Kodak NexPress Developer's Interface Guide

For System 18.0

Guide Specification

Version 18.0

March 2018

LEGAL

Eastman Kodak Company

2400 Mt. Read Blvd

Rochester, NY 14650 USA

Table of Contents

1 Introduction

The NexPress digital production color press is designed to handle a wide range of applications from short-run and quick turnaround, to variable data printing. It uses an open and standards-compliant architecture to integrate existing software applications. The NexPress digital production color press is driven by the NexPress front end controller. The NexPress front end is a scalable and open PDF-based product. It supports hot folder and JDF/JMF-based job submission and JMF-based job and press management.

Some of the key features of the NexPress front end are:

- Built upon Adobe® PDF-based workflow architecture
- Scalable hardware
- Accepts Job Definition Format (JDF) Job Tickets
- Supports hot folder submission
- Supports common static page description language (PDL) formats including Adobe® PDF, Adobe® PostScript.
- Provides JDF/JMF interface for Job Ticket submission, queue management, and status messaging. (Submission of a JDF Job Ticket can reference either PDF or Postscript content.)
- Supports variable data printing using ANSI Standard PPML/VDX.

1.1 Scope

This document is one of two used by third party software developers who design software controllers and output drivers that target the NexPress family of digital production color presses. The complete set of documents is:

- Kodak NexPress Developer's Interface Guide [KNDIG] (This document)
- Kodak NexPress Developer's Interface Reference [KNDIR]

The **Kodak NexPress Developer's Interface Guide** is the main user document. It references the [KNDIR], and provides user description of the various job submission, job management and press management operations for the NexPress front end. This guide also identifies the various content formats for the NexPress products, and describes the job submission methods and control interfaces using these content formats. Detail on the specific formats is provided in the [KNDIR].

The **Kodak NexPress Developer's Interface Reference** provides technical detail on interfaces used in JDF and PPML/VDX submission and in control of the JDF portal. This technical companion document to the [KNDIG] is intended as a reference. The syntax and semantics of Device Capabilities, JDF Intent, JDF Process, JMF and PPML/VDX used by the NexPress front end is provided within this reference document.

These documents do not describe the interface between the NexPress front end and print engine. This interface is not intended to be accessible to third-party workflow tool developers.

It is assumed that readers of this document have a working understanding of the Job Definition Format (i.e. JDF), the Adobe® Portable Document Format (i.e. PDF), and the PPML/VDX standard.

1.2 Glossary of Terms and Abbreviations

The table below explains terms and concepts used throughout this document, which may or may not be unique to the system environment described in this document.

Term	Definition
CGATS	Committee for Graphic Arts Technologies Standards – an ANSI accredited standards body that develops technical standards for the graphic arts industry. http://www.NPES.org
CIP4	International Cooperation for Integration of Processes in Prepress, Press, and Postpress. The consortium responsible for creating and maintaining the JDF Specifications. http://www.CIP4.org

Term	Definition
Combined Process node	A JDF processes node definition that is an aggregate of several JDF processes. Such combined JDF process node definitions serve to model multiple function devices. See NexPress Combined Process Node.
Component	Various versions or parts of semi-finished print products.
Device Capabilities; Device Capabilities File	The Device Capabilities file is a proprietary XML data file stored in the NexPress front end. It is named DevCaps.xml. This file provides a mechanism to inform client software applications of the system features and resources available on the *NexPress digital production color press*. Features include items such as output sheet delivery and finishing capabilities that are dependent on the press configuration. Resources include a complete characterization of all qualified substrates and their names, lists of installed fonts and ICC color profiles, and the names of defined Virtual Printer Hot Folders.
DFE	Digital Front End — The device that processes a print job (including impositioning and Ripping) and sends rasterized print data to the NexPress print engine for imaging onto media. It is referred throughout this document as the Kodak NexPress front end.
DIG	Developer's Interface Guide — A NexPress document that specifies the use of the various interfaces for submitting jobs and managing the press. Provides technical information for third-party application developers to successfully integrate their solution with the *Kodak NexPress digital production color press*.
Document	A meaningful unit of information. It may be represented in various ways including as a file, as a part of a file, or by printed output from a job.
DTD	Data Type Dictionary
Hot Folder	A hot folder is a file system directory that is monitored by a software process. Files placed into a hot folder are treated as input data and are processed by the software monitoring the hot folder. JDF uses this mechanism for file-based JDF job ticket submission.
Internal Job Queue	NexPress Job Queue containing jobs submitted through both the JDF Portal interface and Virtual Printer Hot Folders.
JDF	The Job Definition Format (JDF) is an evolving industry standard created and maintained by the International Cooperation for the Integration of Processes in Prepress, Press, and Postpress (CIP4). JDF is an extensible, XML–based format built on existing technologies such as the Portable Job Ticket Format (PJTF) from Adobe® and on the Print Production Format (PPF) from CIP3. JDF provides a container for a universal electronic job ticket for all kinds of print production data. A JDF structure can describe both the print product to be produced (intent) and the steps required to produce it (processes). The NexPress front end only consumes JDF process.
JDF device	As defined in the JDF Specification, a JDF device interprets JDF data, identifies a compatible process node definition, and employs a machine that executes it appropriately.
JDF job ticket	An XML data structure that specifies a print job. It may include both a print product description (product intent) and process definition (JDF Process). A JDF job ticket is comprised of one or more JDF nodes. Submission through the NexPress JDF Portal requires a JDF job ticket containing a single JDF process node.
JDF Portal	The NexPress front end's job submission interface that conforms to the JDF Specification and accepts JDF job tickets either by file-based submission to a JDF Hot Folder, or by a SubmitQueueEntry JMF message.
JDF Portal Job Queue	Queue containing only jobs submitted through the JDF Portal interface.

Term	Definition
JDF process	A JDF node that specifies a unit of work or a step in the job workflow. Stitching, DigitalPrinting, Imposition, and LayoutPreparation are examples of JDF Processes. A JDF Process node often specifies the unit of work to be done by a JDF Device. The NexPress front end only supports a single combined process node that lists all the required JDF processes.
JDF resources	JDF elements (such as Media, MediaIntent, or Component) that are input to or output from a JDF process or product intent node. JDF resources are always defined as sub-elements of the ResourcePool. The JDF device may consume physical resources that are input to JDF processes (e.g., substrate described by a Media resource input to a digital printing process).
JMF	Job Messaging Format — JDF Specification: *"JMF is a subset of JDF that handles communication among JDF controllers and equipment on the shop floor…. JMF can be used to establish a queue, discover the capabilities of a JDF-enabled device, determine the status of a device, e.g., "RIPing", "Idle", and so on."*
Job	A unit of work that can processed separately. A job within a printing system may be a part of a larger job from the perspective of a print shop. JDF specifies what needs to be produced in a job and how it is to be accomplished. Typically, a job consists of work done on one or more files, using a set of resources, to produce output.
Media Catalog	A data resource managed by the NexPress front end that contains a profile entry for each substrate, or medium, supported by this *NexPress digital production color press*. Each profile entry describes physical characteristics of a supported medium, and the medium name.
NCF	NexPress Capabilities Format. See also: Device Capabilities
NexPress digital production color press	The Kodak NexPress print engine and front end (DFE).
NexPress print engine	The printing device component of the NexPress digital production color press
NexPress Combined Process node	A JDF-Combined Process node (value of the ***Types*** attribute is *Combined*) that conforms to the restrictions identified in the ***Kodak NexPress Developer's Interface Reference.*** The *NexPress digital production color press* Combined Process Node includes LayoutPreparation, Imposition, ColorSpaceConversion, Interpreting, Rendering, Screening, and DigitalPrinting. Beginning with Release 10.1, only DigitalPrinting is required in JDF/@Types, other parameters will use default settings if not specified.
NexPress Conforming Content Files	Content files assembled from composite pages and specified in reader order, including Adobe® PDF documents for static jobs and ANSI CGATS.20 (PPML/VDX) for variable data jobs. NexPress Conforming Content Files may also represent a sequence of pre-imposed sheet surfaces for both static PDF jobs and variable PPML/VDX jobs; each PDF or PPML page definition is a pre-imposed layout to be printed 1-up onto a sheet surface.
NexPress front end	The DFE processor of the *NexPress digital production color press*
Page	A unit of content that is normally imaged onto a rectangular area of a surface of an output medium. The page is normally the unit of output to which medium selection, imposition, and a variety of other output processing options are applied.
PDF	Adobe® Portable Document Format
PDF/VT	Portable Document Format / Variable Transactional
PDL	Page Description Language (i.e. Adobe® PostScript).

Term	Definition
PDM (VDX)	*Portable Digital Master version 2.01* – an early draft of what is now the fully balloted and accredited ANSI CGATS.20-2002 PPML/VDX standard. This is the optimized Structured Document Format used by earlier versions of the NexPress front end for representing the page content of a VDP job. Like PPML/VDX, this also is a PDF-based format that contains PPML data and is used to encode the many Recipient Instances of a VDP job in a highly optimized way. PDM support by the NexPress front end will be deprecated in the future.
PJTF	Portable Job Ticket Format — Developed by Adobe®, this is is the internal data format for representing job ticket control information in the NexPress front end. JDF job tickets are converted to PJTF upon submission.
PPML	Personalized Print Markup Language — Name of the XML-based variable data mark-up language developed by the Print On Demand initiative (PODi). PPML provides semantics for object-level reusability of graphical content. The PDF-based PPML/VDX and PDM formats use PPML data to describe Instance documents and their pages.
PPML/VDX	Personalized Print Markup Language/Variable Data eXchange — The CGATS.20 ANSI standard PDF and PPML-based file format for representing variable content documents. A Structured Document Format that may be device and workflow-independent. It separates the variable data creative design process from the details of the print manufacturing process. This standard was accredited after the first release of the NexPress digital production color press. PDM is closely related to this standard. Like PDM, it is PDF-based and uses PPML data to describe Recipient Instances and their pages. PPML data is embedded within one of the PDF files of the set files that comprise an Instance.
Recipient Instance	The pages of a VDP content data stream that belong to a particular recipient. Represented in the PPML data of a PPML/VDX instance as a JOB element.
RIP	Raster Image Processor – converts vector and sampled image data interpreted from page content formats such as PDF, PostScript, and PPML/VDX into raster image data that is used to drive the print imaging apparatus of the digital printer.
SDK	Software Developer's Kit — A *collection of documentation and tools intended to enable* third party application developers to successfully integrate their variable data solutions (using ANSI PPML/VDX) with the *NexPress digital production color press*.
Sheet	A unit of a medium, typically a sheet of paper or an area on a continuous roll of roll-fed paper, on which output is printed. Sheets may be cut and folded after printing. For example, a sheet with no folds may contain one or two Pages, one per imaged side; a brochure with one fold may contains four Pages, one located on each side of the fold per side.
Structure component	A partial component of a print product such as the cover pages, body pages, dust cover, or insert pages of a book. Note that a book may be an assembly of structure components.
Structured Document Format	Page content definition format that is structured such that the data that makes up graphical page definitions are defined independent of each other (e.g. Adobe® PDF and ANSI PPML/VDX). The pages of a document using Structured Document Format may be indexed and the data can be efficiently accessed in random fashion.
URI	Uniform Resource Identifier as defined in RFC 2396: Uniform Resource Identifiers (URI)
URL	Uniform Resource Locator as defined in RFC 2396: Uniform Resource Identifiers (URI)
VDP	Variable Data Printing — The concept of customizing each print (or Instance) of a run of a digital print job.

Term	Definition
VDX	Short term for PPML/VDX. See PPML/VDX
Virtual Printer Hot Folder	An active operating system folder that provides a mechanism to submit print job data for execution by the NexPress front end. Each Virtual Printer Hot Folder has a pre-assigned Virtual Printer Job Ticket that specifies processing requirements for the PDF, PDM2, PostScript, or PPML/VDX data submitted into it.
VPJTT	Virtual Printer Job Ticket Template – Specifies settings required by NexPress front end to process content files submitted into a Virtual Printer Hot Folder. The NexPress Client software is used to specify its settings.
XML	Extensible Markup Language

1.3 Document Conventions

This section identifies the standard conventions used in this document. Following these conventions will help ensure consistency throughout the document.

1) PDF operators, PDF keywords, the names of keys in PDF dictionaries, and other predefined names are written in a bold sans serif type font; for example, the **ID** key.
2) Operands of PDF operators or values of dictionary keys are written in an italic sans serif font, for example the */Catalog* key.
3) PPML and JDF element and XML element names in general are written in a bold sans serif type font, for example the **DOCUMENT** element.
4) Values of attributes of XML elements are written in an italic sans serif font. For example: *LineArt.*
5) Attribute names of PPML and XML elements are written in a bold italic sans serif font. For example, the ***ProcessColorModel*** attribute of the **ColorColorantControl** element.
6) In some cases the Xpath notation may also be used when showing the relationship of an XML element attribute or sub-element to an element. For example: ***ColorantControl/@ProcessColorModel***.
7) Placeholders for normally variable information are written in an italic serif font. For example: "The first value specifies the number of *columns* of page cells and the second value specifies the number of *rows* of page cells in the multi–up grid".
8) Within examples, use of bold font has no technical significance and is used for emphasis only.

1.4 Constraints

1) Similar to the CIP4 Interoperability Conformance Specification (ICS) documents, all NexPress front end supported elements, sub-elements, and attributes are presented in this document in a tabular format with the following column headings: "Name or Value", "Manager", "Worker", and "Description". Please refer to [BICS] for how to read ICS documents if further understanding of notations and format is needed.
2) NexPress front end also conforms to the size limits for attribute values as specified in [BICS].
3) Refer to [JDF] for additional detail on specific JDF references mentioned in this document. Information from the JDF Specification will not be duplicated in this document whenever possible. This will help in maintenance of this document and help to eliminate any discrepancies with [JDF].
4) NexPress front end, System 9.0 uses JDF 1.3.

