# Portable Job Ticket Format

*Adobe Developer Support*

**Technical Note #5620**

**Version 1.1**
**2 April 1999**

**Adobe Systems Incorporated**

Corporate Headquarters
345 Park Avenue
San Jose, CA 95110-2704
(408) 536-6000

Eastern Regional Office
24 New England
Executive Park
Burlington, MA 01803
(617) 273-2120

Adobe Systems Europe Limited
Adobe House, Mid New Cultins
Edinburgh EH11 4DU
Scotland, United Kingdom
+44-131-453-221

Adobe Systems Japan
Yebisu Garden Place Tower
4-20-3 Ebisu, Shibuya-ku
Tokyo 150 Japan
+81-3-5423-8100

PN LPS5620

# Contents

# Portable Job Ticket Format

## 1 Introduction

This document describes the Portable Job Ticket Format (PJTF). PJTF provides a mechanism for specifying the instructions and the location of the contents needed to execute a print job. PJTF is based on the Portable Document Format (PDF), and the data representations allowed within PJTF are defined directly or indirectly in terms of PDF object types.

See Portable Document Format 1.2 Reference Manual for complete descriptions of recognized PDF object types.

A job ticket is a data representation which conforms to PJTF and which specifies the instructions and the location of the contents for one print job. Job tickets can exist as stand-alone files, or they can reside in the same file as a PDF document.

A primary use of PJTF is to prepare a device (typically a printer or imagesetter) to receive and process the files that describe a printed document in a high-level Page Description Language (PDL) such as PDF. The job ticket provides the device setup information, while the PDL describes the content to be marked on the output medium.

### 1.1 Job Ticket Objects

Job tickets are made up of job ticket objects which conform to the PDF dictionary object specification. Job ticket objects comprise sets of key/value pairs. Each key/value pair represents one attribute of the PJTF object.

Attribute keys are PJTF names.

In general, attribute values may be of any PJTF data type. The allowable types for any specific attribute value depend on the semantics of that attribute and of the PJTF object within which the attribute resides.

See section 1.4, “PJTF Data Types” for a description of the data types allowed within PJTF objects.

PJTF is not restricted to objects that directly relate to printing. It allows for a full description of a printing process which consists of several production steps.

For example, the **Preflight** (page 91), **Layout** (page 77), **Trapping** (page 66) and similar objects provide instructions for steps which may occur prior to printing; a wide variety of objects provide instructions and specify the location of content for printing; the **Finishing** (page 36) and **Delivery** (page 35) objects provide instructions for steps which may occur after printing; and the **Accounting** (page 38), **Audit** (page 25), **Scheduling** (page 37) and similar objects are provided to support the administration of the job throughout its life.

## 1.2 Document Organization

Most of the information in this specification is provided in a series of tables that define attributes of job ticket objects. Each table is preceded by text which describes the purpose and usage of the object, as well as an enumeration of the locations in the Job Ticket structure where the object is expected to occur.

For each object attribute, the key is specified, the allowable types for its value are listed, and a description of the usage and limitations of the attribute is provided. When the allowed values for the attribute can be specified in a enumeration, that enumeration is provided as part of the attribute description.

When the value for an attribute is another object, the tables in this specification provide cross references to help explain usage.

Some sections of this document refer to:

*PostScript® Language Reference, Third Edition*
Addison-Wesley, 1999, ISBN 0-201-37922-8

which is available from Adobe in PDF format:

http://partners.adobe.com/supportservice/devrelations/PDFS/TN/PLRM.pdf

## 1.3 Conventions used in this Specification

Text styles are used to identify objects, keywords and values.

- Job ticket objects and the names of keys used in those objects are written in boldface. Examples are **JobTicketContents**, **Layout** and **CTM**.

- Values which job ticket keys may take on are written in an italic sans serif font. Examples are *true* and *Device*.

The special convention **ObjectName**::**KeyName** is used to identify a key for a specific object. For example, **JobTicketContents**::**TrappingSourceSelector** and **PlacedObject**::**ClipBox**.

## 1.4 PJTF Data Types

PJTF supports 12 types for attribute values: boolea*n*, *number*, *name*, *array*, *dictionary*, *stream*, *rectangle*, *filespec*, *text*, *string*, *date* and *phone number*.

The definition of the PJTF data types: *boolean*, *number*, *name*, *array*, *dictionary*, *stream*, *rectangle* and *filespec* are identical to the PDF definitions of those types, and are not repeated here.

PJTF attributes of type *text* are PDF strings as described in the Portable Document Format 1.2 Reference Manual. The contents of text objects may be localized.

PJTF *strings* are PDF *strings* as described in the Portable Document Format 1.2 Reference Manual with specific restrictions applied: the string may not be PDFDocEncoded, and may not be localized. Strings are specified when systems will attempt to perform an exact match on the string.

PJTF *dates* are PDF *strings* which are formatted to conform to the definition of the date data structure in PDF.

PJTF *phone numbers* are represented as encoded strings, consistent with the URL standard proposed in Request for Comment 2303, "Minimal PSTN address format in Internet Mail." The RFC is available at:

http://info.internet.ini.edu:80/in-notes/info/files/rfc2303.txt

Note that when the value for a PJTF attribute is expected to be another PJTF object, this specification will list the type as *dictionary*.

## 1.5 Version 1.1 Update

Objects and attributes which have been added for version 1.1 of this specification are identified in the margin as shown here. PJTF 1.1

Objects and attributes which have been superseded for version 1.1 of this specification are identified in the margin as shown here.

PJTF 1.0

## 2 Terminology

Because job tickets adhere to the PDF specification, most of the terms used in discussing job tickets can be found in the *Portable Document Format Reference Manual*, version 1.0 and in it successor, the *Portable Document Format 1.2 Reference Manual*. See these publications for additional background information.

In addition, the following terms have specific meanings with regard to job tickets:

*Document* — One or more PDL files that produce one or more pages of printed output.

*Page Description Language* — (PDL) Any language which can be used to specify the contents and of pages which comprise print jobs.

*Page Coordinate Space* — The default (untransformed) coordinate space for the document. This is the coordinate space used by marking operations in the PDL to describe the page contents and layout. In the PostScript and PDF language specifications, this is referred to as *default user space*.

*Job, or print job* — The set of operations specified by a job ticket and its referenced documents.

*Job Ticket Manager* — A software entity that can generate job tickets.

*Job Ticket Processor*— A software entity that can consume job tickets and perform operations based on the settings found therein. Such a software entity might be embedded in a printing device.

*Device* — A machine, including hardware and software, that produces output from a print job. Typical devices are printers and imagesetters.

*Simple Printing* — A printing process in which pages are imaged directly onto media, one page per media surface. In this process, the job ticket provides the sequence of pages to be imaged.

## 3 Controlling the Printing Process

Job tickets have two substantial areas to address. First, the job ticket must specify the source material for the graphical content of each page. Second, the job ticket must specify the manner by which the page contents are imaged onto the media. These two areas are described in separate structures of the Job Ticket; the first is handled by the **Documents (**page 39) array, the second is handled by **Layout** (page 77) or **PrintLayout** (page 78) objects.

**Document** objects identify PDL files with content to render. The entire **Documents** array specifies a sequence of pages to image. Keys within the **Document** object provide information about the pages contained within those **Documents**, such as the dimensions of the pages, and the colorants used in the page descriptions.

**PageRange** objects are used to describe pages within **Documents** which differ in some way from those described by the **Document** object.

The **Layout** and **PrintLayout** object hierarchies are used to specify media and the imposition of page contents onto that media.

## 3.1 Specifying Page Dimensions

Three keys are provided which describe the dimensions of the PDL pages which make up the job: **MediaBox**, **TrimBox** and **BleedBox**. Each of these boxes is defined as a rectangle in page coordinate space.

Each of these keys may occur in **JobTicketContents** (page 27), **Document** (page 39), or **PageRange** (page 43) objects.

These boxes may be used by Job Ticket Processors, such as imposition engines, which add instructions to the job ticket to control the positioning of page contents onto media. Four example processes are described below. One or more of the boxes is used in each of these processes (except Simple printing):

- Simple printing
- Printing a finished page
- Printing an intermediate page
- Building an imposition

### Simple printing

This refers to a printing process in which there is no **Layout** or **PrintLayout** hierarchy. The **Documents** array specifies the sequence of pages to be printed, and the page contents are image directly onto the finished media. In this case, **MediaBox**, **TrimBox** and **BleedBox** are ignored.

### Printing a finished page

This refers to the typical desktop or shared page printer, where the finished media is most often fed to the printer and the page content is positioned on that media. The **TrimBox** best reflects this content, and this should be used for the value of the **PlacedObject**::**ClipBox** for the page.

