Label*** attribute has the value of the recipient identifier taken from the original recipient database. The ***Label*** attribute of each **DOCUMENT** attribute has a value that identifies the particular component.

It is important to note that **TICKET_REF** elements referring to a binding style must not be specified outside of the **JOB** element since it is not proper to bind together printed components which target multiple recipients. It is also preferred that **TICKET_REF** elements referring to binding style always be specified within the **DOCUMENT** element so that the **DOCUMENT** element receives the semantic of encapsulating all of the pages for the print component.

```
<PPML>
    …
    <TICKET_REF ExtIDRef="120GSMUncoatedWhite"/>
    <TICKET_REF ExtIDRef="TwoSidedHeadToHead"/>
    <JOB Label="R0001">
        <DOCUMENT Label="CoverLetter">
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
        <DOCUMENT Label="PostCard">
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
        <DOCUMENT Label="Brochure">
            <TICKET_REF ref="SaddleStitchTop"/>
            <TICKET_REF ref="120GSMCoatedWhite"/>
            <PAGE …> … </PAGE>
            . . .
            <PAGE …> … </PAGE>
        </DOCUMENT>
    </JOB>
    <JOB Label="R0002">
        <DOCUMENT Label="CoverLetter">
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
        <DOCUMENT Label="PostCard">
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
        <DOCUMENT Label="Brochure">
            <TICKET_REF ref="SaddleStitchTop"/>
            <TICKET_REF ref="120GSMCoatedWhite"/>
            <PAGE …> … </PAGE>
            . . .
            <PAGE …> … </PAGE>
        </DOCUMENT>
    </JOB>
    <JOB Label="R0003">
        <DOCUMENT Label="CoverLetter">
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
        <DOCUMENT Label="PostCard">
            <PAGE> … </PAGE>
            <PAGE> … </PAGE>
        </DOCUMENT>
        <DOCUMENT Label="Brochure">
            <TICKET_REF ExtIDRef="SaddleStitchTop"/>
            <TICKET_REF ExtIDRef="120GSMCoatedWhite"/>
            <PAGE …> … </PAGE>
            . . .
            <PAGE …> … </PAGE>
        </DOCUMENT>
    </JOB>
</PPML>
```

Annex A

Converting PPML/VDX instances to PPML file sets

Users of current PPML workflows may wish to extract the PPML data stream from a PPML/VDX instance and convert PPML/VDX resource references to references suitable for use with the extracted PPML. We name such a process a PPML/VDX *unpacking tool*. This can be thought of as an adaptor tool, not an "unpacking" tool, because there is nothing special a PPML consumer needs to do differently in terms of resolving PPML external content references.

This section outlines the procedure that an unpacking tool should carry out to locate the PPML data used by a PPML/VDX instance, extract the PPML data to a separate file or location if necessary, and modify the filename references in the PPML as required for the recipient's environment. In addition to performing the basic tasks described in this document, a well-designed unpacking tool should also perform preflight steps on the PPML/VDX instance as described elsewhere in this application note.

It should not be necessary to modify the URIs of the EXTERNAL_DATA_ARRAY elements. The point here is to target a PPML/VDX data set to a PPML/GA device interface. URIs in EDA are independent of packaging in the case of PPML data embedding w/in PDF file.

The steps an unpacking tool should carry out are:

1. Determine the conformance level of the PPML/VDX instance.

   To do this, the unpacking software must locate the **Info** dictionary in the PPML/VDX-Layout file and read the value of the ***GTS_PPMLVDXConformance*** entry.

   If the value is *PPML/VDX-Strict:2002,* the unpacking software should expect to find an unencrypted XML **PPMLVDX** data structure inside the PPML/VDX-Layout file. If the value is *PPML/VDX-Relaxed:2002,* the **PPMLVDX** XML data stream may either be encrypted or not. In either case, this stream may be compressed whereby an unpacking tool must be able to decompress this entry.

   If the value is not present or has a value different from those described above, then the PPML/VDX instance is invalid. Such a file cannot be expected to contain valid PPML data.

2. Read the value of the **GTS_PPMLVDXData** entry in the PPML/VDX layout file's **Catalog** dictionary.

   The PPML/VDX standard requires that the ***GTS_PPMLVDXData*** value in the **Catalog** dictionary must be an indirect reference to the stream object containing the **PPMLVDX** element. If this entry is not present, the file is not a valid PPML/VDX-Layout file

3. Use the indirect reference obtained in Step 2 to look in the PPML/VDX file's cross-reference table and find the location (the absolute file offset) of the stream object that contains the **PPMLVDX** element.

4. Within the stream object at the file offset obtained in Step 3, find the **PPMLVDX** element.

5. Within the **PPMLVDX** element, find the **Layout** sub-element.

6. Examine the Layout element to determine whether it contains a **PPML** element or a **PPMLRef** element. It must contain either of these and not both.

If the conformance level of the PPML/VDX instance is *PPML/VDX-Strict:2002*, the **Layout** element *must* contain a **PPML** element rather than a **PPMLRef** element. If a relaxed instance contains a **Layout** element that contains a **PPMLRef** element, verify the existence of the PPML file referred to by the **PPMLRef** element's ***Src*** attribute. In either case, resolve the URI of the ***Src*** attribute and obtain the PPML data file.

7. Parse the PPML element and modify the values of the ***Src*** attribute of **EXTERNAL_DATA_ARRAY** elements in the PPML file as required for the PPML/VDX recipient's environment.

   Using the information in Appendix D of the *PPML Functional Specification*, Version 2, as a guide, create valid URI references to PPML/VDX layout and content PDF files. These will reference the files under the control of the receiver of the PPML/VDX instance.

The PPML/VDX-Layout file is valid as a PDF file to be referenced, as are the content files of the PPML/VDX instance. Use the steps described in the section on preflighting as a guide to resolving URI references that are not immediately resolvable. If MD5 checksums are provided in the **ContentBindingTable/Binding** elements for referenced content files, these may be used by the unpacking tool to either verify that correct files are referenced, or as a method to derive correct URI references for ones that are not immediately resolvable.