1.5 References

[1] Adobe® Portable Document Format Reference Manual, Version 1.6, 2004, Adobe® Systems Incorporated, [PDF]

[2] Adobe Portable Job Ticket Format Technical Note #5620, Version 1.1, April 1999, Adobe Systems Incorporated, http://partners.adobe.com/asn/developer/PDFS/TN/5620.pdf, [PJTF]

[3] JDF Specification, Release 1.3, September 2005, www.cip4.org, [JDF]

[4] XML Specification, Version 1.0, February 1998, www.w3.org, [XML]

[5] PPML Functional Specification, Version 2.0, 27-March-2002, Print On Demand Initiative (PODi), [PPML]

[6] ANSI CGATS.20:2002 Graphic Technology, Variable printing data exchange using PPML and PDF (PPML/VDX), Approval dated 8-July-2002, American National Standards Institute, Inc. available from NPES, [PPML/VDX]

[7] Application Notes for CGATS.20-2002, Prepared by CGATS SC6/TF2, August 2004, http://www.npes.org/standards/tools.html, [PPML/VDX-AN]

[8] Base Interoperability Conformance Specification (ICS), Version 1.0, December 2004, www.cip4.com, [BICS]

[9] Kodak NexPress Developer's Interface Guide, for System 9.0, Version 2.0, January 2007, Eastman Kodak Company, [KNDIG]

[10] Kodak NexPress Developer's Interface Reference, for System 9.0, Version 2.0, January 2007, Eastman Kodak Company, [KNDIR]

[11] Portal Digital Master, Version 1.01, July 2001, NexPress Solutions LLC, [PDM]

[12] ISO 16612-2 Graphic Technology – Variable Data Exchange Part 2: Using PDF/X-4 and PDF/X5 (PDF/VT-1 and PDF/VT-2) , First Edition, 15-August-2010. [PDF/VT]

2 Differences between Software Releases

2.1 Changes in System 18.0

- Support for multiple-size substrates via Media/@NXP:MediaSizeName

2.2 Changes in System 17.0

- Support for InterpretingParams/@JobPrintSpeed and InterpretingParams/@JobPTRType

2.3 Changes in System 16.0

- WhiteFlood, WhiteImage and AutoWhite overcoats for White dry ink
- Support for InterpretingParams/@MirrorAround

2.4 Changes in System 15.2P01

- Enhanced Ink Optimization

2.5 Changes in System 15.2

- Halo Effect Correction
- DuraCoat Overcoat

2.6 Changes in System 15.1

- Windows 2012 Server Primary

2.7 Changes in System 15.0

- Economy Mode
- RGB Image Smoothing
- Ink Optimization
- Windows 2008 Server R2 Primary
- Windows 7 Pro Secondary

2.8 Changes in System 14.0

- New dry ink support: white, gold, pearlescent, neon pink

2.9 Changes between System 12.0 and System 12.1

- There have been no changes to the JDF Portal interface in System 12.1

2.10 Changes between System 11.2 and System 12.0

- Addition of support for PDF/VT-1, single file Variable Data Exchange.

2.11 Changes between System 11.0/11.1 and System 11.2

2.11.1 JDF

- Red Fluorescing clear dry ink support.

2.12 Changes between System 10.1 and 11.0/11.1

2.12.1 JDF

- Color Management for RGB color spaces may not be disabled.
- Dimensional clear dry ink support.

2.13 Changes between System 10.1 and System 10.2

2.13.1 JDF

- JDF Job Ticket support for M700e finishing options
- M700e related Preflight messages, warnings, and errors returned via JMF Notifications
- M700e finisher substrate mismatch events reported via JMF Error Notifications

2.14 Changes between System 10.0 and System 10.1

2.14.1 JDF

- A multicast locator was added for use by the Creo Prinergy Workflow Controller.
- Restrictions have been removed that required a specific set of types in the JDF Process combined node. Now only DigitalPrinting is required.
- Most of the configuration files in \CDFE_CONFIG\InitialConfig\Printready for use by Heidelberg PrintReady version 2.1 have been removed.
- Dry ink values were defined for micr, raised clear, and xd clear.
- Values added to support iqFlood, iqImage, xdPhoto, and xdGraphic.
- DeviceCondition and StatusDetails are reported for press condition "NeedsAttention".
- Resource Check warnings and errors are reported in JobPhase/@Comment for queued jobs.
- Reports descriptions and availability of media and dry ink resources.
- Custom attribute NXP:StopPoints created to stop job during processing.
- **JDFStorage directory (See Section 6.2.7) –**
  Use of the JDFStorage directory for completed JDF tickets has been deprecated.
- **Logfile location (See Section 3.4.4.3) –**
  Location of JDF Portal communications log is now specified by environment variable %CDFE_LOGFILES%.
- References to "NexStation" have been replaced with "NexPress".
- **Removed HDM:Shutdown (See Section** Error! Reference source not found.**) –**
  Removed JMF commands HDM:Shutdown and HDM:CheckFolderAccess. "Shutdown" is equivalent to HDM:Shutdown, and continues to be supported.

2.15 Changes between System 9.0 and System 10.0

- JDFHotFolder is available as a network share whenever the JDF Portal Queue is open.

- **No content filesize restriction in 3-part MIME attachments (See Section 6.2.1) –**
  PDF content files accompanying JMF SubmitQueueEntry in 3-part MIME are now spooled directly to a file to eliminate the filesize restrictions of earlier releases. To enable this behavior, the content-file part must be identified within the MIME header using "Content-Disposition: Attachment" and "Content-Type: application/pdf".
- "DeviceDefault" is now interpretated as "Gradation" for the following attributes within RenderingParams/NXP:CLCDDIQParams: HTTextBW, HTGraphicsBW, HTImageBW, HTTextCMYK, HTLineCMYK, HTGraphicsCMYK, HTImageCMYK.
- Support for "ftp:" added to QueueSubmissionParams/@URL.

2.16 Changes between System 8.5 and System 9.0

- Postscript support has been added for content files in a JDF runlist. In previous releases only PDF was supported.
- Content-Type in the HTTP header of a JMF response will use application/vnd.cip4-jmf+xml if this type is used in the request.
- When the JDF queue is closed or blocked, any JDF Tickets dropped onto the JDF hot folder are deleted. In the previous release, these files remained in the hot folder directory until the queue was opened.
- Support was added for URI scheme "FTP:" in **RunList/Layout/FileSpec/@URL**. This allows content files on an FTP server to be referenced within a JDF JobTicket.
- **Runlist/@Directory** is supported. It can be used to identify a base path for FILE: URLs containing a relative path. Previously relative paths had to be located within the JDF Hotfolder.
- **Runlist/@Directory** can also be used to designate a search path for VDX PDL content. In this case the URL must be either a local path (drive letter required if not on the C: drive) or a UNC path. For example:

```
<RunList Directory="/parentDirectory/theDirectory"  …/>
<RunList Directory="d:/parentDirectory/theDirectory" …/>
<RunList Directory="//servername/sharename" …/>
```

- Whenever applicable, failed JMF commands return one of the standard exception codes defined in the JDF 1.3 specification.
- Restrictions on media definition have been loosened so that **Location/@LocationName** is not always required. If omitted from the JDF, LocationName for body and cover medias will be determined from the media partitioning.
- Additional values in **Media/Location/@LocationName** are supported.
- **QueueFilter** is now supported on incoming JMF commands and queries.
- Persistent JMF messages that are sent to registered subscribers will include **Notification/@JobID** and **Notification/@NXP:QueueEntryID** when these values are known.
- JMF KnownMessages response includes SubmissionMethods query.
- When **Device/@DeviceID** is supplied in a JDF ticket, it is matched against the ***SenderID*** reported by the device in JMF responses.
- **JDF/NodeInfo** was moved to a resource in the JDF 1.3 specification. It is used to designate ***@TargetRoute*** for the completed JDF. The deprecated notation continues to be supported as well.
- Support was added for **JDF/@Activation**. A value of "Held" in a job ticket will set the initial state of the submitted job to "Held".
- **DigitalPrintingParams/@NXP:ColorFlow** has been added to identify jobs that should adhere to the ColorFlow policy of the press.
- **RenderingParams** with selected proprietary attributes has been added.
- **ColorSpaceConversionOp/@ConvRGBGrayToBlack** and **ColorSpaceConversionOp/@RGBGrayToBlackThreshold** have been added to control use of black colorant in RGB Graphics images.

- Proprietary attribute **NodeInfo/@NXP:DeleteWhenDone** has been added to the JDF ticket so that hotfolder jobs can designate whether to be automatically removed upon completion or abort.
- The JDF Portal Configuration file is not new, but the configuration file was not previously referenced in this document. For System 9.0, usage of the configuration value bDeleteWhenDone has been enhanced to control auto-delete behavior when submitted JDF hotfolder jobs reach completion or are aborted.
- **Device/ModelName** has been defined to facilitate rejection of a JDF ticket for situations in which the JDF was written using device-specific attributes and values that may be interpreted incorrectly on other JDF device. No action is taken in System 9.0 from this attribute.

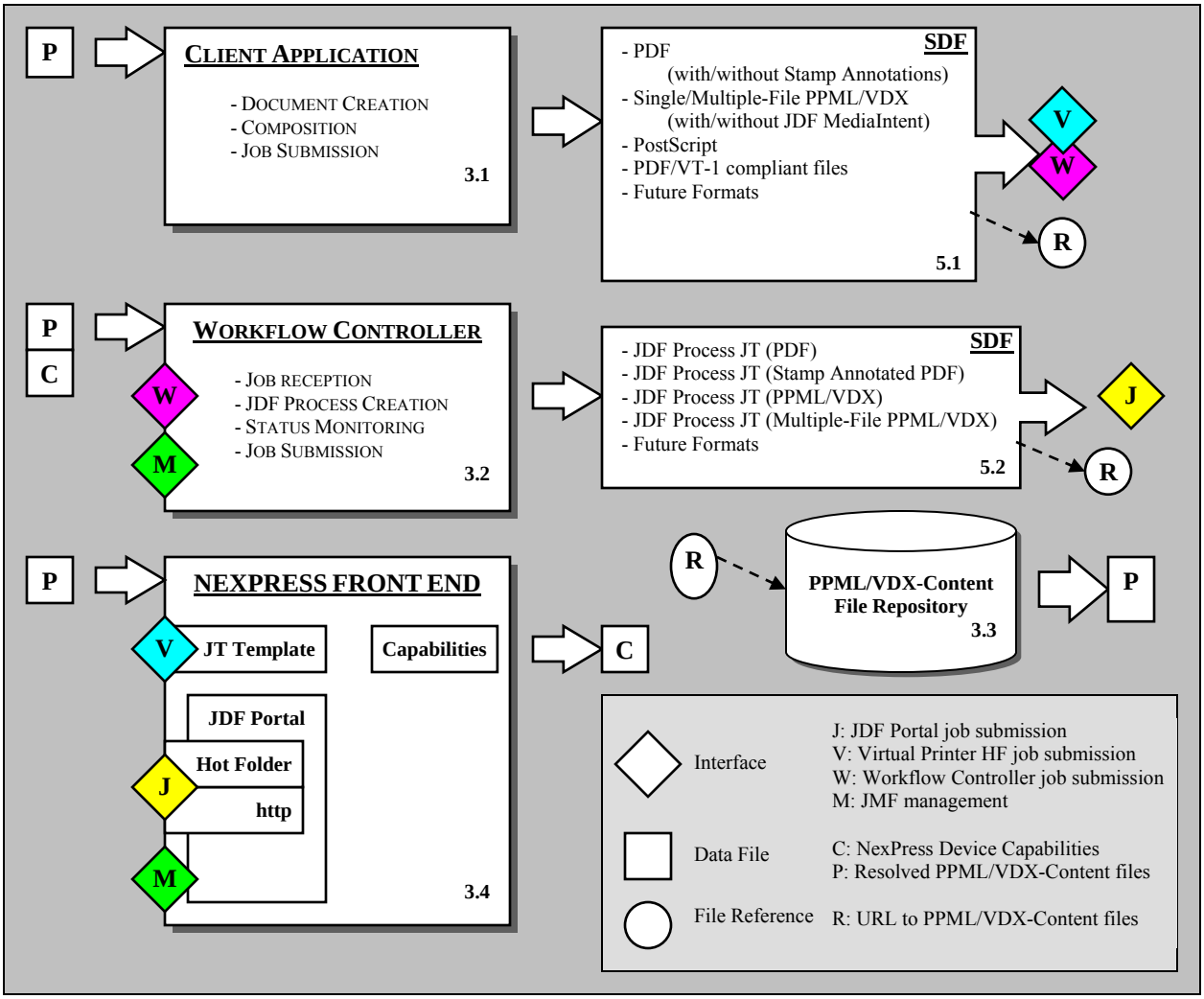
2.17 Changes between System 8.4 and System 8.5

The primary development goal for the NexPress JDF Portal in System 8.4 was to support Base ICS Level 2. The Base ICS (Interoperability Conformance Specification)[BICS] is published by CIP4. In System 8.5, development focused on aspects of ICS Level 3 support that were thought to have the greatest benefit for users of the NexPress.

- The NexPress Combined Process node does not have to be the root node. Any JDF product intent nodes in a job ticket received through the JDF Portal Interface are ignored, but the ticket will be processed if a NexPress Combined Process Node is found.
- The JMF **SubmitQueueEntry** command can use "CID:", "HTTP:", or "FILE:" access to reference a file containing the JDF Process job ticket.
- Support was added for MIME packaging of content files. MIME packaged submissions using a MIME file to the JDF Portal Hot Folder, or a 3-part MIME with JMF SubmitQueueEntry are both supported.
- The location of the JDF Portal Hot Folder has changed to \\<servername>\HotFolder\JDFHotfolder.
- Support for JMF Query SubmissionMethods was added; it returns the hot folder location.
- Support for **FileSpec/Disposition** was added to provide automatic deletion of content files. By default, the content file is preserved.
- Support was added for **QueueSubmissionParams/Disposition**. Any job without this Disposition element will be automatically deleted when the job status becomes *Completed* or *Aborted.*
- The completed JDF file can be sent to a web server by specifying "HTTP://servername:PORT#/" in the URI for ***TargetRoute*** or ***ReturnURL***.
- Support is extended for PDF version 1.6 as produced by Adobe® Acrobat® 7.
- To assist developers who are integrating a controller with the JDF Portal, all JMF communications received and sent by the NexPress front end are logged. Only JMF messages and MIME-encoded JDF job tickets are written to the log; received PDF content is not saved.

3 NexPress Workflow

The diagram shows the NexPress workflow. There are Client Applications generating content files and optional Workflow Controllers receiving some of this content, transforming it into JDF Process-based content, and providing other value-added functions. The NexPress front end provides a rich set of interfaces to submit and manage jobs and to print the content files on the NexPress digital production color press.

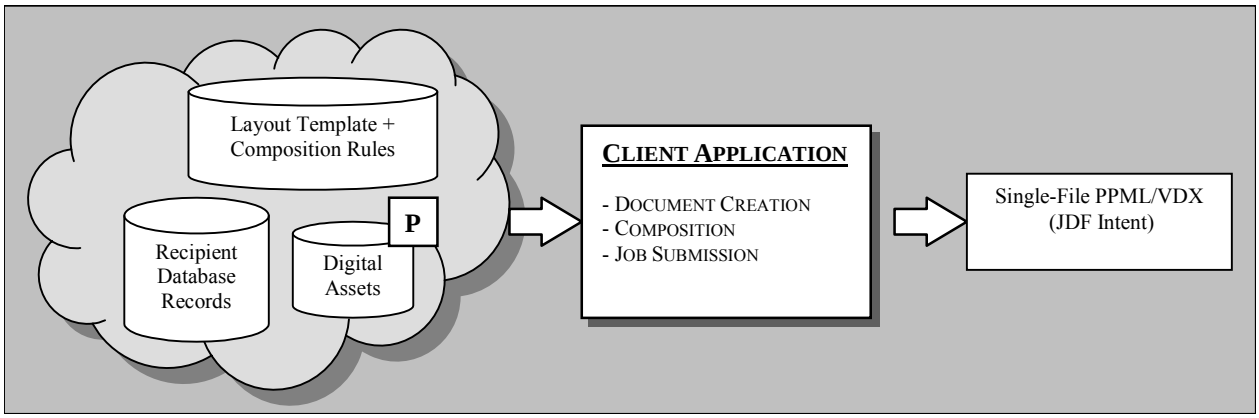


3.1 Client Application

The box in the diagram labeled **Client Application** represents the functionality of prepress applications used to create static and variable page content data. **Client Applications** create Structured Document Formats for consumption by Workflow Controllers (through W interface), the NexPress front end (through V or J interfaces), and by other **Client Applications** including NexPress workflow tools. The Structured Document Formats created by such applications should generally be device and production-workflow independent. They may include print product description data (also referred to as Product Intent data) that describe the basic intent of the final print product but not the production processes required to produce the final print product.