### Printing an intermediate page

This refers to a process where the page is being output to some intermediate media (such as film) prior to the production of the finished page. Using the **MediaBox** as the value of the **PlacedObject**::**ClipBox** ensures that all printer's marks or other annotations of the original content are included on the page.

The **BleedBox** may be used to specify the clip region of the content that is to be imaged when the output process will generate printer's marks.

### Building an imposition

This refers to a process where several pages will be imaged per surface, and additional printer's marks relative to the full surface may be added (by means of **MarkDocuments**). Using the **BleedBox** as the value of **PlaceObject**::**ClipBox** ensures that only printer's marks pertaining to the final content is imaged.

The **TrimBox** specifies the location and size of the content in page coordinate space and may be used to generate the **PlacedObject**::**SourceClipPath**.

## 3.2 Layout and PrintLayout

PJTF 1.1

There are two mechanisms provided for controlling the flow of page images onto media: **Layout** and **PrintLayout**. **Layout** explicitly identifies all page content for each **Sheet** imaged and references these pages by means of the **Documents** and/or **MarkDocuments** array. **PrintLayout** is a template approach to printing and relies upon the full **Documents** hierarchy to specify the page content to image.

When neither **Layout**, nor **PrintLayout** instructions are provided, the job is produced via simple printing. See section 3.3, "Simple Printing" for more information.

**Layout** (page 77) objects specify an array of **Signature(s)** (page 81). Each Signature specifies an array of **Sheet(s)** (page 83), and each **Sheet** can have up to two **Surfaces** (**Front** and **Back**) (page 85), where the page images are to be placed using

**PlacedObjects** (page 86). A **Sheet** that specifies no **Surface** content will be blank. Pages which are to be printed must be placed onto **Surfaces** using **PlaceObject** objects which explicitly identify the document (via the **Doc** key) and page (via the **Ord** key). Thus, the **Layout** hierarchy specifies explicitly which pages will be imaged.

**PrintLayout** (page 78) objects specify a single **Signature** of **Sheet**(s) where page contents are imaged. The (virtual) sequence of pages which is to be imaged via **PrintLayout** is defined by the **PageRange** objects for the **Document** objects in the **JobTicketContents**::**Documents** array. Pages are drawn in order from this sequence to satisfy the **PlaceObject** objects in the **Surfaces** for the **Signature** in the **PrintLayout**, and the **Signature** is repeated until all pages of the sequence are consumed. Each time the **Signature** is repeated, pages are consumed in 'chunks' whose size is determined by the value of **PrintLayout**::**MaxOrd** + 1 (if present), or by the largest **Ord** value for any **PlacedObject** in the **Signature** (if **MaxOrd** is absent).

Since **PlacedObject**s are used repeatedly to draw pages out of the sequence of pages, they do not refer to specific pages (via the combination of the **Doc** and **Ord** keys, as for **Layout**). Instead, **Ord** is used to identify which page out of the next 'chunk' of pages is to be imaged on the **Surface**.

For example, to print all pages of a job two-up duplex, the **PrintLayout**::**Signature** might have one **Sheet** with two **Surface**s, each containing two **PlacedObjects**. In this case, the **PlacedObject**::**Ord** values would be 0, 1, 2 and 3, **PrintLayout**::**MaxOrd** would be 3 (or could be absent), and the contents would be processed four pages at a time.

Note that if **PrintLayout**::**MaxOrd** were 4 in the last example, pages would be consumed in chunks of five, and every fifth page would be skipped (that is, would not be printed).

Attributes of the **Media** are given for each **Sheet** used in printing. Because the same **Signature** is repeated until all pages are consumed, the **Documents** hierarchy can provide hints or preferences about special needs for sets of page content (via **InsertPage**) and whether alternate media should be inserted (via **NewSheet** and **Trailer**). Use of **InsertPage**, and **NewSheet** and **Trailer** objects will *only* affect a **PrintLayout Signature**. Inserted media is a means to separate sections of the document content. Alternate content would be printed only as necessary to fill areas which would normally have page content, due to insertion of new media or to designating where a document section will begin (odd or even position of the **Signature**).

## 3.3 Simple Printing

In the absence of **Layout** or **PrintLayout** instructions, the job will be produced via simple printing. In this case, the pages from each **Pages** array of each **Document** in the **JobTicketContents**::**Documents** array is printed in order. **NewSheet** and **Trailer** objects are honored when encountered in the sequence of pages, but **InsertPage** objects are ignored. Trapping instructions are ignored unless **JobtTicketContents**::**TrappingSourceSelector** is *Contents*. **ColorantControl**, **Rendering** and other printing controls are honored when encountered in the sequence of pages.

## 3.4 Front/Back Alignment

PJTF 1.1

Production two-sided output is accomplished by specifying both **Front** and **Back Surfaces** for **Sheets** in a **Layout** or **PrintLayout** (see section 3.2, "Layout and PrintLayout" on page 12).

The **SurfaceContentsBox** is provided to specify the area of each **Surface** into which all content (including printer's marks) will be imaged. The **PlacedObject**::**CTM** key is used to position page contents onto the **Surface** within the **SurfaceContentsBox**. When present, the **SurfaceContentsBox** defines the coordinate space that **PlacedObject**::**CTMs** transform page contents into; when absent, the **PlacedObject**::**CTMs** transform contents into the coordinate space of the **Surface** itself.

**Media**::**Dimensions** allows a range of values so that multiple media sizes might match the media requested for a job. The **Sheet**::**LockOrigins** key is provided so that an imposition process may position the contents of the **Front** and **Back Surfaces** (and therefore the **SurfaceContentsBox**) to have a common origin and edge. This allows proper alignment of the contents for any allowed **Media** size.

The coordinate space of the **Front Surface** is always set up with its origin at its lower left corner, and **PlaceObject**::**CTMs** for **Front Surfaces** always position page contents into the first quadrant.

When **Sheet**::**LockOrigins** is *true*, the coordinate space of the **Back Surface** is set up with its origin at the lower right corner of the **Surface**, so that the **PlacedObject**::**CTM** positions page contents into the second quadrant (negative x increasing to the left, y increasing upward). This ensures proper alignment of content when the media size varies. **Surface**::**SurfaceContentsBox** must be specified for **Back Surfaces** when **Sheet**::**LockOrigins** is *true*.

When **Sheet**::**LockOrigins** is *false*, the coordinate space of the **Back Surface** is set up with its origin at the lower left corner of the **SurfaceContentsBox**, so that the **PlacedObject**::**CTM** positions page contents into the first quadrant (x increasing to the right, y increasing upward). In this case, the imposition process must know the exact size of the media in order to ensure proper alignment of **Front** and **Back** content.

## 3.5 Trapping

PJTF 1.1

Trapping instructions can be stored in job tickets. Several objects are provided: **Trapping** (page 66), **TrappingDetails** (page 66), **ColorantDetails** (page 67), **DeviceColorant** (page 68),**TrappingParameters** (page 69), **ColorantZoneDetails** (page 75) and **TrapRegion** (page 75).

Trapping instructions may be specified in the contents hierarchy in **JobTicketContents** (page 27), **Document** (page 39) and **PageRange** (page 43) objects, and also in a **Layout** (page 77) or **PrintLayout** (page 78) hierarchy, using **Layout** or **PrintLayout** objects with **PlacedObject** (page 86) objects.

The **TrappingSourceSelector** (page 33) key in the **JobTicketContents** (page 27) object is used to determine which trapping information will be used. When **TrappingSourceSelector** is *none*, no trap networks are created.

The **Trapping** and **TrappingDetails** objects specify whether pages will be trapped, and what trapping method should be used to create trap networks for the pages.

The **ColorantDetails** and **DeviceColorant** objects provide information to the trapping engine about the named colorants which will be used to print the job. The named colorants identified in these objects must be consistent with the colorants requested in the **ColorantControl** object for each page. An inconsistency will cause trapping to fail.

Trapping is specified for regions of pages using **TrapRegion** (page 75) objects. These objects specify the geometry of the zone and the **TrappingParameters** (page 69) set to be used when trapping that region. **TrappingParameters** objects are identified by name to allow the user to associate specific parameter sets with particular output targets (e.g., a parameter set named ‘NewsPrint’).

The **Trapped** key in the **JTFile** object indicates whether a file has previously been trapped, whether or not such traps can be easily identified by Job Ticket Processors. When the value of this key is *true*, a Job Ticket Processor shall not trap the referenced file unless it can identify the existing traps and replace or update them.

Individual trapping Job Ticket Processors shall determine whether or not to treat the inability to comply with trapping requests for any file or file type as an error.

There is also a **Trapped** key in the **Info** dictionary of PDF files (version 1.3). In the case where the **JTFile** object refers to a PDF file in which the **Trapped** key is present, the values for the **Trapped** key in the job ticket and in the PDF **Info** dictionary are expected to be the same, and Job Ticket Processors shall rely on the job ticket **Trapped** key.