To create a document that uses a variable Structured Document Format (e.g. PPML/VDX), the **Client Application** would contain a Composition Engine to combine a layout template, recipient database records, digital assets (e.g. PPML/VDX-Content Files) and composition rules. It creates the Structured Document Format that best represents the print product, as shown in the following diagram. Note that there are differences in the PPML/VDX supported

by interfaces V and J. Interface J does not support JDF intent (including JDF MediaIntent that is supported by interface V).



To create a document that uses a static Structured Document Format (e.g. PDF), the **Client Application** would utilize a software application such as Adobe® Acrobat Professional, or another creative application that uses a print driver capable of generating the static Structured Document Format.

3.1.1 Kodak Imposition Viewer

The Kodak Imposition Viewer is a NexPress workflow tool. It is an Adobe® Acrobat Plug-in provided with the NexPress digital production color press. This tool allows a prepress operator to open and view both PDF and PPML/VDX-Layout files, define a job ticket, embed a JDF Process ticket into the PDF, and submit the file to the Virtual Printer Hot Folder (Interface V) of the NexPress front end. It does not submit JDF Intent, and will convert imposition parameters specified in JDF Intent into JDF Process prior to submission.

The Kodak Imposition Viewer extends the Virtual Printer Hot Folder interface by allowing an operator to examine PDF or PPML/VDX instances, and prepare the file for printing on the NexPress digital production color press.

For PPML/VDX content, the JDF Product Intent data embedded within the PPML/VDX-Layout file can be displayed by the tool, and can be used as guidance to the operator in specifying the job ticket. The Kodak Imposition Viewer will also convert the JDF Intent elements of BindingIntent, LayoutIntent, and MediaIntent into JDFProcess prior to submission. The tool's job ticket editor functionality allows selection of perfecting, imposition, and media requirements. It produces a JDF Process job ticket that is embedded within the content file (PDF or PPML/VDX-Layout file). Even though a job ticket can be exported from the Kodak Imposition Viewer as a JDF Process file, this JDF Process data cannot be subsequently used for submission to the JDF Portal Interface. In particular, ***RunList/LayoutElement/FileSpec/@URL*** references the content file as an embedded JDF.

It is important to note that because the Kodak Imposition Viewer only supports submission into the Virtual Printer Hot Folder (Interface V), jobs submitted to the NexPress front end from the Imposition Viewer cannot subsequently be managed using the JDF Portal JMF management interface (M).

For more details on the use and features of this NexPress workflow tool, refer to the online Help guide provided with the product.

3.1.2 Kodak SmartBoard Document Mastering Software

Kodak SmartBoard Document Mastering software is another NexPress workflow tool. It works in conjunction with Adobe® Acrobat to provide a powerful make-ready solution for digital print-on-demand. It is available for the Windows PC platform for use with the NexPress digital production color press. SmartBoard allows a prepress operator to open and view both PDF and PPML/VDX-Layout files, define a JDF Process job ticket, and submit the file to the Virtual Printer Hot Folder (Interface V) or JDF Portal (Interface J).

Kodak SmartBoard document mastering software:

- Provides a powerful "make-ready" production workflow tool that improves user productivity by reducing job setup time, and by reducing errors in job reprints because the job ticket is attached to the content.

- Provides powerful tools to assemble jobs, ticket jobs, and direct print or batch print jobs resulting in efficient operations and lower costs through a state-of-the-art PDF workflow within the production workflow.
- Allows addition of third party software to extend capabilities specific to customer needs while retaining purchase and training investment because of its open architecture and adherence to industry standards such as Acrobat, PDF, and Twain.
- Offers multiple configurations to maximize flexibility in the pre-print environment.
- Provides a full range of content repurposing features such as tab creation, color editing and imposition.

For more details on the use and features of this NexPress workflow tool, refer to the online Help guide provided with the product.

3.2 Workflow Controller

The box in the diagram labeled **Workflow Controller** represents the class of software applications characterized by [JDF] as a Controller/Agent. A Controller/Agent creates and submits JDF Process Job Tickets to JDF devices such as the NexPress front end (using interface J). It may also monitor the status of the execution of jobs it submits (using interface M).

Included with the data submitted to interface (W) may be a JDF Product Intent Job Ticket that describes the finished print product in terms of its binding style, media styles, and finished page size and layout. A **Workflow Controller** that supports submission and management of Multiple-file PPML/VDX jobs may need to resolve URL references to the PPML/VDX-Content files. This access is represented in the diagram by input data file (P) fetched from the PPML/VDX Content File Repository.

A **Workflow Controller** that targets a NexPress digital production color press must create a JDF Process Job Ticket using information supplied by JDF Product Intent and/or from other product description information. The **Workflow Controller** must add the NexPress Combined Process Node it has created to the JDF job ticket, and submit the JDF job ticket to the NexPress front end's JDF Portal Interface (J) through one of the identified mechanisms.

Although the NexPress Combined Process Node can be added to a JDF job ticket that already contains JDF Product Intent data, a complete JDF job ticket including both Product and Intent does not have to be submitted to the JDF Portal Interface. Spawning, as defined in [JDF], may be an implemented feature of the **Workflow Controller**. In this case the controller would only submit a spawned JDF job ticket containing a NexPress Combined Process Node to the JDF Portal Interface (J) for execution.

3.2.1 Hybrid Workflow Controller

Some prepress applications and tools may have the combined functionality of both **Client Application** and **Workflow Controller**. Such tools prepare page content data, and create a JDF Process job ticket containing a NexPress Combined Process Node. This ticket is then directly submitted to the NexPress front end's JDF Portal Interface (J). Software applications that have this combined functionality are referred to in this document as **Hybrid Workflow Controllers**.

These workflow tools are commonly used in variable data printing environments. In such environments, the authoring tools are given specific knowledge of the target production workflow, and are tightly coupled to (or directly target) a particular digital printing device. A hybrid workflow controller functions as both the design application and composition engine, and creates both the VDP page content data files and a JDF Process job ticket. It must have complete awareness of the print production workflow and printing device. Product Intent information is often not required, so converting from JDF Product Intent to JDF Process is usually not necessary.

3.3 PPML/VDX Content File Repository

The element in the diagram labeled **PPML/VDX Content File Repository** represents the workflow functionality of a PDF content repository or file system that stores PPML/VDX-Content files external to the NexPress front end. A repository is useful in Multiple-file PPML/VDX workflows that are designed to maximize efficiencies related to the management of recurring, and/or segmented VDP jobs with shared PPML/VDX-Content files.

The **PPML/VDX Content File Repository** may be a file system or a digital asset management system that is directly accessible to the NexPress front end, **Client Application**, and/or **Workflow Controller**. It should be

accessible as a network share or have http server functionality. The repository contains PPML/VDX-Content files used by Multiple-file PPML/VDX Instances. For adequate performance with Multiple-file PPML/VDX jobs utilizing a significant number of large image files, a high-performance file system and network connection are required. File access and data transfer efficiency have a direct impact on job processing performance.

The data files (P), shown as input to the various workflow processors, represent support for resolving and actively fetching PPML/VDX-Content files referenced from the PPML/VDX-Layout file.

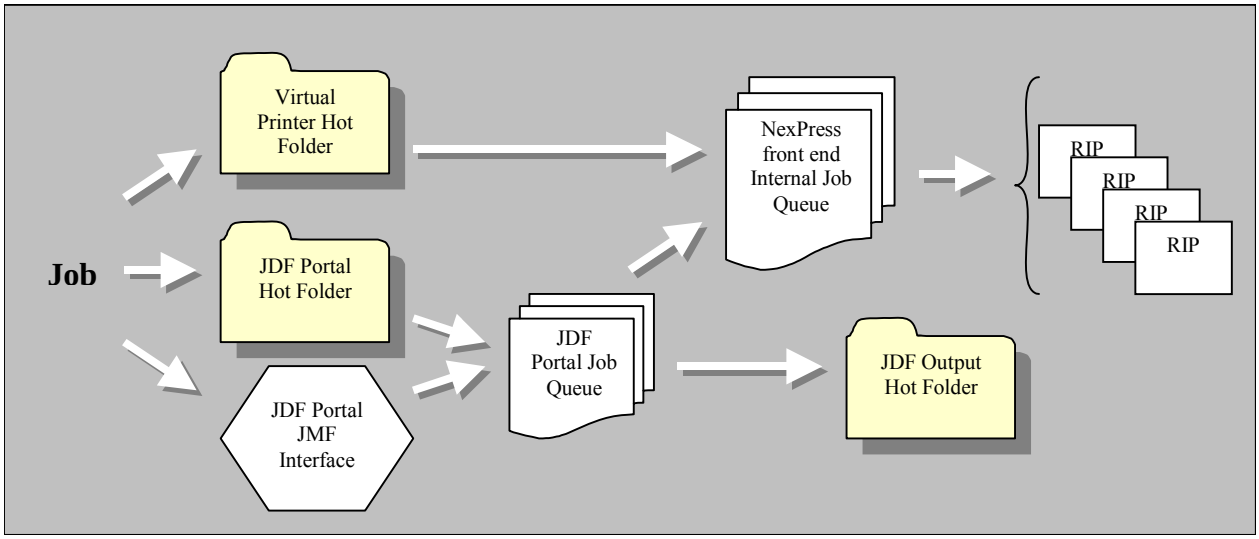
The NexPress front end provides support for Multiple-file PPML/VDX jobs, and can resolve URL references (R) to PPML/VDX-Content files for jobs submitted using either of its job submission interfaces, (V) or (J). When using URL access of type "file," the NexPress front end, the Workflow Controller, and the Client Application all require sufficient network permissions to access the Content File Repository.

3.4 NexPress front end

The NexPress digital production color press is comprised of a NexPress front end and a NexPress digital production color print engine. The NexPress front end functions as both a prepress workflow controller and scalable raster image processor (RIP) to the NexPress digital production color print engine. Print job submission and print job processing workflow can be controlled through the JDF Portal Interfaces (J and M). Print jobs can also be submitted using the Virtual Printer Hot Folder Interface (V), and monitored or controlled using the NexPress Client. Only jobs submitted through the JDF Portal interface can be subsequently monitored or controlled using the JDF Portal Interface (M). Print job resources can only be managed using the NexPress Client.

3.4.1 Job Flow

Multiple job submission interfaces to the NexPress front end are simultaneously active. The flow of jobs through the NexPress front end is shown in the following diagram.



Regardless of the submission interface, all jobs are placed into the Internal Job Queue for processing. Jobs in the Internal Job Queue are referred to as Internal Queue Entries. Internal Queue Entries are RIPped and printed in the order of their arrival. Because job processing capabilities overlap within the NexPress front end, multiple jobs will execute simultaneously depending upon the availability of processing resources.

Jobs submitted to a Virtual Printer Hot Folder are directly placed into the Internal Job Queue; jobs submitted to the JDF Portal interface first enter the JDF Portal Job Queue, and are subsequently placed into the Internal Job Queue by the JDF Portal. JDF jobs in the JDF Portal Job Queue are referred to as JDF Portal Queue Entries. Each job in the Internal Job Queue is identified by a unique CDFE Job Identifier. Portal Queue Entries have an additional QueueEntryID that is required for job identification in JMF messaging on Interface (J).

A completed JDF ticket is created for each job submitted to the JDF Portal Interface. This ticket can be written to an explicit location by specifying ***JDF/NodeInfo/@TargetRoute*** in the job ticket or ***QueueSubmissionParams/@ReturnURL*** in the JMF SubmitQueueEntry. If a TargetRoute or ReturnURL is not specified, the completed ticket is instead written to c:\HotFolder\JDFHotfolder\Unmerged. This completed JDF is written only if the job has been submitted by the portal into the NexPress front end's Internal Queue; a JDF job removed prior to processing will not produce an output JDF ticket. JDF job tickets accumulate in \\HotFolder\JDFHotfolder\Unmerged until deleted by the user.

Only the NexPress Combined Process Node is updated within the output JDF. The status of a finished job is reported in ***JDF/@Status***; the value is "Completed" or "Aborted". Entries are added to **JDF/AuditPool** for significant processing events. ***JDF/ResourcePool/Component/@Status*** becomes *Available* when the job is completed successfully.

3.4.2 NexPress Client

The NexPress front end has its own operator interface and management application, referred to as the NexPress Client. The NexPress Client allows qualified front end operators to setup and control the baseline processing capabilities and operational state of the NexPress digital production color press. A NexPress Client is available locally on the NexPress front end; it can also be installed and run on additional networked computers if remote access is desired. The NexPress Client does not provide a programmatic interface that can be utilized by a Workflow Controller.

The NexPress Client provides functionality to

- Create a Virtual Printer Hot Folder and configure the folder's Job Ticket Template
- Manage all resource data including installation and management of fonts, media profiles, and ICC profiles stored on the NexPress front end
- Monitor and control jobs that have been submitted to the Internal Job Queue

3.4.3 Job Submission Interface

The NexPress front end has two available job submission interfaces through which print jobs may be submitted. These interfaces are the Virtual Printer Hot Folder (V) and the JDF Portal (J). Refer to section 6 for details on supported content that can be submitted through each interface.

3.4.3.1 Virtual Printer Hot Folder Job Submission Interface (V)

The NexPress front end's Virtual Printer Hot Folder Job Submission Interface (V) is file-based and proprietary. It accepts a single data file in one of the Structured Document Formats listed previously in the NexPress workflow diagram. Submission is performed using a desktop GUI drag-and-drop, or a software application that writes the data file directly to the Virtual Printer Hot Folder.

Multiple Virtual Printer Hot Folders can be created and active. Virtual Printer Hot Folders are managed using the NexPress Client; each folder is identified by a unique, user-defined name. When creating a new Virtual Printer Hot Folder, it is recommended that the identifying name be descriptive of the Virtual Printer Job Ticket Template configuration. It is also recommended that the hot folder name conform to any namespace scheme defined by the hosting enterprise. A list of available Virtual Printer Hot Folders is reported in the Device Capabilities File. Refer to section 5.2 for more information.

Each Virtual Printer Hot Folder has its own Virtual Printer Job Ticket Template to define the default processing settings for page content files (e.g. PDF files) submitted to the hot folder. An operator uses the NexPress Client to configure this Virtual Printer Job Ticket Template.

3.4.3.2 JDF Portal Job Submission Interface (J)

The NexPress front end's JDF Portal Job Submission Interface, or JDF Portal Interface (J), supports two submission methods: a JDF Portal Hot Folder and an HTTP JMF Interface. The JDF Portal Interface is independent of the Virtual Printer Hot Folder Job Submission Interface (V).

Unlike the Virtual Printer Hot Folder that accepts submission of PDL files directly, the JDF Portal Interface only accepts JDF job ticket files. Each job ticket file must contain exactly one NexPress Combined Process Node as identified by the value of attribute ***JDF/@Types***. The submitted JDF job ticket data should specify all parameters for processing the print job, and identify by URL reference the structured document data file that provides page

content for the job. Default values for Job Ticket attributes are fixed and cannot be viewed nor configured by an operator.

The syntax and semantics of the JDF job ticket, including the definition of a NexPress Combined Process Node, are detailed in [KNDIR]. Any JDF product intent nodes in a job ticket received through the JDF Portal Interface (J) are ignored; the ticket will only be processed if a NexPress Combined Process Node is found.

3.4.3.3 JDF Portal Hot Folder

The JDF Portal Hot Folder supports drag and drop submission of JDF job tickets as defined by [KNDIR] and [JDF]. In contrast to Virtual Printer Hot Folders, there is only a single JDF Portal Hot Folder, and there is no configurable Virtual Printer Job Ticket Template assigned to that hot folder. A submitted JDF job ticket is merged with the system default job ticket during job processing.

The NexPress expects workflows that use hot folder submission to be "Drop and Forget", and will automatically remove a JDF ticket from the portal queue upon completion or when aborted through the JMF interface. This behavior is configurable; see "JDF Portal Configuration File" in [KNDIR] for more information.

3.4.3.4 JDF Portal HTTP JMF Interface

The JDF Portal JMF Interface supports HTTP based submission of JDF job tickets as defined by [KNDIR] and [JDF]. The NexPress JDF Portal uses port 49700. The JDF Portal HTTP server provides only limited HTTP support. In particular, the following restrictions must be observed:

- Only the HTTP POST command is supported.
- HTTPS is NOT supported.
- Content-Length is required; Chunked Transfer-Encoding is NOT supported.
- All HTTP messages are sent to Port 49700.
- Content-Type may be text/xml or application/vnd.cip4-jmf+xml. The same type will be used by the JDF Portal in its response.

Example: HTTP POST

```
POST * HTTP/1.1
Content-Type: text/xml
User-Agent: NxpJdfTest
Host: KP-GDF00:49700
Content-Length: 329
Cache-Control: no-cache

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2006-06-20T09:28:57-
    04:00" Version="1.1">
  <Query ID="m060620092857_0036" Type="KnownMessages" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xsi:type="QueryKnownMessages"/>
</JMF>
```

To submit a job using JMF SubmitQueueEntry, the JDF job ticket can be attached as a MIME package, or referenced as a URI using a scheme of HTTP or FILE.

The JDF Portal also supports MIME packaging of content files to either the JDF Portal Hot Folder or to the HTTP interface. MIME encoding is not recommended for content files larger than 150MB.

Refer to section 6.2 for details about submission to the JDF Portal Interface.

3.4.4 Job and Press management Interface

The NexPress front end packages the press configuration into an XML file using a proprietary schema. This Device Capabilities File (C) can be obtained directly from the press at \CDFE_DATA\CDFExchg\DevCaps.xml. It is also available via http as cdfeUisAccess/devcaps.xml. Alternatively, this same device capabilities can be requested using a proprietary JMF query of type "NXP:DeviceCapabilities".

Remote job control and monitoring is performed using the JMF Management Interface (M). JDF jobs that have been submitted by the JDF portal to the Internal Job Queue can also be monitored and controlled using the NexPress Client.

3.4.4.1 Device Capabilities File (C)

The NexPress digital production color press provides a mechanism for describing its basic device capabilities to an external client application. The Device Capabilities File expresses press capabilities using the NexPress Capabilities Format (NCF). Refer to [KNDIR] for a complete definition of NCF including a listing of the NCF data type dictionary (DTD) used to define its XML syntax. NCF is NexPress proprietary; it is not related to the device capabilities defined by [JDF] even though some of the semantics overlap.

The NexPress front end maintains the Device Capabilities File. The file is updated dynamically as resources are added or deleted; it contains the following information:

- The network name of NexPress digital production color press.
- The names of all Virtual Printer Hot Folders defined for the press. A human-readable comment describing each Virtual Printer Hot Folder is also provided. The parameter settings of a Virtual Printer Job Ticket Template are not communicated in the Device Capabilities File.
- The media catalog for the press. Media entries include the substrate name, type, size, and other characteristics. The Device Capabilities file does not provide the currently loaded media.
- Other system resources including the fonts, imposition templates, ICC profiles, spot color tables, and screening options.
- The physical characteristics of substrate supply locations.
- The available finishing options and physical characteristics of delivery locations.

The operating state and status of a NexPress digital production color press is not described in the Device Capabilities File.

3.4.4.2 JMF Management Interface (M)

The JDF Portal JMF Interface (M) responds to HTTP POST commands directed at port 49700. Using this interface, **Client Applications** can manage the JDF Portal queue and monitor all jobs submitted to the JDF Portal. Because only jobs submitted through the JDF Portal Interface (J) can be monitored or controlled using the JMF Interface (M), the JMF Interface is primarily used by **Workflow Controllers** that are utilizing the NexPress front end as a remotely controlled and monitored printing device.

The NextPress front end supports most of the functionality required for Level 2 compliance of the JDF Base ICS. In addition, it also supports many of the JMF messages required for Level 3 compliance. Refer to specifications [JDF] and [KNDIR] for detail on specific JDF syntax and semantics. Refer to Sections 5 and 6 for general information on Job Submission and Management using the JMF Interface.

3.4.4.3 JDF Portal Communications Log

To assist developers who are integrating a controller with the JDF Portal, all JMF communications received and sent by the NexPress front end are logged. Only JMF messages and MIME-encoded JDF job tickets are written to the log; received PDF content is not saved. A timestamp and a tag with the text "INCOMING" or "OUTGOING" precede each entry in the log file.

The logging mechanism rotates between 20 log files; each file targeted at 256KB. After addition of an entry that writes beyond this targeted size, the log file is closed. A new log file is opened with the next entry. These logs are created in %CDFE_LOGFILES%\JDFPortal\. Logs are named ServerLog1.txt, ServerLog2.txt, … ServerLog20.txt.

By default, logging is enabled. Logging can be disabled by modifying the file %CDFE_LOGFILES%\JDFPortal\LogConfig.txt. It contains a single line; the first field designates the logging option. Valid values are ALL, IN, OUT, and NONE. The JDF Portal must be restarted for the configuration changes to take effect.

3.4.5 NexPress Processing

The NexPress front end is a high-powered server capable of multiple, simultaneous, resource intensive tasks. It is scalable in terms of processing capability and performance. The base configuration consists of a single server with

dual CPUs and a multiple-RIP processor. Additional server nodes can be added to the base configuration to provide more RIP processors. In this multi-node configuration, one of the servers is designated as the primary node. All jobs and JMF communication are directed to the primary node. In many cases prepress processing of multiple jobs will occur in parallel.

3.4.5.1 Imposition Support

The NexPress front end provides a parameterized method for specifying an imposition layout of a print run; this method is known as In-RIP Imposition. During In-RIP Imposition, the NexPress front end will determine the final layout for each surface sheet based on the following job characteristics:

- aspects of the selected media,
- trim size,
- the number of rows and columns of pages per sheet surface,
- perfecting requirements,
- page distribution scheme (i.e. saddle, perfect, sequential).

The NexPress front end automatically generates sheet marks as controlled by the job ticket. Such marks are specified relative to the parameterized imposition layout.

For jobs submitted through the JDF Portal interface, the imposition requirements are specified in the JDF job ticket's NexPress Combined Process Node definition.

3.4.5.2 RIP Caching

The NexPress front end supports complex variable data print (VDP) jobs. The pages of a conventional print job are generally RIPped only once regardless of the number of copies. In contrast, only a single copy of a VDP job is usually printed because the content on each page is unique. This additional processing makes VDP jobs very RIP-intensive.

Even though page content of VDP jobs is unique, much of the graphical content is recurring. The NexPress front end attempts to minimize the need for redundant RIPping of page-content data that recurs in an equivalent rendering context (across multiple pages). Previously RIPped recurring content of the job is cached and reused in its rendered state for the duration of the job context. The cache is flushed once the job completes. This cache, however, is limited in size; the number of pre-RIPped elements from the NexPress front end's cache memory will be reduced if capacity is reached. The NexPress front end will automatically re-fetch and RIP recurring source data as needed if renditions are reduced from its cache. Performance may be diminished if significant re-RIPping occurs due to insufficient cache memory.

The NexPress front end does not provide tools to manage its VDP raster cache memory. Raster-caching is accomplished automatically, and is transparent to the operator and to any submitter. No operator knowledge is required of the content elements to be cached.

3.4.5.3 Multiple Substrate Support

The NexPress digital production color press supports multiple substrates, or media, in a print job. Some workflows specify page to medium mapping by tagging pages within the content file. These tags are stored as symbolic name strings in the content file meta-data. In a PDF document, stamp annotation is used to tag the pages. The mapping of a symbolic name within the page content file to a substrate on the press is specified in the process job ticket. The software that creates the process job ticket data (Virtual Printer Job Ticket, or JDF job ticket) generally provides the user a way to specify the mapping of content-file tags to substrates.

4 Supported Page Content Formats

The page content formats currently supported by the NexPress front end include the following *Structured Document Formats:*

- Adobe® Portable Document Format [PDF].
- ANSI CGATS.20 (PPML/VDX) Standard [PPML/VDX].
- ISO 16612-2 Part 2: Portable Document Format/Variable Transactional [PDF/VT]

The *PDL* formats currently supported include:

- Adobe® PostScript 3.0 (and earlier versions)
- NexPress Portal Digital Master as defined by the PDM Specification [PDM]

This document only provides details on support for the Adobe® PDF and ANSI PPML/VDX file formats. These two content file format are the only ones supported by the NexPress JDF Portal interface.

4.1 Adobe® Portable Document Format (PDF)

The NexPress front end is capable of consuming PDF data that conforms to the Adobe® Systems Inc. version 1.7 and earlier Adobe® PDF specifications.

4.1.1 Specifying Media Using PDF Stamp Annotations

The pages of PDF files may be "rubber stamped" with special *Stamp Annotations*. These annotations provide meta-information to a Workflow Controller or to the NexPress front end that can be mapped to a specific media.

Multiple substrates are supported in a print job. The NexPress front end supports two methods to identify the substrate for a PDF page. As an alternative to *PDF Stamp Annotations*, *Cover Mode* can be selected in the job ticket to select substrates for the first and last pages. A job will fail resource check if a required substrate is not installed.

The PDF Stamp Annotation method is job-ticket based, but also requires specific meta-data to be present in the PDF. Semantically, the use of PDF Stamp Annotations is simple. Using stamp annotations, the pages of a PDF file can be mapped to one of the logical media: *Body*, *Cover*, *Insert, or Insert[n], where n=2-8*. Substrates are specified in the process job ticket as a mapping from logical media to a physical medium from the NexPress front end's media catalog.

The creator of the process job ticket (i.e. Workflow Controller) specifies this mapping. When using a Virtual Printer Hot Folder, the Virtual Printer Job Ticket Template defines the mapping. When using the JDF Portal job submission interface, this mapping is defined in the NexPress Combined Process node.

Pages within a PDF file that have no stamp annotation are implicitly mapped to *Body* media. Consequently a single-substrate PDF-based job does not need to be stamp-annotated; it will use *Body* media. Explicit *Cover* ,*Insert, and Insert[2-8]* stamp annotations are required only when multiple media are used.

Stamp Annotations are visible and printable if their graphical representation overlaps any part of the page content area displayed by the Adobe® Acrobat viewer application. They are also visible in the Acrobat viewer when located in the gray area off of the page content area. If the stamp annotation overlaps the page area it may print as part of the page content; this is likely to be undesirable. It is recommended that stamp annotations used to identify logical substrates be placed in the gray area beside the page image so the annotation will never be printed regardless of the setting of the *Annotations* check box within the Adobe® Acrobat print window.

4.2 ANSI CGATS.20 – PPML/VDX

The PPML/VDX format is a structured VDP content file format. It is supported by the Virtual Printer Hot Folder and JDF Portal interfaces. ANSI PPML/VDX is a structured document format that guarantees page independence and device independence. This means that like Adobe® PDF, it is possible to produce a PPML/VDX instance in a creative design workflow that is indifferent to the print production workflow used to manufacture it.

PPML/VDX specifies two conformance levels known as PPML/VDX-Strict and PPML/VDX-Relaxed.

- **PPML/VDX-Strict** requires all PDF files to conform to either the PDF/X-1a or PDF/X-3 standards, and requires the PPML data to be embedded within the PPML/VDX layout file. It also requires that all **Binding** references in the **ContentBindingTable** specify MD5 checksums and unique IDs for all referenced PPML/VDX-content files. The purpose of this conformance level is to allow the specifier of the PPML/VDX job to maximize control over the integrity of the PPML/VDX instance. This relieves the receiver of the exchanged data from most liability issues related to the correctness of the exchanged data. The receiver is guaranteed that all color data is identified and that the completeness of the file set can be verified.

- **PPML/VDX-Relaxed** relaxes many of the restrictions of PPML/VDX-Strict. For example, the PPML data may be specified in a separate XML data file, and the MD5 Checksum and unique ID in **Binding** references in the **ContentBindingTable** are optional. The use of PDF is not limited to PDF/X-1a and PDF/X-3; any PDF data can be used. The purpose of this conformance level is to enable the use of PPML/VDX data in trusted exchange scenarios, as well as within environments with more controlled and integrated authoring and print production workflows. The degree of integrity verification support depends upon the liability requirements of the VDP application, and how tightly integrated the workflow.

The NexPress front end is capable of consuming PPML/VDX data that conforms to both the *PPML/VDX-Strict* and *PPML/VDX-Relaxed* conformance levels as defined by the PPML/VDX standard (for more details on the PPML/VDX conformance levels refer to the [PPML/VDX] and [PPML/VDX-AN]). The NexPress front end consumes PDF; however, in the case of PDF/X conforming PDF data, it does not interpret output rendering intents.

In VDP applications, the structure of the data format used must enable consuming 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 page description languages (PDL). PPML/VDX, a structured document format based on a subset of the PODi Personalized Print Markup Language (PPML), incorporates the same features for enabling RIP efficiency as other proprietary (i.e. PostScript-based) VDP formats.

PPML/VDX data specifies reader order sequences of independent pages with no implied print production ordering. Print production information such as imposition layout, page distribution scheme, and printing device parameterization is specified in JDF job ticket data external to the PPML/VDX data. Alternatively, PPML/VDX can be used in workflows in which the composition system itself specifies production parameters via the job ticket.