See Section 6.3, "In-RIP Trapping," in the *PostScript Language Reference, Third Edition*, for more information on trapping and trapping controls.

## 3.6 Named Colorant Aliasing

PJTF 1.1

Colorant names for the PDL content of a job may not be consistent between files included in the job, or even between pages within a single file. The Portable Job Ticket Format provides a mechanism to ensure that all named colorants for a job share a single consistent name space and that all instances of each named colorant utilize a consistent emulation.

**ColorantAlias** objects are used to translate colorant names which occur in PDL files into the job name space for named colorants. All colorant names in job ticket objects, such as **ColorantControl**::**ColorantParams** and **ColorantControl**::**ColorantOrder**, and the entries in **TrappingDetails**::**ColorantDetails** dictionaries should reflect this job-wide name space.

In addition to ensuring a consistent colorant name space via **ColorantAlias** objects, PJTF provides the **ColorSpaceSubstitute** object to ensure that all instances of each colorant are defined using the same color space, and in particular, the same emulation (tint transform).

Named colorant aliasing may be a two-step process.

In the first step, all instances of named colorants in **DeviceN** and **Separation** color spaces are examined to determine whether any of their colorant names appear as values in the **Aliases** array of **ColorantAlias** objects. When matches are found, the colorant name is replaced by the value of the **ReplacementColorantName** key in the **ColorantAlias** object.

In the second step, color spaces are examined to determine whether their colorant name(s) match the values of the **TargetColorantNames** key for any **ColorSpaceSubstitute** object. When matches are found, the use of the color space object is replaced by the use of the color space resource object referred to by the **ColorSpace** key in the **ColorSpaceSubstitute** object.

It is important to note that colorant aliasing is only applied to the contents of files referenced by **JTFile** objects.

Also note that colorant aliasing cannot be performed on process colorants which are implicitly requested (e.g., when the PDF **k** operator is used to set the current fill color to some combination of CMYK values.)

Finally, note that color space substitution may only be performed on **Separation** or **DeviceN** color spaces in PDF files.

## 3.7 Separations and Pre-separated Files

PJTF 1.1

The Portable Job Ticket Format allows for PDL content which has been pre-separated. The Job Ticket Manager will specify attributes of the file(s) containing the pre-separated PDL content. There are two different forms of pre-separated PDL content. The first has all pages of one colorant in a file, and multiple files are then required to fully specify all planes of any document page content via the **JTFile**::**FilesDictionary** key. The second groups all planes of each document page and, therefore, the order and number of planes per page must be provided via the **JTFile**::**PlaneOrder** key.

In the first case, each PDL “page” reflects only one colorant or plane of a document page. In the second case, the file may contain both pre-separated and composite PDL “pages”: the existence of composite page is indicated by the special entry *All* within the **PlaneOrder**::**Planes** array.

All references to pages in this specification are to a document page as a virtual composite page (all planes taken into account).

It is ambiguous whether a PDL page reference is to a single color plane or the virtual composite page, until the **JTFile** object is consulted. If either the **FilesDictionary** or **PlaneOrder** key is present, a PDL page reflects only a single color plane. If both keys are absent, the page is said to be composite.

## 3.8 Resources in Job Tickets

PJTF 1.1

Printing resources are typically either embedded in PDL files or installed on devices; their formats are provided by PDL specifications. In addition, the name space of names used to refer to resources is typically PDL-specific. However, job tickets must be able to refer to resources in a PDL-independent way.

The **ResourceAlias** object (page 25) is used to create names by which job ticket objects can refer to resources which exist in PDL files, or which are provided by devices.

Note that the intent of the **ResourceAlias** object is to provide a consistent way for job tickets to refer to resources across the entire job. This solves the problem where there are different resources of the same type with the same name which exist in different files. By using **ResourceAlias** objects, all references from the job ticket to the (named) resource will refer to the same resource.

In addition to the **ResourceAlias** object, the **Resources** key for the **JobTicket** object (page 23) is used to specify resources as PDF objects which are embedded within job tickets.

**ResourceAlias** objects may reference resources in one of four ways:

- When resources are provided within the job ticket as elements of the **JobTicket**::**Resources** array, **ResourceAlias**::**Location** is **This**, and **ResourceAlias**::**Source** is an index into the array.

- When resources are provided in external PDF or JTF files as elements of the **JobTicket**::**Resources** array within that file, **ResourceAlias**::**Location** is **File**, the file is identified by the **ResourceAlias**::**SourceFile** key and the specific resource is identified by the **ResourceAlias**::**Source** key, whose value is an index into the array.

- When resources are provided in external files of types other than PDF (or JTF), **ResourceAlias**::**Location** is **File**, and the resource is identified by the combination of the **ResourceAlias**::**SourceFile** key and the **ResourceAlias**::**SourceName** key. The Job Ticket Processor must be able to resolve the resource reference, given the type of the external file.

- When resources are provided by devices, **ResourceAlias**::**Location** is **Device**, and the resource is identified by name using the **ResourceAlias**::**SourceName** key. The device must be able to identify and access the named resource.

Note that for the last two cases, the **ResourceAlias**::**ResourceName** key is used to identify the resource in the file or on the device if the **ResourceAlias**::**SourceName** key is absent.

Currently there are only two PJTF keys which refer to resources: **ColorSpaceSubstitute**::**ColorSpace** (page 57) and **TrappingParameters**::**HalftoneName** (page 71).

## 3.9 Preflight Information

PJTF 1.1

Preflighting is the process of examining the components of a print job to ensure that the job will print successfully, and with the expected results. The PJTF supports preflighting jobs via eight objects, **Preflight** (page 91), **Inventory** (page 92), **Profile** (page 93), **Analysis** (page 94), **PreflightConstraint** (page 96), **PreflightResults** (page 97), **PreflightDetail** (page 98), **PreflightInstance** (page 99), and **PreflightInstanceDetail** (page 100).

Preflight checks may be performed on each PDL document identified by means of the **Documents** or **MarkDocuments** arrays of the **JobTicketContents** object.

Preflighting a file is generally a three-step process. First, the application inventories the file, identifying the significant characteristics of all the objects in the file. Next, the characteristics are tested against a set of criteria specified by the user. Finally, discrepancies are reported to the user.

Applications record the instructions for, and results of, preflight operations in Job Tickets, using hierarchies headed by three objects. The **Inventory** hierarchy may be used to record all the information gathered in the first step, although applications need not do this. The **Profile** hierarchy is used to record the criteria used to test the file in the second step. And an **Analysis** hierarchy is used to record results of the tests.

In all three hierarchies, the information is grouped into six (pre-defined) categories: **Colors**, **Document**, **Fonts**, **FileType**, **Images** and **Pages**, although applications may define other categories if needed. See “Extending Keys” on page 22 and Appendix A, “Extending Job Ticket Keys” on page 103 for more information about adding categories to the preflight hierarchies.

In a **Profile** hierarchy, each category is populated with **PreflightConstraint** objects. Each **PreflightConstraint** object specifies a test which the application will perform when analyzing the file.

In the **Inventory** and **Analysis** hierarchies, each category is populated with **PreflightResults** objects which record information about specific characteristics of the file. Such information is recorded in two ways:

- Information which is specific to one instance of some file object is recorded via **PreflightInstance** and **PreflightInstanceDetail** objects which occur in the **PreflightResults**::**PreflightInstances** array. For example, to record information about each font used in the file, the **PreflightResults**::**PreflightInstances** array contains one **PreflightInstance** object groups a set of **PreflightInstanceDetail** objects. Each **PreflightInstanceDetail** object records one specific characteristic of the font.

- Information which applies to the file as a whole is recorded via **PreflightDetail** objects which occur in the **PreflightResults**::**PreflightInstances** array. For example, to record all the page sizes used in the file, the **PreflightResults**::**PreflightDetails** array would contain several **PreflightDetail** objects, one for each page size used in the file.

An **Inventory** hierarchy may be used to record all information about the file. Preflight tools are not required to create an **Inventory** hierarchy as part of the preflight information they record. However, tools may find it useful to record this information as part of the job ticket, allowing them to avoid re-parsing the entire file in order to perform a new **Analysis**.

**Profile** hierarchies specify the constraints against which the file is tested. Each **Analysis** hierarchy reflects the results of evaluating the file characteristics, which may be recorded in an **Inventory** hierarchy, against a set of tests recorded in a **Profile** hierarchy.

**PreflightConstraint** objects record the specific details for the constraints specified in the **Profile** object. **PreflightDetail** and **PreflightInstanceDetail** objects record results returned in the **Analysis** objects, while **PreflightInstance** objects group **PreflightInstanceDetail** objects for instances of file objects. The details recorded are typically PDL-specific. See Appendix F, “Preflight Semantics for PDF” on 121, for an example of preflight constraints for one language.