The ability to late-stage target a PPML/VDX job 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 in order for it to be efficiently manipulated and reordered in preparation for print production. This is important because the order in which pages are utilized on a sequence of imposed sheets (printer spreads) is usually different from the reader order in which the PPML data specifies them. VDP languages that combine layout information with actual content data cannot guarantee efficient parsing and manipulation as required by an imposition process. This is why embedding graphical content data within the PPML data (using the PPML **INTERNAL_DATA** element) is prohibited by the PPML/VDX standard. For similar reasons, PDF is used to specify graphical content data.

4.2.1 Single-File PPML/VDX instances

Single-file PPML/VDX data is supported by the NexPress front end and supporting workflow tools. A single-file PPML/VDX instance consists of the PPML/VDX-Layout file with the PPML data embedded. This is the most compact and portable form of a PPML/VDX instance; no external PDF file references need to be resolved to obtain dependent PPML/VDX-content files or PPML data. The PDF pages used as page content referenced by **EXTERNAL_DATA_ARRAY** elements in the PPML data are completely contained within the PPML/VDX-Layout file itself.

In accordance with the PPML/VDX Standard [PPM/VDX], the **ContentBindingTable** element must only contain a ***Self*** sub-element whose ***Src*** attribute value must exactly match that of the ***Src*** attribute of all PPML **EXTERNAL_DATA_ARRAY** elements.

4.2.2 Multiple-file PPML/VDX instances

Multi-file PPML/VDX data is supported by the NexPress front end and supporting workflow tools. A multiple-file PPML/VDX instance consists of a single PPML/VDX-layout file and one or more additional PPML/VDX-content files.

As required by the PPML/VDX Standard, the **ContentBindingTable** of the PPML/VDX-Layout file must contain a **Binding** sub-element for each unique PPML/VDX content file that is referenced by **EXTERNAL_DATA_ARRAY** elements in the PPML data.

For PPML/VDX-Strict conformance, the PPML/VDX Standard requires that the ***MD5_Checksum*** and ***UniqueID*** attributes of each **Binding** element are specified. This permits the receiver of the data to verify completeness of all data comprising the PPML/VDX instance. These attribute values should be tested against those determined from the resolved PPML/VDX-content files prior to submission; however, the NexPress front end ignores the ***MD5_Checksum*** and ***UniqueID*** attributes of each **Binding** element.

The VDP composition engine generates PPML/VDX instance data and writes **Binding*/@Src*** elements that reference PPML/VDX-content files. **The URL reference of the *Src* attribute must be resolvable in the receiving environment.** Specifically, the controller (such as the NexPress front-end) receiving the PPML/VDX-Layout file must have access to any referenced PPML/VDX-content files.

The NexPress front end and accompanying workflow tools support the URL schemes *file:, ftp:,* and *http:* for the ***Src*** attribute of **Binding** elements. Use of *file*: can be problematic because the NexPress must have non-authenticated access to the specified files. This requirement is especially difficult to satisfy in a Windows 2003 Server environment if the NexPress is not a member of the domain. For this reason, usage of *ftp*: or *http*: is recommended.

4.3 PDF/VT

The NexPress front end is capable of consuming PDF data conforming to the ISO 16612-2 standard for PDF/VT-1 (single file) documents. The implementation is similar in some respects to VDX support (printing record ranges, display of records in Imposition Viewer/Content Edit) and similar to PDF in other respects (media selection using stamps).

The NexStation uses records as the only means of segmentation in its PDF/VT support. If the record layer indicator is missing the NexStation will treat the PDF/VT document as a single record.

Content editing of a PDF/VT file is disabled as the NexStation Content Edit plug-in is not able to update the *DPartRoot* dictionary.

Existing NexStation media stamp annotations are available for any page in the PDF/VT file and are processed accordingly (see section 4.1.1 above). No other media selection mechanism is available.

4.4 Adobe® Postscript

The NexPress front end only supports Adobe® Postscript through the Virtual Printer Hot Folder interface. Postscript is converted to PDF prior to printing, and is not supported in JDF Job Tickets.

4.5 NexPress Portable Digital Master (PDM)

The NexPress legacy format known as the *NexPress Portable Digital Master* (*PDM*) format is an earlier version of the draft standard that eventually became the ANSI-accredited CGATS.20-2002 (PPML/VDX) standard in July of 2002. Although PDM is suitable for VDP applications, the use of PPML/VDX is recommended instead due to the workflow advantages its features enable, as well as its support by the NexPress front end's JDF Portal job submission interface **J** (section 3.4.3.2). PDM has been deprecated; support may be removed in future releases of the NexPress front end.

5 Press Management

The NexPress front end supports press management using the NexPress Client and the JMF Management Interface (M). Not all management functions are available through both interfaces. The NexPress Client provides a more complete interface; all press management functions are supported except those specifically controlling the JDF Queue. It is described in Section 3.4.2. The JMF Interface is primarily limited to control of the JDF Queue and jobs submitted through the JDF Portal. It is defined in Section 3.4.3.4.

Chapter 5 of the JDF specification [JDF] provides a complete description of JMF and its use. Because the NexPress does not support all the messages and their attributes defined in [JDF], refer to [KNDIR] for details of the syntax and semantics of the NexPress front end's conformance to JMF. It is also important to note that the NexPress only provides limited JMF subscription support. Unless otherwise noted, it should be assumed that subscription to specific queries is not available.

5.1 Configuring a Virtual Printer Job Ticket Template

The NexPress Client provides the only way to configure a Virtual Printer Job Ticket Template. The Job Ticket Editor (JTE) within the NexPress Client supports creation and management of Virtual Printer Hot Folders. The JTE specifies settings for a Virtual Printer Job Ticket Template. These settings include media selection and the mapping of a PDF stamp annotation-identified medium to an entry in theNexPress front end's media catalog, imposition layout, disjointing, and collating. The Virtual Printer Job Ticket Template is a complete workflow process specification; content files submitted to a Virtual Printer Hot Folder do not require additional job ticket parameterization as well.

5.2 Managing Press Capabilities

JMF provides a query to obtain a list of supported JMF commands.

Press capabilities cannot be modified through the JMF Management Interface. Where applicable, capabilities can be modified using the NexPress Client. The JMF Management Interface provides a proprietary mechanism for obtaining the NexPress Device Capabilities File; however, the NexPress does not provide support for the DevCaps element defined in the JDF 1.3 Specification [JDF]

5.2.1 Obtaining a list of KnownMessages

To obtain a list of JMF messages supported by the NexPress JDF Portal, a JDF Client should send a JMF **KnownMessages** query. The response lists the JMF commands and queries supported by the JDF Portal implementation.

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2008-01-16T16:17:31-
    05:00" Version="1.1">
    <Query ID="m080116161731_9366" Type="KnownMessages" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xsi:type="QueryKnownMessages"/>
</JMF>

…

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="KP-GDF00" TimeStamp="2008-01-16T16:20:19-
    05:00" Version="1.3">
    <Response ID="m080116162019_0043" ReturnCode="0" Type="KnownMessages"
    refID="m080116161731_9366" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ResponseKnownMessages">
        <MessageService Command="true" Type="CloseQueue"/>
        <MessageService Command="true" Type="HoldQueue"/>
        <MessageService Command="true" Type="OpenQueue"/>
```

```
            <MessageService Command="true" Type="ResumeQueue"/>
            <MessageService Command="true" Type="AbortQueueEntry"/>
            <MessageService Command="true" Type="HoldQueueEntry"/>
            <MessageService Command="true" Type="RemoveQueueEntry"/>
            <MessageService Command="true" Type="ResumeQueueEntry"/>
            <MessageService Command="true" Type="SubmitQueueEntry"/>
            <MessageService Command="true" Type="SuspendQueueEntry"/>
            <MessageService Command="true" Type="StopPersistentChannel"/>
            <MessageService Query="true" Type="KnownMessages"/>
            <MessageService Query="true" Type="QueueStatus"/>
            <MessageService Query="true" Type="Resource"/>
            <MessageService Persistent="true" Query="true" Signal="true" Type="Status"/>
            <MessageService Query="true" Type="SubmissionMethods"/>
            <MessageService Query="true" Type="NXP:DeviceCapabilities"/>
        </Response>
    </JMF>
```

5.2.2 Querying Supported Submission Methods

The JMF SubmissionMethods query is used primarily to determine the JDF Portal Hot Folder location. The hot folder location is \\<servername>\HotFolder\JDFHotfolder. The server's response message also indicates support for MIME and URLSchemes. Refer to [JDF] for further details.

Example: Supported Submission Methods query

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2008-01-16T16:20:19-
    05:00" Version="1.1">
    <Query ID="m080116162019_9368" Type="SubmissionMethods"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="QuerySubmissionMethods"/>
</JMF>

…

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="KP-GDF00" TimeStamp="2008-01-16T16:22:43-
    05:00" Version="1.3">

    <Response ID="m080116162243_0043" ReturnCode="0" Type="SubmissionMethods"
    refID="m080116162019_9368" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ResponseSubmissionMethods">
        <SubmissionMethods HotFolder="file://KP-GDF00/HotFolder/JDFHotfolder" Packaging="MIME"
        URLSchemes="file http ftp"/>
    </Response>

</JMF>
```

5.2.3 Querying for the Device Capabilities File

The Device Capabilities File content can be retrieved using the NexPress front end JDF Portal JMF Interface. Refer to section 3.4.4.1 and [KNDIR] for more detail on the Device Capabilities File.

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2008-01-16T16:24:22-
    05:00" Version="1.1">
    <Query ID="m080116162422_9369" Type="NXP:DeviceCapabilities"/>
</JMF>

…

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="KP-GDF00" TimeStamp="2008-01-16T16:26:48-
    05:00" Version="1.3">

 <Response ID="m080116162648_0043" ReturnCode="0" Type="NXP:DeviceCapabilities"
    refID="m080116162422_9369">
        <ManufacturingCapabilities>
            <DeviceID>KP-GDF00</DeviceID>
            <PrinterList>
                <PrinterDefaults>
                …
                </SubstrateDeliveryList>
                <PageOrderList>
                    <PageOrder Enum="FaceDown"/>
                    <PageOrder Enum="FaceUp"/>
                    <PageOrder Enum="FaceDownReverseOrder"/>
                    <PageOrder Enum="FaceUpReverseOrder"/>
                </PageOrderList>
            </FinishingOptions>
        </ManufacturingCapabilities>
    </Response>
</JMF>
```

5.3 Accessing press status

5.3.1 Retrieving press status

Status of the NexPress digital production color press is obtained using the JMF **Status** command. The amount of detail in the response is controlled by the values of ***StatusQuParams/@DeviceDetails*** ("None", "Brief", or "Full"), ***StatusQuParams/@JobDetails***("None", "Brief", or "Full"), and ***StatusQuParams/@QueueInfo*** ("false", or "true"). If omitted, an attribute will use its default value; refer to the JDF Specification [JDF] for further information.

Example: Full device details, brief job details, include queueInfo

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2008-01-17T10:22:08-
    05:00" Version="1.1">
    <Query ID="m080117102208_9371" Type="Status" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="QueryStatus">
        <StatusQuParams DeviceDetails="Full" JobDetails="Brief" QueueInfo="true"/>
    </Query>
</JMF>

…

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="KP-GDF00" TimeStamp="2008-01-17T11:31:33-
    05:00" Version="1.3">

    <Response ID="m080117113133_0043" ReturnCode="0" Subscribed="false" Type="Status"
    refID="m080117102208_9371" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ResponseStatus">
        <DeviceInfo CounterUnit="Sheets" DeviceStatus="Running">
            <Device Class="Implementation" DeviceID="KP-GDF00" DeviceType="NexPress 10.01.00 BLD0159 /
            1300 OFFICIAL" Manufacturer="Kodak" ModelName="NexPress_NPP"/>
            <JobPhase JobPartID="n080117102322_8468" PercentCompleted="100" QueueEntryID="KP-
            GDF00_565135" StartTime="2008-01-17T11:31:07-05:00" Status="InProgress"
            StatusDetails="RIP'ing">
                <Comment Language="en" Name="NxpPreflightWarning">Font: Availability|Helvetica-Bold|Not
                embedded|Embed fonts completely</Comment>
            </JobPhase>
        </DeviceInfo>
```

```
        <Queue DeviceID="KP-GDF00" Status="Running">
            <QueueEntry JobPartID="n080117102322_8468" xmlns:NXP=www.nexpress.com
            NXP:CDFEJobID="566846" Priority="50" QueueEntryID="KP-GDF00_565135" StartTime="2008-01-
            17T11:31:07-05:00" Status="Running" SubmissionTime="2008-01-17T11:31:06-05:00"/>
        </Queue>
    </Response>

</JMF>
```

The response in the above example indicates that the JDF Portal Job Queue has a status of ***Running***, and contains a single job identified by ***QueueEntryID***=*KP-GDF00_565135*. ***NXP:CDFEJobID*** provides the corresponding id of this job on the NexPress Client. It is important to note that ***DeviceInfo/@DeviceStatus*** does not correspond to the current operating state of the press, but instead indicates whether any jobs submitted through the JDF Portal Interface are in the Internal Job Queue.

5.3.2 Subscribing to press status

The NexPress provides JMF Subscription support for the Status Query. To establish a subscription, a JDF Client creates a persistent channel connection over which the NexPress front end can send asynchronous JMF Status messages. These asynchronous messages are known as JMF Signal messages. JMF Signal messages are generated and posted to the persistent channel when there is a change in the status of the JDF Portal Job Queue or in the execution status of a JDF Portal Queue Entry.

To create a persistent channel, a **Subscription** sub-element is added to the Status Query. ***Subscription/@URL*** defines where JMF **Signal** messages are to be posted. URL must use an *http:* scheme, a port number is optional. The persistent channel is assigned a ***refID*** that matches ***Query/@ID*** to which the subscription was attached. This ***refID*** is required to unsubscribe and close the persistent channel.

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2008-01-17T12:39:10-
    05:00" Version="1.2">
    <Query ID="m080117123549_9170" Type="Status" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="QueryStatus">
        <StatusQuParams DeviceDetails="Full"/>
    <Subscription URL="http://KP-GDF00::40000/"/>
    </Query>
</JMF>

… [RESPONSE] …

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="KP-GDF00" TimeStamp="2008-01-17T12:44:32-
    05:00" Version="1.3">

    <Response ID="m080117124432_0043" ReturnCode="0" Subscribed="true" Type="Status"
    refID="m080117123549_9170" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ResponseStatus">
        <DeviceInfo CounterUnit="Sheets" DeviceStatus="Running">
            <Device Class="Implementation" DeviceID="KP-GDF00" DeviceType="NexPress 10.01.00 BLD0159 /
            1300 OFFICIAL" Manufacturer="Kodak" ModelName="NexPress_NPP"/>
            <JobPhase JobPartID="n080117102322_8468" PercentCompleted="100" QueueEntryID="KP-
            GDF00_567598" StartTime="2008-01-17T11:40:38-05:00" Status="Stopped" StatusDetails="RIP'ing"/>
        </DeviceInfo>
    </Response>

</JMF>

… [SIGNAL] …

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="KP-GDF00" TimeStamp="2008-01-17T12:44:32-
    05:00" Version="1.3">

    <Signal ID="m080117124432_0044" Type="Status" refID="m080117123549_9170"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="SignalStatus">
        <DeviceInfo CounterUnit="Sheets" DeviceStatus="Running">
            <Device Class="Implementation" DeviceID="KP-GDF00" DeviceType="NexPress 10.01.00 BLD0159 /
            1300 OFFICIAL" Manufacturer="Kodak" ModelName="NexPress_NPP"/>
            <JobPhase JobPartID="n080117102322_8468" PercentCompleted="100" QueueEntryID="KP-
            GDF00_567598" StartTime="2008-01-17T11:40:38-05:00" Status="Stopped" StatusDetails="RIP'ing"/>
        </DeviceInfo>
    </Signal>

</JMF>
```

The example JMF **Query** message above creates a persistent channel for a Status Query with DeviceDetails="Full". ***Response/@Subscribed*** confirms that the persistent channel was created. The NexPress front end will post asynchronous JMF **Signal** messages to the URL *http://KP-GDF00::40000* over the persistent channel, and identify these signals with a RefID of *m080117123549_9170*. An initial JMF **Signal** message is posted when the channel is opened. The NexPress does not maintain a mapping of persistent channels with the originating query; all generated signals are sent to every open persistent channel regardless of the subscription request.

5.3.3 Unsubscribing to press status

To close a persistent channel, the JMF **StopPersistentChannel** command is sent to the JMF Management Interface. The value of ***StopPersChParams/@URL*** must match the URL of the original subscription; the value of ***StopPersChParams/@ChannelID*** must match refID assigned to the channel.

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2008-01-17T12:41:45-
    05:00" Version="1.2">
    <Command ID="m080117123549_9170" Type="StopPersistentChannel"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CommandStopPersistentChannel">
        <StopPersChParams ChannelID="m080117123549_9170" URL="http://KP-GDF00::40000/"/>
    </Command>
</JMF>
```

6 Job Submission

The NexPress front end offers two job submission interfaces: the Virtual Printer Hot Folder and the JDF Portal Interface. These interfaces are described in section 3.4.3. The choice of job submission interface for a given job depends upon workflow requirements, available workflow tools and Workflow Controller support. The Virtual Printer Hot Folder and the JDF Portal Interface provide different levels of job ticket and PDL support.

6.1 Submitting to the Virtual Printer Hot Folder

The following table summarizes the Structured Document Formats that can be submitted through the Virtual Printer Hot Folder. VPJTT stands for Virtual Printer Hot Folder Job Ticket Template.

Format	PDL	Job Ticket	Page Composition	Media Identification
PDF, PDF/VT	PDF	PJTF/VPJTT	PDF	VPJTT
PDF, PDF/VT (Stamp Annotated)	PDF	PJTF/VPJTT	PDF	PDF Stamp Annotations
Single/Multiple File PPML/VDX	PDF	VPJTT	PPML/VDX	VPJTT
PostScript	PS	VPJTT	PDF (produced from PS upon submission)	VPJTT
Single/Multiple File PPML/VDX (with embedded JDF MediaIntent, or embedded JDF Process)	PDF	Embedded JDF (MediaIntent or Process)	PPML/VDX	JDF Media Intent or JDF Process
PDF	PDF	Embedded JDF Process	PDF	JDF Process

The table above identifies the PDL, job ticket, page composition and media identification for each Structured Document Format. A PJTF job ticket can be embedded in the PDL for table entries containing PJTF/VPJTT Job Tickets; the embedded ticket will be merged with the VPJTT.

6.1.1 PDF & PDF/VT

Adobe® PDF files can be submitted directly to the Virtual Printer Hot Folder interface. Each file submission constitutes a single print run. Job ticket settings are determined as follows:

- Merging of PJTF settings embedded in the PDF file with the Virtual Printer Job Ticket Template. Embedded PJTF settings take precedence over a corresponding value from the VPJTT. Refer to [KNDIR] for specific merge rules.
- If the submitted PDF file doesn't contain embedded PJTF, all job ticket settings are obtained from the Virtual Printer Job Ticket Template. Support for embedded PJTF may be deprecated in future releases. NexPress extensions to PJTF are not documented in this interface guide, or in [KNDIR]. Creation of a JDF job ticket, and submission to the JDF Portal interface, is recommended as an alternative to embedded PJTF. A conversion tool has been created for internal use that converts NexPress PJTF tickets into JDF; customer availablity of this conversion library has not been determined. Contact your support representative if you need to convert existing PJTF tickets into JDF.

6.1.2 PDF & PDF/VT with Stamp Annotations

The Virtual Printer Hot Folder interface accepts direct submission of Adobe® PDF files that contain stamp annotations for media usage; each file submission constitutes a single print run. Job ticket settings are determined as follows:

- Merging of PJTF settings embedded in the PDF file with the settings specified in the Virtual Printer Job Ticket Template. Embedded PJTF settings take precedence over a corresponding value from the VPJTT. Refer to [KNDIR] for specific merge rules.
- If the submitted PDF file doesn't contain embedded PJTF, all job ticket settings are obtained from the Virtual Printer Job Ticket Template.
- The logical to physical media mapping of PDF stamp annotations is specified by the combined VPJTT and embedded PJTF. Non-stamp annotated pages of a PDF file are treated by the NexPress front end as implicitly mapped to *Body* media. Consequently, a single-substrate PDF-based job does not need to use PDF stamp annotation because all its pages implicitly map to *Body* media. The use of explicit *Cover* and *Insert* PDF stamp annotations is only necessary when a second or third media is required.

6.1.3 PostScript

Adobe® PS files can be submitted directly to the Virtual Printer Hot Folder interface. Each file submission constitutes a single print run. All PostScript data is converted to PDF within the NexPress front end. In this way, all of the benefits of a PDF based job ticketed workflow are available for jobs initially encoded in PostScript. Prepress processing of the PostScript data is subject to settings specified in the Virtual Printer Job Ticket Template associated with the Virtual Printer Hot Folder. Embedded PJTF and PDF Stamp annotations are not used with PostScript.

6.1.4 Single/Multiple-File PPML/VDX

A PPML/VDX-Layout file (either Single-File PPML/VDX or Multiple-File PPML/VDX formats) can be submitted directly to the Virtual Printer Hot Folder interface. Each file submission constitutes a single print run. Job ticket settings are specified in the Virtual Printer Job Ticket Template or from the embedded JDF Process Job Ticket. Embedded PJTF is ignored for PPML/VDX submissions.

Submission into the JDF Portal Interface with a fully specified JDF Process job ticket is preferred. Alternatively, JDF Process or JDF MediaIntent can be embedded into the PPML/VDX and submitted into a Virtual Printer Hot Folder.

Example: PPML/VDX

This example shows a Single-File PPML/VDX. Each **JOB** element shown contains three **DOCUMENT** elements, and each **DOCUMENT** element specifies the reader-ordered content pages.

```
%PDF-1.5
…
587 0 obj
<< /Length … >>
stream
<PPMLVDX>
    <ContentBindingTable> … </ContentBindingTable>
    <Layout>
        <PPML>
            …
            <PAGE_DESIGN TrimBox="0 0 612 792" BleedBox="-18 -18 630 810"/>
            <JOB Label="0">
                <DOCUMENT Label="FCover">
                    <PAGE0/>
                    <PAGE1/>
                </DOCUMENT>
                <DOCUMENT Label="Body">
                    <PAGE2/>
                    <PAGE3/>
                    <PAGE4/>
                    …
```

```
                    <PAGEm/>
                </DOCUMENT>
                <DOCUMENT Label="BCover">
                    <PAGEm+1/>
                    <PAGEm+2/>
                </DOCUMENT>
            </JOB>
            …
            <JOB Label="3"">
                <DOCUMENT Label="FCover">
                    <PAGE0/>
                    <PAGE1/>
                </DOCUMENT>
                <DOCUMENT Label="Body">
                    <PAGE2/>
                    <PAGE3/>
                    <PAGE4/>
                    …
                    <PAGEn/>
                </DOCUMENT>
                <DOCUMENT Label="BCover">
                    <PAGEn+1/>
                    <PAGEn+2/>
                </DOCUMENT>
            </JOB>
            …
        </PPML>
    </Layout>
</PPMLVDX>
endstream
endobj
…
%%EOF
```

6.1.5 Single/Multiple-File PPML/VDX with JDF MediaIntent

A PPML/VDX-Layout file (either Single-File PPML/VDX or Multiple-File PPML/VDX formats) containing embedded JDF MediaIntent can be submitted directly to the Virtual Printer Hot Folder interface. Each file submission constitutes a single print run. Job ticket settings are determined as follows:

- Merging of **MediaIntent** resources in JDF Product Intent with the settings specified in the Virtual Printer Job Ticket Template. **MediaIntent** resource settings take precedence over the media settings in the VPJTT. The value of attribute ***MediaIntent/StockBrand/@Actual*** attribute identifies the logical to physical media mapping. Refer to [KNDIR] for more detail.
- If the submitted PDF file doesn't contain embedded JDF **MediaIntent**, all job ticket settings are obtained from the Virtual Printer Job Ticket Template.
- Virtual Printer Job Ticket Template settings MUST provide appropriate default process parameterization to insure proper processing, since the NexPress front end uses only the **MediaIntent** resource. In general, the VPJTT specifies imposition layout, sheet-loading parameters (i.e. page distribution scheme, etc…), collation, output stack disjointing requirements, and the names of the physical medias corresponding to each of the three possible logical medias.

Example: Embedded JDF MediaIntent

The following example contains an embedded JDF Product Intent job ticket that could be used with the previous PPML/VDX example.

```
…
20 0 obj
<< /Length 21 0 R >>
```

```
stream
<?xml version="1.0" encoding="UTF-8"?>
<JDF ID="Booklet" Status="Waiting" Type="Product" Version="1.3" xmlns="http://www.CIP4.org/JDFSchema_1_1">
<ResourcePool>
    …
    <MediaIntent Class="Intent" ID="CoverMedia" Status="Available">
        <MediaType DataType="EnumerationSpan" Actual="Paper"/>
        <FrontCoatings DataType="EnumerationSpan" Actual="Glossy"/>
        <StockBrand DataType="StringSpan" Actual="Media1"/>
    </MediaIntent>
    <MediaIntent Class="Intent" ID="BodyMedia" Status="Available" >
        <MediaType DataType="EnumerationSpan" Actual="Paper"/>
        <FrontCoatings DataType="EnumerationSpan" Actual="Matte"/>
        <StockBrand DataType="StringSpan" Actual="Media0"/>
    </MediaIntent>
    …
</ResourcePool>
…
</ResourceLinkPool>
    …
    <MediaIntentLink Usage="Input" rRef="CoverMedia"/>
    <MediaIntentLink Usage="Input" rRef="BodyMedia"/>
    …
</ResourceLinkPool>
…
endstream
endobj
…
```

6.2 Submitting to the JDF Portal Interface

The following table summarizes the Structured Document Formats that can be submitted through the JDF Portal Interface.

Format	PDL	Job Ticket	Page Composition	Media Identification
JDF Process JT (PDF, PDF/VT)	PDF PDF/VT	JDF Process	PDF, PDF/VT	JDF Process
JDF Process JT (Stamp Annotated PDF, PDF/VT)	PDF, PDF/VT	JDF Process	PDF, PDF/VT	PDF Stamp Annotations and JDF Process
JDF Process JT (Single / Multiple file PPML/VDX)	PDF	JDF Process	PPML/VDX	JDF Process

The JDF Portal Interface provides both an HTTP port and a hot-folder to receive submission requests.

JDF Portal Hot Folder: \\<servername>\HotFolder\JDFHotfolder

JDF Portal JMF Interface: Port 49700 services HTTP Post requests on behalf of the JDF Portal JMF Interface

A JDF Job Ticket is required for submission through the JDF Portal Interface; a page content file or PPML layout file cannot be submitted directly. Instead the JDF Process job ticket must reference the content file in its **RunList/LayoutElement/FileSpec**.

The first Combined Process Node found within the root JDF element of the submitted job ticket file will be processed. There are no configurable default job ticket settings for the JDF Portal Hot Folder; the JDF Process job ticket should include all required job settings.

The NexPress will resolve all dependent data resources, including any referenced content files. Job submission will fail, and printing will be aborted, if a content file cannot be resolved.

6.2.1 The JDF Portal JMF Interface

The NexPress front end's JDF Portal will accept JDF job tickets through its JDF Portal JMF Interface. JMF Submissions use the SubmitQueueEntry command; the JDF Process Job Ticket is usually packaged as a MIME attachment but URIs of "HTTP:", "FTP:", or "FILE:" scheme can also be used.

Content files can also be MIME encoded; these three part MIME packages contain a JMF **SubmitQueueEntry** command as the first part, a JDF Process job ticket as the second part, and PDF formatted file (i.e. PDF or PPML/VDX) as the third part. The MIME header for the content file attachment should specify "Content-Disposition: Attachment" and "Content-Type: application/pdf"; failure to do so limits the size of the attachment to several hundred megabytes. The content filename can also be specified with the disposition; e.g. Content-Disposition: Attachment; filename="test.pdf".

The JDF Portal assigns each submission a unique JDF Portal Queue Entry ID; this ID is returned in the SubmitQueueEntry response. The JDF Portal Queue Entry ID is used for job queue management, it can also be used to track job progress.