Applications can define constraints within any of the defined constraint categories for any file type. In addition, applications may add to the set of defined constraints and constraint categories, defining both the new category(ies) and the constraint(s) within the category(ies).

Whether constraints are specified for predefined or new constraint categories, the eventual values for those constraints are always expressed as **PreflightConstraint** objects which are part of a **Preflight**::**Profile**. And the results are always either arrays of **PreflightDetail** objects or **PreflightInstance** objects which group **PreflightInstanceDetail** objects for **Analysis** results.

## 4 Header of a Job Ticket File

The header of a stand-alone job ticket file begins with a %JTF-1.2 comment. Such a file is referred to in this document as a Job Ticket Format (JTF) file. The revision number (1.2) corresponds to the version of the PDF language specification on which the job ticket file is based, and may change.

Note that the **Version** key in the **JobTicket** object indicates the version of the job ticket specification for the job ticket.

When a job ticket is included in a PDF file, the file header begins with a %PDF-1.2 comment (the revision number may change). In this case, the job ticket resides in the **JobTicket** object of the file's catalog.

In all cases, a single job ticket shall control the printing process. In some cases, a job ticket may contain a **JTFile** object which refers to a PDF file containing another job ticket. A Job Ticket Processor may examine objects within that secondary job ticket to locate controls which are not specified in the primary job ticket. However, controls in the secondary job ticket shall not override controls in the primary job ticket.

A Job Ticket Manager may use information detailed in a job ticket embedded in a PDF when building a JTF file it creates for Job Ticket Processors that follow.

## 5 Job Ticket Format

Job tickets are accommodated in PDF through the addition of the new key **JobTicket** to the **Catalog** object in the body of a PDF or JTF file.

*Table 1 JobTicket object within Catalog object*

| Key | Type | Semantics |
|---|---|---|
| **JT** (JobTicket) | dictionary | *(Required for JTF files. Optional for PDF files.)* A **JobTicket** object (page 23). |

The term *Optional* indicates an entry that is not always required. Some optional entries may be required, depending on the values of other keys.

## 5.1 Extending Keys

As with PDF 1.2, a Job Ticket Manager may add keys to any object identified under the **JobTicket** dictionary. Any extension must conform to the second-class registry rules (using registered developer-specific prefixes) as defined in Appendix F of the *Portable Document Format 1.2 Reference Manual*.

See Appendix A, “Extending Job Ticket Keys” on page 103 for an example of how to extend job ticket keys.

## 5.2 Default Values

Default values are generally not provided for optional keys in this specification. In some cases, a default value is provided in one of two ways:

- Either, a specific default value is provided for the key, or
- The definition of the key indicates that the value of some other key provides the default

When another key provides the default, the alternate may be a different key in the same object (where the optional key is omitted), or a key for a different object. In all cases where such alternates are used, they are explicitly stated in the specification (e.g., **BleedBox** defaults to **MediaBox**, and **PageRange**::**MediaBox** defaults to **Document**::**MediaBox**).

When the specification does not provide a default value for optional key, devices or Job Ticket Processors must provide a default value.

## 5.3 Inheritance and Overrides in Job Tickets

Job tickets are organized into hierarchical trees of objects. All objects are descended from the root **JobTicket** object (page 23).

Many objects defined by this specification may exist at more than one level in the tree. In these cases, the higher-level instances are over-ridden by lower-level instances. However, such overrides occur on a per-object basis. Lower-level instances must be complete and will override all keys in the higher level instance.

No mechanism is currently provided to allow lower-level instances to override higher-level instances on a per-key basis.

For example, a job ticket might include a **ColorantControl** object in its **JobTicketContents** object and in one of its **PageRange** objects. In that case, the

**PageRange**::**ColorantControl** object would override the **JobTicketContents**::**ColorantControl** object for that one **PageRange**, but the **JobTicketContents**::**ColorantControl** object would be used for all other **PageRanges** in job.

However if, continuing the example, the **PageRange**::**ColorantControl** object did not contain a **ProcessColorModel** key, the device's default value for **ProcessColorModel** would be used for that **PageRange**, not the **JobTicketContents**::**ColorantControl**::**ProcessColorModel** key.

## 5.4 JobTicket Objects

A **JobTicket** object describes how to create the general environment required to process a job. The objects in the **JobTicket** describe job-level settings to invoke when processing documents and other job-related objects referenced within the **JobTicket** object.

*Table 2 JobTicket attributes*

| Key | Type | Semantics | |
|---|---|---|---|
| **A** (Audit) | array | *(Required)* An array of one or more **Audit** objects (page 25). | |
| **Cn** (Contents) | array | *(Optional)* An array of exactly one **JobTicketContents** (page 27) object. | |
| **R** (Resources) | array | *(Optional)* An array of PDF objects. These objects may be PDF resources, or they may be resources defined in compliance with some other PDL. If not PDF resources, they must be PDF stream objects with the following keys in their stream dictionaries:<br>**FileType** This string identifies the PDL in which the resource is defined.<br>**ResourceType** This string identifies the resource type in the PDL identified by **FileType**. | PJTF 1.1 |
| **RA** (ResourceAliases) | array | *(Optional)* An array of **ResourceAlias** objects (page 25). | PJTF 1.1 |

*Table 2 JobTicket attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **V** (Version) | number | *(Required)* Lowest version of the job ticket specification with which this particular job ticket complies. **Version** implies that there are no objects or keys which appear in the job ticket which come from a higher version of the job ticket specification. |

## 5.5 Audit Objects

**Audit** objects keep track of changes in a job ticket. They are useful in multistep production work environments.

**Audit** objects can occur as elements in the **JobTicket::Audit** or **JTFile**::**Audit** arrays.

*Table 3 Audit attributes*

| Key | Type | Semantics |
|---|---|---|
| **Au** (Author) | dictionary | *(Optional)* An **Address** object (page 34) for the person responsible for this change of the job ticket. |
| **C** (Comment) | text | *(Optional)* A comment describing details of the action documented by this **Audit** object. PJTF 1.1 |
| **Dt** (Date) | date | *(Required)* Date on which the action was concluded. |
| **D** (Details) | dictionary | (*Optional*) Arbitrary key/value pairs describing details of the action documented by this **Audit** object |
| **Fi** (Files) | array | *(Optional)* An array of **JTFile** objects (page 47) which reference the physical files affected by the operation recorded by this **Audit** object. PJTF 1.1<br><br>The **JTFile** objects in this array shall also occur either in **Document**::**Files** array for some **Document** object which occurs in the **JobTicketContents**::**Documents** array or **JobTicketContents**::**MarkDocuments** array, or as the value for a **PageRange**::**JTFile** key. |

*Table 3 Audit attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **JTM** (JTManager) | string | *(Required)* Identifies the Job Ticket Manager that acted on the job ticket. The name of the process or application that created the file. |

## 5.6 ResourceAlias Objects

PJTF 1.1

**ResourceAlias** objects provide a mechanism for the job ticket to identify resources which will be used by the job. In some cases, the resources will be provided as separate files (whose only purpose is to provide those resources). In other cases, the resource will be provided within the job ticket in the **JobTicket**::**Resources** array (page 23). In still other cases, the resource will reside on the device. In all cases, the **ResourceAlias** object is used to identify that resource so that it may be referenced from job ticket objects.

Currently, only two types of **ResourceAliases** are referenced by job ticket objects. Aliases for **Halftone** resources are referenced by the **TrappingParameters**::**HalftoneName** key, and aliases for **ColorSpace** resources are referenced by the **ColorSpaceSubstitute**::**ColorSpace** key. Identification of any other resource type is not currently supported by this mechanism.