Example: SubmitQueueEntry using two-part MIME

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary=-----967a0f3d165acc20
-------967a0f3d165acc20
Content-Type: application/vnd.cip4-jmf+xml
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="MessageSenderDevice" TimeStamp="2006-06-
    19T10:44:06-04:00" Version="1.3">
    <Command ID="m060619104406_0022" Type="SubmitQueueEntry"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CommandSubmitQueueEntry">
        <QueueSubmissionParams Priority="2" URL="cid:jdfTag@kodak.com"/>
    </Command>
</JMF>
-------967a0f3d165acc20
Content-Type: application/vnd.cip4-jdf+xml
Content-ID: <jdfTag@kodak.com>
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n060619104402_0007" Status="Ready" Type="Combined"
    Types="LayoutPreparation Imposition ColorSpaceConversion Interpreting Rendering Screening DigitalPrinting"
    Version="1.3" xmlns:NXP="www.nexpress.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="Combined">
…
    <ResourcePool>
        <ScreeningParams Class="Parameter" ID="r060619104402_0008" NXP:ScreeningID="1"
    Status="Available"/>
        <Component Amount="1" Class="Quantity" ComponentType="PartialProduct" ID="r060619104402_0009"
        Status="Available"/>
        <RunList Class="Parameter" ID="r060619104402_0010" Status="Available">
            <LayoutElement>
                <FileSpec URL=file://KP-GDF00/TestFiles/develop/onepage.pdf
                UserFileName="BC Plus_2005.pdf"/>
            </LayoutElement>
        </RunList>
…
</JDF>
-------967a0f3d165acc20—
```

6.2.2 The JDF Portal Hot Folder

The NexPress front end's JDF Portal will accept JDF job tickets through its JDF Portal Hot Folder. JDF job tickets can be copied into the hot folder, or placed into the folder using a simple drag and drop operation. No JMF response

is sent for jobs submitted to the hot folder. Signals will, however, be triggered by jobs submitted to the hot folder, and are sent to any open persistent channels.

6.2.3 Locating Content Files

The JDF Specification [JDF] provides several options for the location of content files and how they are specified within the JDF Job Ticket. ***JDF/ResourcePool/RunList/LayoutElement/FileSpec/@URL*** identifies within the JDF Job Ticket where the file is located. URL schemes of "FILE:", "FTP:", and "HTTP:" are supported. "CID:" is also supported for MIME packaged submissions using a MIME file to the JDF Portal Hot Folder, or a 3-part MIME with JMF SubmitQueueEntry.

There are several additional issues that need to be considered when a URL scheme of "FILE:" is used:

- The JDF Portal process within the NexPress front end must have appropriate permissions to access the specified file. This is not a problem if the content file is located on a directory or local share. It can become problematic in some networks if the NexPress is not part of the domain. When retrieving content from a remote server it is recommended to use the "HTTP:" URL scheme to avoid such network issues.
- Content files are not removed when the JDF Job has completed unless **Runlist/Disposition** is used.

6.2.3.1 Auto-Deletion of Content Files

FileSpec/Disposition controls automatic deletion of content files. By default, the content file is preserved.

To enable automatic deletion, set ***Disposition/@DispositionAction***="Delete". Only content files using a URL scheme of "FILE:" local to the DFE will be removed. When auto-deletion is enabled, the content file will be removed as the job finishes processing. ***Disposition/@Until*** can be used to specify an alternate removal time. The value of this attribute must be specified using ISO 8601 Date/Time format (i.e. yyyy-mm-ddThh:mm:ss-tz:tz). The file will be deleted within 5 minutes of the "Until" time provided.

6.2.4 Media Selection

Any JDF or PJTF embedded in the PDF file is ignored for submissions through the JDF Portal interface.

To assign media within a JDF file, the Media element can be partitioned by ***SheetIndex*** and/or ***RunTags***, or an explicit LocationName value can be used with an unpartitioned Media element. See Section 4.1.1 for details on specifying media using PDF stamp annotations. These annotations can be used with ***RunTags*** to assign pages to specific media.

6.2.4.1 Media/Location/@LocationName

To define media using ***Media/Location/@LocationName***, Media elements are defined and given LocationName values of Media0, Media1, Media2, or Media3. These LocationNames identify the four "logical trays" of the NexPress respectively known as "Body", "Cover", "Insert", and "Separator". With engine configurations of more than four supplies, "Media4", "Media5", …, "Media10" can be used to identify logical trays containing "Insert2", "Insert3", … "Insert8" respectively.

The NexPress can infer values of **Location/@LocationName** from **DigitalPrintingParams** or **Media** partitioning if only Body and Cover media are used. For all other jobs, **Location/@LocationName** must be defined for each **Media**.

Example: Media Selection Using LocationName

In the following example, body pages will use SterlingUltraDigitalGloss_118_350x470, cover pages will use Wausau_Bright_White_350_350x470, and insert pages will use AquaTEC_Permanent_Uncoated_237_350x470.

```
<Media Brand="SterlingUltraDigitalGloss_118_350x470" Class="Consumable" Dimension="992.11 1332.262"
   ID="r060619104402_0011" NXP:ClearCoatBack="none" NXP:ClearCoatFront="none" Status="Available">
      <Location LocationName="Media0"/>
</Media>
<Media Brand="Wausau_Bright_White_350_350x470" Class="Consumable" Dimension="992.11 1332.262"
   ID="r060619104402_0012" NXP:ClearCoatBack="none" NXP:ClearCoatFront="none" Status="Available">
      <Location LocationName="Media1"/>
</Media>
```

```
<Media Brand="AquaTEC_Permanent_Uncoated_237_350x470" Class="Consumable" Dimension="992.11
  1332.262" ID="r060619104402_0013" NXP:ClearCoatBack="none" NXP:ClearCoatFront="none"
  Status="Available">
      <Location LocationName="Media2"/>
</Media>
```

6.2.4.2 DigitalPrintingParams/@SheetIndex

To define media using ***DigitalPrintingParams/@SheetIndex***, the DigitalPrintingParams should be partitioned using a SheetIndex that corresponds to the assigned sheet number. For example, to specify cover media, a SheetIndex of 0 is the front cover, -1 is the back cover, and a range of 1 ~ -2 is all the sheets in between. When Media is partitioned or DigitalPrintingParams contains a MediaRef, the value of LocationName has no special significance.

Example: Media Selection Using SheetIndex

In the following example, body pages use SterlingUltraDigitalGloss_118_350x470, front cover pages use Wausau_Bright_White_350_350x470, and back cover pages use AquaTEC_Permanent_Uncoated_237_350x470.

```
<Media Brand="SterlingUltraDigitalGloss_118_350x470" Class="Consumable" Dimension="992.11 1332.262"
  ID="MediaForBody" NXP:ClearCoatBack="none" NXP:ClearCoatFront="none" Status="Available">
 <Location LocationName="Media0"/>
</Media>
<Media Brand="Wausau_Bright_White_350_350x470" Class="Consumable" Dimension="992.11 1332.262"
  ID="MediaForFrontCover" NXP:ClearCoatBack="none" NXP:ClearCoatFront="none" Status="Available">
 <Location LocationName="Media1"/>
</Media>
<Media Brand="AquaTEC_Permanent_Uncoated_237_350x470" Class="Consumable" Dimension="992.11
  1332.262" ID="MediaForInsert" NXP:ClearCoatBack="none" NXP:ClearCoatFront="none" Status="Available">
 <Location LocationName="Media2"/>
</Media>
…
<DigitalPrintingParams  PartIDKeys="SheetIndex"… >
    <DigitalPrintingParams SheetIndex="0">
        <MediaRef rRef="MediaForFrontCover" />
    </DigitalPrintingParams>
    <DigitalPrintingParams SheetIndex="1 ~ -2">
        <MediaRef rRef="MediaForBody" />
    </DigitalPrintingParams>
    <DigitalPrintingParams SheetIndex="-1">
        <MediaRef rRef="MediaForInsert" />
    </DigitalPrintingParams>
</DigitalPrintingParams>
…

- or -

…
<Media ID="MyMedias" Class="Consumable" Status="Available" PartIDKeys="SheetIndex">
    <Media SheetIndex="0" Brand="Wausau_Bright_White_350_350x470">
        <Location LocationName="Media1"/>  <!- Media for Front Cover -->
    </Media>
    <Media SheetIndex="1 ~ -2" Brand="SterlingUltraDigitalGloss_118_350x470">
        <Location LocationName="Media0"/>  <!- Media for Body -->
    </Media>
    <Media SheetIndex="-1" Brand="AquaTEC_Permanent_Uncoated_237_350x470">
        <Location LocationName="Media2"/>  <!- Media for Back Cover / Insert -->
    </Media>
</Media>
```

6.2.4.3 DigitalPrintingParams/@RunTags

DigitalPrintingParams/@RunTags is used in the NexPress label methods (Stamp Annotation and PPML/VDX labels) for PDL page to media mapping.

When using Stamp Annotation, only the attribute values "SubstrateTypeCover" and "SubstrateTypeInsert" are valid RunTags values.

When using PPML/VDX labels, the attribute values must correspond to the arbitrary values of the PPML DOCUMENT/@Label attributes.
If the value of RunTags has no correspondence with a PDF stamp annotation or PPML DOCUMENT/@Label then the media partition is ignored.

Example: Media Selection Using RunTags

In the following example, body pages use SterlingUltraDigitalGloss_118_350x470, and front cover pages use Wausau_Bright_White_350_350x470.

```
  <Media Brand="SterlingUltraDigitalGloss_118_350x470" Class="Consumable" Dimension="992.11 1332.262"
    ID="MediaForBody" NXP:ClearCoatBack="none" NXP:ClearCoatFront="none" Status="Available">
   <Location LocationName="Media0"/>
  </Media>
  <Media Brand="Wausau_Bright_White_350_350x470" Class="Consumable" Dimension="992.11 1332.262"
    ID="MediaForCover" NXP:ClearCoatBack="none" NXP:ClearCoatFront="none" Status="Available">
   <Location LocationName="Media1"/>
  </Media>
…
<DigitalPrintingParams  PartIDKeys="RunTags"… >
     <MediaRef rRef="MediaForBody" />
     <DigitalPrintingParams RunTags="SubstrateTypeCover">
          <MediaRef rRef="MediaForCover" />
     </DigitalPrintingParams>
</DigitalPrintingParams>
…

- or -

…
<Media ID="MyMedias" Class="Consumable" Status="Available" PartIDKeys="RunTags">
     <Media RunTags="SubstrateTypeCover" Brand="Wausau_Bright_White_350_350x470">
          <Location LocationName="Media1"/>  <!- Media for Cover -->
     </Media>
</Media>
```

6.2.4.4 Media/@SheetIndex and Media/@RunTags

If partitioning by both ***SheetIndex*** and ***RunTags***, partition first by ***SheetIndex***, then by ***RunTags***. These PartIDKeys are used together to apply the combination of NexPress cover-mode and label methods of page-to-media mapping.

Example: Media selection examples using both SheetIndex and RunTags

```
…
<Media ID="MyMedias" Class="Consumable" Status="Available" Brand=" SterlingUltraDigitalGloss_118_350x470"
     PartIDKeys="SheetIndex RunTags">
     <Location LocationName="Media0"/> <!-- Body substrate default-->
     <Media SheetIndex="0 -1" Brand=" Wausau_Bright_White_350_350x470">
          <Location LocationName="Media1"/>  <!- Cover media -->
     </Media>
     <Media SheetIndex="1 ~ -2" >
          <Media RunTags="SubstrateTypeCover" Brand=" Wausau_Bright_White_350_350x470">
               <Location LocationName="Media1"/>
          </Media>
          <Media RunTags="SubstrateTypeInsert" Brand=" AquaTEC_Permanent_Uncoated_237_350x470">
               <Location LocationName="Media2"/>
          </Media>
     </Media>
```

```
</Media>
…
```

6.2.5 Single/Multiple PPML/VDX with JDF Process

The JDF Portal Interface does not process JDF Intent; consequently JDF MediaIntent cannot be used with PPML/VDX submitted through the JDF Portal. Instead the PPML/VDX must specify imposed production sheet surfaces directly. Production sheets are imposed sheet layouts in which the PPML **PAGE** elements describe the contents of complete impositions including sheet marks (not individual reader content pages). These sheet definitions are presented in the order they are to be printed.

In the JDF Process job ticket, PPML **PAGE** elements should be treated as imposed sheet surface definitions and not as a reader-ordered set of pages to be imposed. The **LayoutPreparationParams** resource usually will specify a one-up layout (***LayoutPreparationParams/@NumberUp***="*1 1*", and either one or two-sided printing. Sheet marks and the details of the imposition layout are specified as part of the PPML graphical page content, and PPML pages are directly mapped to the front and/or back surfaces of a substrate. This is known as **Plain Imposition.**

NOTE: Supplemental PPML/VDX-Content files from the **ContentBindingTable** (for Multiple-file PPML/VDX) are fetched as content data during the resource collection phase of job execution. The job is aborted if any content files cannot be resolved.

Example: Media selection using RunTag partitioning with PPML/VDX

```
<PPML ...>
...
    <JOB>
        <DOCUMENT Label="FCover">
            <PAGE> ... </Page>
            <PAGE> ... </Page>
        </DOCUMENT>
        <DOCUMENT Label="Body">
            <PAGE> ... </Page>
            <PAGE> ... </Page>
            <PAGE> ... </Page>
            <PAGE> ... </Page>
            ...
        </DOCUMENT>
        <DOCUMENT Label="BCover">
            <PAGE> ... </Page>
            <PAGE> ... </Page>
        </DOCUMENT>
    </JOB>
    ...
    <JOB>
        <DOCUMENT Label="FCover">
            <PAGE> ... </Page>
            <PAGE> ... </Page>
        </DOCUMENT>
        <DOCUMENT Label="Body">
            <PAGE> ... </Page>
            <PAGE> ... </Page>
            <PAGE> ... </Page>
            <PAGE> ... </Page>
            ...
        </DOCUMENT>
        <DOCUMENT Label="BCover">
            <PAGE> ... </Page>
            <PAGE> ... </Page>
        </DOCUMENT>
    </JOB>
</PPML>
```

```
…
<Media ID="MyMedias" Class="Consumable" Status="Available"  PartIDKeys="RunTags">
    <Media RunTags="FCover" Brand="Wausau_Bright_White_350_350x470">
        <Location LocationName="Media1"/>  <!- Cover media -->
    </Media>
    <Media RunTags="Body" Brand="SterlingUltraDigitalGloss_118_350x470">
        <Location LocationName="Media0"/> <!-- Body substrate -->
    </Media>
    <Media RunTags="BCover" Brand="Wausau_Bright_White_350_350x470">
        <Location LocationName="Media1"/>  <!- Cover media -->
    </Media>
</Media>
…
```

Example: PPML/VDX print range specified by set numbers

The following sample shows the specification of a print range using set numbers. Using zero-based counting, the instances 0 to 99 are to be printed from the total number of records in the VDX file. ***RunList/@Sets*** is supported only for VDX/PPML files because VDX instances are RIPped and printed independently.