**ResourceAlias** objects can occur as elements in the **ResourceAlias** array of **JobTicket** objects.

*Table 4 ResourceAlias attributes*

| Key | Type | Semantics | |
|---|---|---|---|
| **L** (Location) | name | (*Required*) Defines where the resource is to be found. Must be one of the following: | |
| | | *Device* | The resource is known to the device. **SourceName** (or **ResourceName** if **SourceName** is absent) must exist to identify the resource. |
| | | *File* | The resource exists in an external file which is identified by **SourceFile**.<br><br>If the file is a PDF or JTF file, **Source** must be present to identify the element of the **JobTicket**::**Resources** array within the file.<br><br>If the file is of any other type, **SourceName** (or **ResourceName** if **SourceName** is absent) must exist to identify the resource in the file. |
| | | *This* | The resource exists in the **JobTicket**::**Resources** array of this job ticket, and **Source** is an index into the array. |
| **RN** (ResourceName) | name | *(Required)* The name used to reference the resource from job ticket objects.<br><br>If **SourceName** is absent, and **Location** is **Device** or **Location** is **File** and **Source** is absent, **ResourceName** identifies the resource on the device or in the file. | |
| **RT** (ResourceType) | name | *(Required)* The type of resource. Currently **ColorSpace** and **Halftone ResourceType**s are supported. | |

*Table 4 ResourceAlias attributes*

| Key | Type | Semantics |
|---|---|---|
| **S** (Source) | integer | *(Optional – Required if **Location** is This or if **Location** is File and the file identified by **SourceFile** is a PDF or JTF file.)* An index into the **JobTicket**::**Resources** array of the file identified by **SourceFile**, or in this job ticket if **Location** is **This**. The number must be a valid index into the **Resources** array, and must identify a resource of the type specified by **ResourceType**. |
| **SF** (SourceFile) | dictionary | *(Optional)* A **JTFile** object (page 47) identifying the file which holds the resource. **SourceFile** may be ignored if **Location** is **Device**. |
| **SN** (SourceName) | name | *(Optional)* The name of the resource in **SourceFile** (if different from **ResourceName**). This key is ignored if **Source** is present. |

## 5.7 JobTicketContents Objects

The keys in the **JobTicketContents** object describe job-level settings to invoke when processing documents and other job-related objects referenced within the **JobTicketContents** object. A job may include the **NewSheet** and/or **Trailer** when **PrintLayout** or simple printing is the printing method. Each copy of the job includes the optional **NewSheet** or **Trailer** (for **PrintLayout** or simple printing). (When using the **Layout**, the job content is always fully specified by its array of **Signatures**; thus **NewSheet** and **Trailer** will be ignored.) A job will always begin with the first **Signature**.

**JobTicketContents** objects occur as elements in the **JobTicket**::**Contents** array.

*Table 5 JobTicketContents attributes*

| Key | Type | Semantics |
|---|---|---|
| **Type** | name | *(Required)* Object type. Always **JobTicketContents**. |
| **A** (Accounting) | dictionary | *(Optional)* An **Accounting** object (page 38). |

*Table 5 JobTicketContents attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **Ad** (Administrator) | dictionary | *(Optional)* An **Address** object (page 34) identifying the person to whom errors and notifications involving this job are reported. On most systems, the information is echoed to this address as well as directed to the normal administrative process. |
| **Bl** (BleedBox) | rectangle | *(Optional)* A rectangle in page coordinate space units specifying a region of the page. The **BleedBox** encompasses all marks which are intended to be imaged on a final trimmed page or spread, including content which may extend outside the boundaries of the trimmed page or spread. The **BleedBox** represents the maximum extent of the final trimmed page output in a production environment. In such environments, a "bleed area" is desired, to accommodate physical limitations of cutting, folding and trimming equipment. The **BleedBox** shall not extend outside the **MediaBox**, and if it does, the effective **BleedBox** shall be the intersection of the **BleedBox** with the **MediaBox**.<br><br>When absent, the value of **MediaBox** is used for the **BleedBox**. |
| **Co** (ColorantControl) | dictionary | *(Optional)* A **ColorantControl** object (page 52). A **ColorantControl** object describes how to control output color rendering. |
| **Cm** (Comments) | text | *(Optional)* A human-readable note regarding the job. |
| **Dl** (Delivery) | array | *(Optional)* Indicates how many units are to be produced and where they are to be delivered. The value is an array of **Delivery** objects (page 35) in order of delivery priority. |
| **D** (Documents) | array | *(Optional)* An array of **Document** objects (page 39) in the order in which they are to be processed. |

PJTF 1.1

*Table 5 JobTicketContents attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **EM** (EndMessage) | text | *(Optional)* Human-readable message to be displayed at job end. |
| **F** (Finishing) | array | *(Optional)* An array of **Finishing** objects (page 36). Each **Finishing** operation is specified in a separate object, and the array determines the order of operations. |
| **FP** (FontPolicy) | dictionary | *(Optional)* A **FontPolicy** object (page 33) that describes how to handle missing fonts. |
| **IH** (IgnoreHalftone) | boolean | *(Optional)* If *true*, all operators that set halftones within any documents in this job are ignored. This includes the **setscreen**, **setcolorscreen**, and **sethalftone** operators in PostScript files, and halftone resources in extended graphics state resources in PDF documents. |
| **IPD** (IgnorePagedevice) | boolean | *(Optional)* This key controls the effect of calls to the PostScript **setpagedevice** operator from within source documents for the job. Many of the device controls specified through **setpagedevice** are available within the job ticket.<br><br>If *true*, calls to **setpagedevice** which specify device behavior which may be controlled by the job ticket are ignored. |
| **IP** (InsertPage) | dictionary | *(Optional)* An **InsertPage** object (page 61). A single instance of blank or alternate page content may be inserted to ensure that the job starts on the designated page position (odd or even). This object will specify the alternate page content explicitly. Applies only to **PrintLayout** printing. This object is applied after **NewSheet** is satisfied. |
| **IS** (InsertSheet) | dictionary | *Obsolete in version 1.1.*<br><br>*(Optional)* An **InsertSheet** object (page 62). Inserts a sheet after each copy of the job. |

PJTF 1.0

*Table 5 JobTicketContents attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **JN** (JobName) | text | *(Optional)* Human-readable job name. | |
| **L** (Layout) | dictionary | *(Optional)* **Layout** object (page 77) that describes how pages are to be laid out on sheets. If this key and **PrintLayout** are both absent, simple printing is used. | PJTF 1.1 |
| **MB** (MediaBox) | rectangle | *(Optional)* A rectangle in page coordinate space units specifying a region that contains the maximum imageable area of the page. This rectangle includes any extended area surrounding the finished page for bleed, printers marks, or other similar purpose. Content outside the **MediaBox** may be safely discarded without changing the meaning of the PDL file.<br><br>The value specified by this key overrides any similar information provided within the PDL file (such as a MediaBox key in a PDF file). | PJTF 1.1 |
| **MD** (MarkDocuments) | array | *(Optional)* An array of **Document** objects (page 39) whose pages contain marks which are not part of the content for the job. Examples include trim marks, color bars, and watermarks. | PJTF 1.1 |
| **MS** (MediaSource) | dictionary | *(Optional)* A **MediaSource** object (page 58). The medium described here is used when individual **Sheets** have not specified a media source. | |
| **MU** (MediaUsage) | dictionary | *(Optional)* A **MediaUsage** object (page 60). Indicates whether roll-fed media will be advanced and/or cut; the media usage described here is used for simple printing when no **Layout** is specified. | |

*Table 5 JobTicketContents attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **Ns** (NewSheet) | dictionary | *(Optional)* An **InsertSheet** object (page 62). Enables one to define an initial sheet for **PrintLayout** or simple printing. The **InsertSheet** object will specify whether the inserted sheet will be imaged or not. **FillContent** is ignored for this instance of **NewSheet**. If **NewSheet** is present, each copy of the job will include this insert. PJTF 1.1 |
| **PL** (PrintLayout) | dictionary | *(Optional)* **PrintLayout** object (page 78) that describes how pages are to be laid out on sheets. If the **Layout** key is present, this key is ignored unless the device cannot honor the **Layout** key. If this key and **Layout** are both absent, simple printing is used. |
| **R** (Rendering) | dictionary | *(Optional)* A **Rendering** object (page 64). The rendering described here is used when individual documents and pages do not provide a rendering description. |
| **Sc** (Scheduling) | dictionary | *(Optional)* A **Scheduling** object (page 37) indicating criteria that will determine when a job can be processed or printed. |
| **SM** (StartMessage) | text | *(Optional)* Message to be displayed to operator at job start. |
| **S** (Submitter) | dictionary | *(Optional)* An **Address** object (page 34) identifying the person who submitted this job. |
| **T** (Trapping) | dictionary | *(Optional)* A **Trapping** object (page 66). The trapping settings described here are used when individual documents and pages do not provide trapping settings. PJTF 1.1 |

*Table 5 JobTicketContents attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **TB** (TrimBox) | rectangle | *(Optional)* A rectangle in page coordinate space units specifying a region which is the intended finished (trimmed) size of the page. For example, the dimensions of an A4 sheet of paper. The **TrimBox** shall not extent outside the **MediaBox** and if it does, the effective **TrimBox** shall be the intersection of the **TrimBox** with the **MediaBox**. In some cases, the **MediaBox** may be larger than the **TrimBox** and include printing instructions, color bars, cut marks, or other printers marks.<br><br>When absent, the value of **MediaBox** is used for the **TrimBox**. | PJTF 1.1 |
| **TI** (Trailer) | dictionary | *(Optional)* An **InsertSheet** object (page 62). Specifies if a sheet will be inserted at the end of the job (when using the **PrintLayout** or simple printing method). This object will specify how to complete the current **Sheet** being imaged. The inserted sheet may be imaged with page content specified in this object. If **Trailer** is present, each copy of the job will include this insert. | PJTF 1.1 |
| **TD** (TrappingDescription) | text | *(Optional)* A descriptive name to apply to the trap network which will be produced by a trapping application as a result of the **TrapRegion** objects for this job. | PJTF 1.1 |
| **TP** (TrappingParameters) | dictionary | *(Optional)* A dictionary in which each key is the name of a **TrappingParameters** set and each value is a **TrappingParameters** object (page 69). These objects specify the sets of trapping parameters which will be used to create trap networks for pages in the job. | PJTF 1.1 |

*Table 5 JobTicketContents attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **TR** (TrapRegions) | array | *(Optional)* An array of **TrapRegion** objects (page 75). These objects specify the trapping regions and trapping parameters which will be used to create trap networks for pages in the **Documents** in this job.<br><br>Note that trap networks are created for a page only when there is at least one **TrapRegion** object for that page. PJTF 1.1 |
| **TSS** (TrappingSourceSelector) | name | *(Optional)* Controls which trapping information should be honored. Recognized values are:<br><br>*Contents* — Only **TrappingParameters** and **TrapRegion** objects in **JobTicketContents**, **Document** and **PageRange** objects are used. All others are ignored.<br>*Layout* — Only **Trapping** and **TrappingParameters** objects in **Layout** objects and **TrapRegion** objects in **PlacedObject** objects are used. All others are ignored.<br>*PrintLayout* — Only **Trapping** and **TrappingParameters** objects in **PrintLayout** objects and **TrapRegion** objects in **PlacedObject** objects are used. All others are ignored.<br>*None* — All trapping information is ignored and no trap networks are created.<br><br>However, the effect of this key on files whose **JTFile**::**Trapped** key has the value of *true* will depend on the Job Ticket Processor that processes the trapping objects. PJTF 1.1 |

## 5.8 FontPolicy Objects

During execution, the job may explicitly specify a font that the Job Ticket Processor cannot locate. In these cases, **FontPolicy** objects specify the desired fallback behavior.

**FontPolicy** objects can occur as the value for the **JobTicketContents**::**FontPolicy** key.

*Table 6 FontPolicy attributes*

| Key | Type | Semantics |
|---|---|---|
| **UDF** (UseDefaultFont) | boolean | *(Optional)* States whether or not the device shall resort to a default font if a font cannot be found. This is the normal behavior of the PostScript interpreter, which defaults to Courier when a font cannot be found. |
| **UFE** (UseFontEmulation) | boolean | *(Optional)* States whether or not the device shall emulate a required font if the required font cannot be found. |

If both **UseDefaultFont** and **UseFontEmulation** are *true*, a device first attempts to create an appropriate emulated font when the required font cannot be found. Failing that, the default font is used. If both booleans are *false*, the job shall fail when a font is missing.

## 5.9 Address Objects

**Address** objects provide information for job delivery, accounting, auditing, and administrative purposes.

**Address** objects can occur as the value for any of the following keys:

**JobTicketContents::Administrator**
**JobTicketContents::Submitter**
**JTFile::Authors**
**Audit::Author**
**Delivery::Address**
**Accounting::Addressee**

*Table 7 Address attributes*

| Key | Type | Semantics |
|---|---|---|
| **A** (Address) | text | *(Optional)* A multi-line postal address. Information such as Mail Stop, Apartment Number, etc. are included in this text. |
| **Ct** (City) | text | *(Optional)* City portion of address. |

*Table 7 Address attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **Co** (Company) | text | *(Optional)* The business or organization of addressee. |
| **Cn** (Country) | text | *(Optional)* Country of address. This value conforms to the ISO 3166 standard in which countries are represented as 2-character codes. |
| **E** (Email) | text | *(Optional)* Electronic mail address. |
| **F** (Fax) | phone | *(Optional)* A phone number defining addressee's fax telephone number. |
| **M** (Message) | text | *(Optional)* Special instructions that appear on the package or cover sheet, or are prominently displayed in e-mail. |
| **Na** (Name) | text | *(Optional)* The addressee's name. |
| **P** (Pager) | phone | *(Optional)* A phone number defining addressee's pager telephone number. |
| **Ph** (Phone) | phone | *(Optional)* A phone number defining addressee's telephone number. |
| **PC** (PostalCode) | text | *(Optional)* Zip code or postal code of address. |
| **S** (State) | text | *(Optional)* State or province of address. |
| **T** (Title) | text | *(Optional)* The addressee's title. |

## 5.10 Delivery Objects

A **Delivery** object describes how output from the job is to be delivered.

**Delivery** objects can occur as the value for the **JobTicketContents**::**Delivery** key.

*Table 8 Delivery attributes*

| Key | Type | Semantics |
|---|---|---|
| **A** (Address) | dictionary | *(Optional)* An **Address** object (page 34) identifying where the output is to be delivered. The fields present in the **Address** object depend on the value of **Method**. |
| **Cm** (Comment) | text | *(Optional)* Special instructions for delivery. |
| **Cp** (Copies) | integer | *(Optional)* A non-negative integer specifying how many copies of the entire job are to be delivered to this location. |
| **M** (Method) | string | *(Optional)* Any string.<br><br>May be one of the following names **PickUp**, **SecurePickUp**, **InterofficeMail**, **PostalMail**, **PrintedMatterMail**, **BulkMail**, **ExpressMail**, **BudgetExpressMail**, **OutputBin**.<br><br>The value of this key may affect which keys are needed for the **Address** object, and may imply semantics for the values of the **Address** keys. |

## 5.11 Finishing Objects

**Finishing** objects describe the post-imaging treatments, such as stapling or punching, that are applied to the job output. When multiple **Finishing** objects are specified for a job, different Job Ticket Processors may perform each operation.

Additional key/value pairs are included as needed to fully specify the particular finishing operation. For example, a punch operation might contain the following two key-value pairs: `<</Locations [36 [72 396 720]] /Diameter 18>>`.

**Finishing** objects can occur as elements in the **Finishing** array for **JobTicketContents**, **PrintLayout, Layout, Signature**, **Sheet** or **Tile** objects.

*Table 9 Finishing attributes*

| Key | Type | Semantics |
|---|---|---|
| **D** (Details) | dictionary | *(Optional)* Provides details for the finishing operation. PJTF 1.1 |

Table 9 *Finishing attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **IS** (InsertSheet) | dictionary | Obsolete in version 1.1. *(Optional)* An **InsertSheet** object (page 62). Inserts a sheet after each copy of the job. PJTF 1.0 |
| **O** (Operation) | string | *(Required)* Identifies the category of **Finishing** operation. Some values for this key may correspond to page device keys (e.g., **Bind**, **Collate**, **Fold**, **Jog**, **Laminate**, **Staple**) while others may not (e.g., **Cover**, **DieCut**, **Perforate**, **Punch**, **ShrinkWrap**, **Stitch**, **Trim**.) |
| **S** (SlipSheet) | dictionary | *(Optional)* A **SlipSheet** object (page 63). Specifies that a blank sheet of media will be inserted after the finishing operation is complete. Once inserted, this sheet is subject to any subsequent **Finishing** operations. PJTF 1.1 |

## 5.12 Scheduling Objects

**Scheduling** objects describe time periods during which the job is expected to execute. It can also contain instructions about starting the job, completing the job, or deleting job files.

**Scheduling** objects can occur as the value for the **Scheduling** key for the **JobTicketContents** object.

Table 10 *Scheduling attributes*

| Key | Type | Semantics |
|---|---|---|
| **D** (Deadline) | date | *(Optional)* A date and time by which the user wants this job completed. |
| **Di** (Discard) | date | *(Optional)* A date and time after which the job can be discarded regardless of whether it has completed. |
| **EW** (EndWait) | boolean | *(Optional)* If *true*, operator intervention required before *next* job can begin. |
| **H** (Hold) | boolean | *(Optional)* If *true*, job cannot be scheduled for printing. |

*Table 10 Scheduling attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **OM** (OperatorMessage) | text | *(Optional)* Human-readable text to be shown to the operator when the job begins processing. This message can be used to solicit an operator action, but lack of operator action should not interfere with job processing. |
| **Pw** (Password) | string | *(Optional)* Before the job is processed, the operator is prompted for a password and must enter one that matches this string.<br><br>This string is stored as clear text in the document, unless the document itself is PDF encrypted. |
| **PA** (PrintAfter) | date | *(Optional)* A date and time after which to print the job. |
| **Pr** (Priority) | integer | *(Optional)* A number from 1 to 100, inclusive, assigned by the user to indicate the urgency for printing this job. A higher number indicates a higher priority. |
| **R** (Retain) | integer | *(Optional)* The length of time (in minutes) that a job will be retained after it has completed. A job is the job ticket and all files referenced by **JTFile** objects.<br><br>After this time, the job ticket itself may be deleted, and other referenced files may be deleted according to the **FileRetention** entry associated with each **JTFile** object (page 47). |
| **SW** (StartWait) | boolean | *(Optional)* If *true*, job requires operator intervention before it can be processed. |

## 5.13 Accounting Objects

**Accounting** objects provide the basic information required to manage the job.

**Accounting** objects occur as the value for the **Accounting** key for the **JobTicketContents** object.

*Table 11 Accounting attributes*

| Key | Type | Semantics |
|---|---|---|
| **A** (Addressee) | dictionary | *(Optional)* An **Address** object (page 34) identifying the person to whom to direct this job's billing. |
| **ID** | text | *(Optional)* A job identifier to be used for accounting purposes. |
| **S** (Submitter) | dictionary | *(Optional)* An **Address** object (page 34) identifying the person who submitted the job. |

## 5.14 Document Objects

**Document** objects describe how to create the environment for objects at the document level. This environment can be thought of as a set of exceptions to the job-level environment. A **Document** may include a **NewSheet** and/or **Trailer** specified here or specified within a **PageRange** of the **Pages** array when referenced from a **PrintLayout**::**Signature**.