```
…
<ResourcePool>
    <RunList ID="RLBooklet" Class="Parameter" Status="Available" ComponentGranularity="Set" Sets="0 ~ 99">
        <LayoutElement ElementType="Multiset">
            <FileSpec URL="File://KP-GDF00/TestFiles/develop/VDX/records10_pages10.vdx"/>
        </LayoutElement>
    </RunList>
</ResourceLinkPool>
…
<ResourceLinkPool>
    <RunListLink rRef="RLBooklet" CombinedProcessIndex="6" Usage="Input"/>
</ResourceLinkPool>
```

6.2.6 Identifying a JDF Portal job after submission

The JDF Portal assigns each submission a unique JDF Portal Queue Entry ID; this ID is returned in the SubmitQueueEntry response. A JDF Client must manage the association of each **QueueEntryID** value with a submitted job. This ID is required for control of the submitted job or to display job status. It is difficult to obtain this association for jobs submitted into the JDF Portal Hot Folder rather than through the JMF Interface. Jobs submitted into a Virtual Printer Hot Folder cannot be viewed or managed using the JDF Portal.

A job is assigned a second identifier upon submission to the Internal Job Queue. This ID, known as the NXP:CDFEJobID, is displayed on the NexPress Client. It is reported in the queueEntry of all JDF jobs that have begun processing.

The NexPress Client can be configured to identify jobs submitted through the JDF Portal Interface. By default, the JDF field is not shown in the Client's job list. If desired, change the preferences of one or more tabs in the Jobs screen to display the field. The JDF Portal Queue Entry ID cannot be displayed on the NexPress Client.

The JDF Job Ticket controls the "Job Name" and "File Name" fields in the NexPress Client.

"Job Name" is determined by selecting the first defined field in the JDF job ticket from the following prioritized list:

1. ***JDF/@DescriptiveName***
2. ***JDF/@JobID*** of the NexPress Combined Node. If ***JDF/@JobPartID*** is defined, a '-' is prepended to value of JobPartID and combined with ***JDF/@JobID***.
3. ***JDF/AncestorPool/Ancestor/@DescriptiveName***
4. ***JDF/AncestorPool/Ancestor/@JobID***. If ***JDF/ AncestorPool/Ancestor/@JobPartID*** is defined, a '-' is prepended to value of JobPartID and combined with ***JDF/AncestorPool/Ancestor/@JobID***.
5. ***JDF/@ID***

"File Name" is determined by selecting the first defined field in the JDF job ticket using following list. If more than one FileSpec is found in the JDF, file name is set using the first FileSpec:

1. ***JDF/ResourcePool/RunList/LayoutElement/FileSpec@UserFileName***
2. ***JDF/ResourcePool/RunList/LayoutElement/FileSpec@URL*** (pathname stripped)

6.2.7 Accessing completed JDF Portal Job detail

The NexPress updates the JDF AuditPool as the job is processed. There are two ways to obtain the completed JDF job ticket:

- The NexPress front end can be instructed to write the JDF file to a designated share folder or web server.
- The JDF file can be fetched from the JDF storage directory on the NexPress front end.

To write a completed JDF job ticket file to a designated location, include either ***JDF/NodeInfo/TargetRoute/@URL*** in the JDF Job Ticket or ***QueueSubmissionParams/@ReturnURL*** in the JMF SubmitQueueEntry. If both are supplied, ***ReturnURL*** takes precedence. To identify a web server, the URI has the format "HTTP://servername:PORT#/". To identify a file share that is writeable by the local system account, the URI has the format "FILE://servername/sharename/file."

If no other location is specified for the completed JDF ticket, it will be written to c:\HotFolder\JDFHotfolder\Unmerged. The filename of the completed JDF ticket written to the Unmerged directory is system-generated and cannot be programmatically determined from the JDF Portal interface; use an explicit TargetRoute or ReturnURL to facilitate matching completed JDF with its submitted ticket.

JDF Storage Directory: \\<servername>\HotFolder\JDFHotFolder\Unmerged

Example: Specifying TargetRoute in a JDF Job Ticket

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JDF xmlns="http://www.CIP4.org/JDFSchema_1_1" ID="n060621083401_0074" Status="Ready" Type="Combined"
    Types="LayoutPreparation Imposition ColorSpaceConversion Interpreting Rendering Screening DigitalPrinting"
    Version="1.3" xmlns:NXP="www.nexpress.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="Combined">
…
    <NodeInfo TargetRoute="file://KP-GDF00/ReturnJDF/ReturnJDF01.jdf"/>
</JDF>
```

Example: Specifying ReturnURL in JMF SubmitQueueEntry

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary=-----da2a52b369b4dffb
-------da2a52b369b4dffb
Content-Type: application/vnd.cip4-jmf+xml
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="MessageSenderDevice" TimeStamp="2006-06-
    29T14:53:57-04:00" Version="1.3">
    <Command ID="m060629145357_0022" Type="SubmitQueueEntry"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CommandSubmitQueueEntry">
        <QueueSubmissionParams Priority="2" ReturnURL=" file://KP-GDF00/ReturnJDF/ReturnJDF01.jdf "
        URL="cid:jdfTag@kodak.com"/>
    </Command>
</JMF>
…
```

7 Job Management

Only jobs submitted through the JDF Portal Interface can subsequently be managed using the JMF interface. Jobs submitted into a Virtual Printer Hot Folder cannot be viewed or managed using the JDF Portal, and the state of the JDF Portal Job Queue does not effect submission through the Virtual Printer Hot Folder.

7.1 JDF Portal Job Queue Management

The NexPress front end only supports management of the JDF Portal Queue using the JDF Portal JMF Interface. The portal supports the following queue management commands:

- Opening the JDF Portal Job Queue
- Closing the JDF Portal Job Queue
- Holding the JDF Portal Job Queue
- Resuming the JDF Portal Job Queue

The portal also supports the following queue handling query:

- Querying the JDF Portal Job Queue status

The JDF Specification [JDF] identifies this group as Global Queue Handling messages. Details can be found in [JDF]. Any JMF global queue management messages defined in [JDF], but not supported by the NexPress front end are ignored.

Multiple JDF clients may be accessing the JDF Portal at the same time. Obvious race conditions will exist if multiple JDF clients attempt to control the state of the same JDF Portal Job Queue. It is recommended that only a single master JDF client manage the JDF Portal Job Queue state; if multiple peer JDF clients submit files to the same NexPress front end, it is recommended that the clients coordinate their use of the NexPress front end as a shared resource.

Even though [JDF] provides a complete description of JMF and its use, refer to [KNDIR] for details on the syntax and semantics of the NexPress front end's conformance to JMF.

All queue management commands use the same syntax, varying only by ***JMF/Command/@Type*** and ***JMF/Command/@xsi:type***. A single example is provided as a reference for all queue management commands.

Example: Queue Management Commands (featuring OpenQueue)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2008-01-17T13:33:13-
    05:00" Version="1.2">
    <Command ID="m080117133313_9173" Type="OpenQueue" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xsi:type="CommandOpenQueue">
        <QueueFilter QueueEntryDetails="Brief"/>
    </Command>
</JMF>

…

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="KP-GDF00" TimeStamp="2008-01-17T13:34:17-
    05:00" Version="1.3">

    <Response ID="m080117133417_0043" ReturnCode="0" Type="OpenQueue" refID="m080117133313_9173"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ResponseOpenQueue">
        <Queue DeviceID="KP-GDF00" Status="Waiting"/>
    </Response>

</JMF>
```

7.1.1 Closing the JDF Portal Job Queue

The JDF Portal Job Queue must be in the *Waiting* state and ready to accept jobs prior to submission of a JDF job. The JDF Portal Job Queue defaults to the *Waiting* state (open) after initialization.

If the JDF Portal Job Queue is in a *Waiting* (i.e. opened) state, the JMF **CloseQueue** command will transition the job queue to a *Closed* state.

JDF job tickets submitted to the JDF Portal Hot Folder while the JDF Portal Job Queue is in a *Closed* state will be deleted. JDF job tickets submitted through the JDF Portal JMF Interface are rejected when the JDF Portal Job Queue is in a *Closed* state.

JMF/Command/@Type	CloseQueue
JMF/Command/@xsi:type	CommandCloseQueue

7.1.2 Opening the JDF Portal Job Queue

If the JDF Portal Job Queue is in a *Closed* state, the JMF **OpenQueue** command will transition the queue back to *Waiting*. While *Closed*, the JDF Portal Job Queue will not accept any job submissions from the JDF Portal Hot Folder or the JDF Portal JMF Interface. Any existing entries in the JDF Portal Job Queue will continue to execute, and all supported JMF commands other than **SubmitQueueEntry** may be used

JMF/Command/@Type	OpenQueue
JMF/Command/@xsi:type	CommandOpenQueue

7.1.3 Holding the JDF Portal Job Queue

The JMF **HoldQueue** command will transition the job queue to a *Held* state. While *Held*, no jobs will begin execution. **HoldQueue** has no effect on jobs that have begun processing. New entries may still be submitted to a *Held* queue, but they will not be processed until the queue has been resumed.

JMF/Command/@Type	HoldQueue
JMF/Command/@xsi:type	CommandHoldQueue

7.1.4 Resuming the JDF Portal Job Queue

If the JDF Portal Job Queue is in a Held state, the JMF **ResumeQueue** command will transition the job queue to a *Waiting* state.

JMF/Command/@Type	ResumeQueue
JMF/Command/@xsi:type	CommandResumeQueue

7.1.5 Querying the JDF Portal Job Queue status

JDF Portal Queue status is returned with the reponse from supported Queue Management Commands by including **<QueueFilter *QueueEntryDetails="Brief"*/>** as shown in the example. **QueueStatus** can also be queried directly without affecting the queue state.

Example: QueueStatus Query

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2008-01-17T13:38:54-
    05:00" Version="1.2">
    <Query ID="m080117133854_9174" Type="QueueStatus" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xsi:type="QueryQueueStatus">
        <QueueFilter QueueEntryDetails="Brief"/>
    </Query>
</JMF>

…

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="KP-GDF00" TimeStamp="2008-01-17T13:38:59-
    05:00" Version="1.3">
```

```
    <Response ID="m080117133859_0043" ReturnCode="0" Subscribed="false" Type="QueueStatus"
    refID="m080117133854_9174" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ResponseQueueStatus">
        <Queue DeviceID="KP-GDF00" Status="Waiting"/>
     </Response>

</JMF>
```

7.2 JDF Portal Job Entry Management

The NexPress front end only supports direct management of JDF Portal queue entries using the JDF Portal JMF interface. Any job in the internal Job Queue, including those submitted through the JDF Portal, can also be held (stopped), resumed, aborted or removed (deleted) using the NexPress Client. The portal supports the following queue entry management commands:

- Aborting a JDF Portal Job Entry
- Removing a JDF Portal Job Entry
- Holding a JDF Portal Job Entry
- Suspending a JDF Portal Job Entry
- Resuming a JDF Portal Job Entry

The following queue entry queries are also supported:

- Querying status of a JDF Portal Job Entry

The JDF Specification [JDF] identifies this group as Queue Entry Handling messages. Details can be found in [JDF]. Any JMF queue entry handling messages defined in [JDF], but not supported by the NexPress front end, are ignored.

Queue Entry Handling messages require a ***QueueEntryDef/@QueueEntryID***; the supplied value identifies the affected job.

Even though [JDF] provides a complete description of JMF and its use, refer to [KNDIR] for details on the syntax and semantics of the NexPress front end's conformance to JMF.

All Queue Entry Handling commands use the same syntax, varying only by ***JMF/Command/@Type*** and ***JMF/Command/@xsi:type***. A single example is provided as a reference for all queue entry handling commands.

Example: Queue Entry Handling Commands (featuring AbortQueueEntry)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2006-06-21T14:09:24-
    04:00" Version="1.1">
    <Command ID="m060621140924_0092" Type="AbortQueueEntry"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CommandAbortQueueEntry">
        <QueueFilter QueueEntryDetails="Brief"/>
        <QueueEntryDef QueueEntryID="Nxp2100@KP-GDF00_32"/>
    </Command>
</JMF>
```

7.2.1 Aborting a JDF Portal Job Entry

The JMF **AbortQueueEntry** command will abort the JDF Portal Queue Entry matching the value of ***Command/QueueEntryDef/@QueueEntryID***.

JMF/Command/@Type	AbortQueueEntry
JMF/Command/@xsi:type	CommandAbortQueueEntry

7.2.2 Removing a JDF Portal Job Entry

All jobs submitted through the JDF Portal remain in the JDF Portal queue and in the internal job queue until the job is removed. Jobs submitted with JMF **SubmitQueueEntry** use **QueueSubmissionParams/Disposition** to control automatic removal of the job upon completion. Jobs without this Disposition element are removed by the Portal when the job status becomes *Completed* or *Aborted*. Any job submitted through the JDF Portal Hot Folder will be automatically removed when it finishes processing.

The JMF **RemoveQueueEntry** command will remove the JDF Portal Queue Entry matching the value of ***Command/QueueEntryDef/@QueueEntryID***.

Only JDF Portal Queue entries with a status of *Waiting, Held,* or *Aborted* can be removed.

JMF/Command/@Type	RemoveQueueEntry
JMF/Command/@xsi:type	CommandRemoveQueueEntry

7.2.3 Holding a JDF Portal Job Entry

The JMF **HoldQueueEntry** command will transition to a *Held* status the JDF Portal Queue Entry matching the value of ***Command/QueueEntryDef/@QueueEntryID***.

Only JDF Portal Queue entries with a status of *Waiting* can be held.

JMF/Command/@Type	HoldQueueEntry
JMF/Command/@xsi:type	CommandHoldQueueEntry

7.2.4 Suspending a JDF Portal Job Entry

The JMF **SuspendQueueEntry** command will transition to a *Suspended* status the JDF Portal Queue Entry matching the value of ***Command/QueueEntryDef/@QueueEntryID***.

Only JDF Portal Queue entries with a status of *Running* can be suspended.

JMF/Command/@Type	SuspendQueueEntry
JMF/Command/@xsi:type	CommandSuspendQueueEntry

7.2.5 Resuming a JDF Portal Job Entry

The JMF **ResumeQueueEntry** command will transition to a *Waiting* or *Running* status the JDF Portal Queue Entry matching the value of ***Command/QueueEntryDef/@QueueEntryID***. A QueueEntry with a status of *Held* becomes *Waiting*; an entry with a status of *Suspended* becomes *Running*.

Only JDF Portal Queue entries with a status of *Held* or *Suspended* can be resumed.

JMF/Command/@Type	ResumeQueueEntry
JMF/Command/@xsi:type	CommandResumeQueueEntry

7.2.6 Retrieving status of a JDF Portal Job Entry

The JMF **QueueStatus** or **Status** queries can be used at any time to determine the status of a submitted job.

NOTE: The JDF Portal provides only marginal support for **QueueStatus**. To filter by a specific QueueEntryID, use **Status** rather than **QueueStatus**.

Example: Retrieving Status of a Specific QueueEntryID

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="NxpJdfTest" TimeStamp="2006-06-21T15:27:07-
    04:00" Version="1.1">
    <Query ID="m060621152707_0097" Type="Status" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="QueryStatus">
        <StatusQuParams QueueEntryID="Nxp2100@KP-GDF00_33" QueueInfo="true"/>
    </Query>
</JMF>

…

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

```
<JMF xmlns="http://www.CIP4.org/JDFSchema_1_1" SenderID="Nxp2100@KP-GDF00" TimeStamp="2006-06-
    21T15:37:49-04:00" Version="1.3">

  <Response ID="m060621153749_0856" Subscribed="false" Type="Status" refID="m060621152707_0097"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ResponseStatus">
    <DeviceInfo DeviceStatus="Running">
      <JobPhase JobID="" PercentCompleted="100" Status="InProgress" StatusDetails="RIP'ing"/>
    </DeviceInfo>
    <Queue DeviceID="Nxp2100@KP-GDF00" Status="Waiting">
      <QueueEntry xmlns:HDM="www.heidelberg.com/schema/HDM" HDM:ID="n060620161109_0045"
      HDM:JobPriority="50" HDM:NodeID="n060620161109_0045" JobID="" xmlns:NXP="www.nexpress.com"
      NXP:CDFEJobID="438848" Priority="50" QueueEntryID="Nxp2100@KP-GDF00_33" StartTime="2006-06-
      20T16:59:00-04:00" Status="Running" SubmissionTime="2006-06-20T16:59:00-04:00"/>
    </Queue>
  </Response>

</JMF>
```