**Document** objects occur as the elements in the **Documents** array for the **JobTicketContents** object.

*Table 12 Document attributes*

| Key | Type | Semantics |
|---|---|---|
| **Bl** (BleedBox) | rectangle | *(Optional)* A rectangle in page coordinate space units specifying a region of the page. The **BleedBox** encompasses all marks which are intended to be imaged on a final trimmed page or spread, including content which may extend outside the boundaries of the trimmed page or spread. The **BleedBox** represents the maximum extent of the final trimmed page output in a production environment. In such environments, a "bleed area" is desired, to accommodate physical limitations of cutting, folding and trimming equipment. The **BleedBox** shall not extend outside the **MediaBox**, and if it does, the effective **BleedBox** shall be the intersection of the **BleedBox** with the **MediaBox**.<br><br>When absent, the value of **MediaBox** is used for the **BleedBox**. |
| **Co** (ColorantControl) | dictionary | *(Optional)* A **ColorantControl** object (page 52). A **ColorantControl** object describes how to control output color rendering. |
| **Cm** (Comment) | text | *(Optional)* Human-readable notes regarding the document. |
| **Cp** (Copies) | integer | *(Optional)* A non-negative integer specifying how many copies of this document are produced for each copy of the job.<br><br>This value is multiplied by the value of **Copies** for the **Delivery** object to determine the total number **Documents** included in the delivery.<br><br>This value is ignored when the job is printed using a **Layout** – it applies only to simple printing, or when printing a **PrintLayout**. |
| **Fi** (Files) | array | *(Required if any **PageRange** object uses the index form for its **JTFile** key)* An array of **JTFile** objects (page 47). If both **Pages** and **Files** arrays exist, the **Pages** array will be the source of all page contents. |

PJTF 1.1

*Table 12 Document attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **IP** (InsertPage) | dictionary | *(Optional)* An **InsertPage** object (page 61). A single instance of blank or alternate page content may be inserted to ensure that the **Document** starts on the designated page position (odd or even). This object will specify the alternate page content explicitly. This object is applied after **NewSheet** is satisfied. Applies only to **PrintLayout** printing. |
| **IS** (InsertSheet) | dictionary | Obsolete in version 1.1. PJTF 1.0<br><br>*(Optional)* An **InsertSheet** object (page 62). Inserts a sheet after each copy of the job. |
| **MB** (MediaBox) | rectangle | *(Optional)* A rectangle in page coordinate space units specifying a region that contains the maximum imageable area of the page. This rectangle includes any extended area surrounding the finished page for bleed, printers marks, or other similar purpose. Content outside the **MediaBox** may be safely discarded without changing the meaning of the PDL file.<br><br>The value specified by this key overrides any similar information provided within the PDL file (such as a MediaBox key in a PDF file). |
| **Na** (Name) | text | *(Optional)* The name of the document assigned by the user. |
| **Ns** (NewSheet) | dictionary | *(Optional)* An **InsertSheet** object (page 62). Specifies a completion of the current sheet, if one is being imaged, in order to have this set of page content begin with a new sheet of media. This object may also specify a sheet to be inserted as the first sheet of the document, and if so, whether the inserted sheet will be imaged or blank. If **NewSheet** is present, each copy of this **Document** will include this insert. Applies only to **PrintLayout** or simple printing. PJTF 1.1 |

*Table 12 Document attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **P** (Pages) | array | *(Optional)* An array of **PageRange** objects (page 43). Determines a sequence (ordered set) of pages, which, in the absence of **Layout** information (page 77), is also the printing order.<br><br>The **Pages** array can be omitted if the **JobTicket** object (page 23) is adequate to describe the document settings, and the **Files** array adequately describes the page order. This is taken to mean that all pages in each of the **JTFile** objects referenced in the **Files** array will print in ascending (file and page) order. | |
| **R** (Rendering) | dictionary | *(Optional)* A **Rendering** object (page 64). The rendering described here is used when individual pages do not provide a rendering description. | |
| **T** (Trapping) | dictionary | *(Optional)* A **Trapping** object (page 66). The trapping settings described here are used when individual pages have not provided trapping settings. | PJTF 1.1 |
| **TB** (TrimBox) | rectangle | *(Optional)* A rectangle in page coordinate space units specifying a region which is the intended finished (trimmed) size of the page. For example, the dimensions of an A4 sheet of paper. The **TrimBox** shall not extent outside the **MediaBox** and if it does, the effective **TrimBox** shall be the intersection of the **TrimBox** with the **MediaBox**. In some cases, the **MediaBox** may be larger than the **TrimBox** and include printing instructions, color bars, cut marks, or other printers marks.<br><br>When absent, the value of **MediaBox** is used for the **TrimBox**. | PJTF 1.1 |
| **Tl** (Trailer) | dictionary | *(Optional)* An **InsertSheet** object (page 62). Specifies if a sheet will be inserted at the end of this **Document**. This object will specify how to complete the current **Sheet** being imaged. The inserted sheet may be blank or imaged with page content specified in this object. If **Trailer** is present, each copy of this **Document** will include this insert. Applies only to **PrintLayout** or simple printing. | PJTF 1.1 |

*Table 12 Document attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **TD** (TrappingDescription) | text | *(Optional)* A descriptive name to apply to the trap network which will be produced by a trapping application as a result of the **TrapRegion** objects for this **Document**. | PJTF 1.1 |
| **TP** (TrappingParameters) | dictionary | *(Optional)* Each key is the name of a **TrappingParameters** set and each value is a **TrappingParameters** object (page 69). These objects specify the sets of trapping parameters which will be used to create trap networks for pages in this **Document**.<br><br>If absent, **TrappingParameters** objects specified by the **JobTicketContents**::**TrappingParameters** key are used. | PJTF 1.1 |
| **TR** (TrapRegions) | array | *(Optional)* An array of **TrapRegion** objects (page 75). These objects specify the trapping regions and trapping parameters which will be used to create trap networks for pages in this **Document**.<br><br>Note that trap networks are created for a page only when there is at least one **TrapRegion** object for that page. | PJTF 1.1 |

## 5.15 PageRange Objects

A **PageRange** object specifies operations that can be applied to a set of pages of a document, as indicated by the corresponding **JTFile** object. A **PageRange** may include a **NewSheet** and/or **Trailer** specified here when referenced from a **PrintLayout**::**Signature**.

**PageRange** objects occur as the elements in the **Pages** array for **Document** objects, or as values for the **PageRange** key for **InsertSheet** or **InsertPage** objects.

*Table 13 PageRange attributes*

| Key | Type | Semantics |
|---|---|---|
| **Bl** (BleedBox) | rectangle | PJTF 1.1 *(Optional)* A rectangle in page coordinate space units specifying a region of the page. The **BleedBox** encompasses all marks which are intended to be imaged on a final trimmed page or spread, including content which may extend outside the boundaries of the trimmed page or spread. The **BleedBox** represents the maximum extent of the final trimmed page output in a production environment. In such environments, a "bleed area" is desired, to accommodate physical limitations of cutting, folding and trimming equipment. The **BleedBox** shall not extend outside the **MediaBox**, and if it does, the effective **BleedBox** shall be the intersection of the **BleedBox** with the **MediaBox**.<br><br>When absent, the value of **MediaBox** is used for the **BleedBox**. |
| **Co** (ColorantControl) | dictionary | *(Optional)* A **ColorantControl** object (page 52). A **ColorantControl** object describes how to control output color rendering. |
| **Cp** (Copies) | integer | *(Optional)* A non-negative integer specifying how many copies of this **PageRange** are produced for each copy of the **Document** object (page 39).<br><br>This value multiplied by the value of **Copies** for the **Document** object this determines the total number of times the **PageRange** appears in the job.<br><br>This value is ignored when the job is printed using a **Layout**; it applies only to simple printing, or when printing a **PrintLayout**. |
| **IP** (InsertPage) | dictionary | *(Optional)* An **InsertPage** object (page 61). A single instance of blank or alternate page content may be inserted to ensure that the **PageRange** starts on the designated page position (odd or even). This object will specify the alternate page content explicitly. This object is applied after **NewSheet** is satisfied. Applies only to **PrintLayout** printing. |

*Table 13 PageRange attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **IS** (InsertSheet) | dictionary | *Obsolete in version 1.1.*<br><br>*(Optional)* An **InsertSheet** object (page 62). Inserts a sheet after each copy of the job. | PJTF 1.0 |
| **JTF** (JTFile) | integer or dictionary | *(Required)* When an integer, **JTFile** is an index into the **Files** array of the **Document** object parent to this **PageRange** object. Index values begin with zero (first element in **Files** array).<br><br>When a dictionary, **JTFile** is a **JTFile** object (page 47). | |
| **MB** (MediaBox) | rectangle | *(Optional)* A rectangle in page coordinate space units specifying a region that contains the maximum imageable area of the page. This rectangle includes any extended area surrounding the finished page for bleed, printers marks, or other similar purpose. Content outside the **MediaBox** may be safely discarded without changing the meaning of the PDL file.<br><br>The value specified by this key overrides any similar information provided within the PDL file (such as a **MediaBox** key in a PDF file).<br><br>If absent, the value of **Document**::**MediaBox** is used. | |
| **Ns** (NewSheet) | dictionary | *(Optional)* An **InsertSheet** object (page 62). Specifies a completion of the current sheet, if one is being imaged, in order to have this set of page content begin with a new sheet of media. This object may also specify a sheet to be inserted as the first sheet of the **PageRange**, and if so, whether the inserted sheet will be imaged or blank. If **NewSheet** is present, each copy of this **PageRange** will include this insert. Applies only to **PrintLayout** or simple printing. | PJTF 1.1 |
| **R** (Rendering) | dictionary | *(Optional)* A **Rendering** object (page 64). Settings in this dictionary take precedence over settings in the **Document** object (page 39) or **JobTicketContents** object's (page 27) **Rendering** dictionary. | |

*Table 13 PageRange attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **T** (Trapping) | dictionary | *(Optional)* A **Trapping** object (page 66). Settings in this dictionary take precedence over settings in the **Document** object's (page 39) or **JobTicketContents** object's (page 27) **Trapping** dictionary. | PJTF 1.1 |
| **TB** (TrimBox) | rectangle | *(Optional)* A rectangle in page coordinate space units specifying a region which is the intended finished (trimmed) size of the page. For example, the dimensions of an A4 sheet of paper. The **TrimBox** shall not extent outside the **MediaBox** and if it does, the effective **TrimBox** shall be the intersection of the **TrimBox** with the **MediaBox**. In some cases, the **MediaBox** may be larger than the **TrimBox** and include printing instructions, color bars, cut marks, or other printers marks.<br><br>When absent, the value of **MediaBox** is used for the **TrimBox**. | PJTF 1.1 |
| **TI** (Trailer) | dictionary | *(Optional)* An **InsertSheet** object (page 62). Specifies if a sheet will be inserted at the end of this **PageRange**. This object will specify how to complete the current **Sheet** being imaged. The inserted sheet may be blank or imaged with page content specified in this object. If **Trailer** is present, each copy of this **PageRange** will include this insert. Applies only to **PrintLayout** or simple printing. | PJTF 1.1 |
| **TD** (TrappingDescription) | text | *(Optional)* A descriptive name to apply to the trap network which will be produced by a trapping application as a result of the **TrapRegion** objects for this **PageRange**. | PJTF 1.1 |
| **TP** (TrappingParameters) | dictionary | *(Optional)* A dictionary in which each key is the name of a **TrappingParameter** set and each value is a **TrappingParameters** object (page 69). These objects specify the sets of trapping parameters which will be used to create trap networks for pages in this **PageRange**.<br><br>If absent, **TrappingParameters** objects specified by the **Document**::**TrappingParameters** key are used. | PJTF 1.1 |

*Table 13 PageRange attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **TR** (TrapRegions) | array | *(Optional)* An array of **TrapRegion** objects (page 75). These objects specify the trapping regions and trapping parameters which will be used to create trap networks for pages in this **PageRange**. <br><br> Note that trap networks are created for a page only when there is at least one **TrapRegion** object for that page. PJTF 1.1 |
| **W** (Which) | array | *(Optional)* An array of two integers [*N M*] where *N* and *M* are page numbers in the specified file and M > N. If omitted, all pages of the **JTFile** object (page 47) are used. <br><br> The first page in a file is always page 0. To specify a range of one page, use *N*=*M*. However, the value of –1 for M is treated as a special case (see below). <br><br> The following special value of *M* is recognized: <br><br> –1 Indicates the last page in a file. This is useful for printing ranges of pages when the number of pages in a file is unknown. For example, /Which [3 –1] instructs the Job Ticket Processor to print pages 4 through the end of the file. <br><br> An error occurs if this array specifies pages which fall outside the range of total pages in the file. <br><br> Note that the case of M < N, which was allowed in PJTF 1.0 to indicated printing in inverse order, is eliminated for PJTF 1.1 PJTF 1.0 |

## 5.16 JTFile Objects

A **JTFile** object provides attributes of a **JTFile** used by the job.

**JTFile** objects can occur as the value for the **JTFile** key in **PageRange** objects or as elements in the **Files** array for **Document** objects.

*Table 14 JTFile attributes*

| Key | Type | Semantics |
|---|---|---|
| **A** (Audit) | array | *(Optional)* Array of **Audit** objects. Each **Audit** object records some operation which affected the physical files referenced by this object's **File** or **FilesDictionary** key. The **Audit** objects in this array shall also occur in the **JobTicket::Audit** array. PJTF 1.1 |
| **Au** (Authors) | array | *(Optional)* Array of **Address** objects identifying the person(s) who created the file. |
| **Cm** Comment) | text | *(Optional)* Human-readable notes regarding the file. |
| **CP** (CreatingProcess) | string | *(Optional)* The name of the process or application that created the file. |
| **DP** (DecodeParams) | array | *(Optional)* An array of dictionaries. Parameters used by the decoding filters specified with the **Filters** key (page 50). The number and types of items in the array must match the items in the **Filters** array. If a filter has no parameters, enter an empty dictionary. |
| **Fi** (File) | filespec or name | *(Optional – Required if FilesDictionary is absent, disallowed if FilesDictionary is present)* Indicates a PDF file specification object (see Section 6.6.4 of the *Portable Document Format 1.2 Reference Manual*) or a name. The following name values are supported: *This* — The document is contained in the same PDF as this job ticket. *Follows* — The document will follow the job ticket when the two are streamed separately to a device. |
|  | Note: | *Only one* **JTFile** *object in a job ticket can use Follows.* |

*Table 14 JTFile attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **FD** (FilesDictionary) | dictionary | PJTF 1.1 *(Optional – Required if* ***File*** *is absent, disallowed if* ***File*** *is present)* Keys are Colorant names and values are file specification objects (see Section 6.6.4 of the *Portable Document Format 1.2 Reference Manual*).<br><br>Note that the colorant names in this dictionary are subject to colorant aliasing as specified in the **JobTicketContents** object (page 27) for the job.<br><br>Presence of this key indicates that the **JTFile** has been pre-separated, and that information for each colorant plane is stored in a separate file. Omission of this key indicates that the file is not pre-separated, unless the **JTFile**::**PlaneOrder** key is present.<br><br>Each file referred to in this dictionary must have the same page count.<br><br>The sequence of virtual pages defined by the **FilesDictionary** is constructed from the parallel sequences of separation pages. In this construction each virtual page is constructed by composing the corresponding separation pages, one from each file specified in the **FilesDictionary**. Here “corresponding” means the same position in the sequence of (separation/virtual) pages.<br><br>For example, the fourth virtual page will be constructed by composing the fourth page out of each of the files referenced in the **FilesDictionary**. |
| **FR** (FileRetention) | name | *(Optional)* Determines whether or not this file, or set of files for pre-separated files, will be retained beyond the **Scheduling** object’s **Retain** period. Allowed values are:<br><br>*UntilSuccess* — Retain this file until the job successfully completes and the **Scheduling** object’s (page 37) **Retain** period expires.<br><br>*Never* — Never retain this file after the **Scheduling** object’s **Retain** period expires, even if the job failed.<br><br>*Always* — Retain this file, regardless of when the job completes. |

*Table 14 JTFile attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **FT** (FileType) | string | *(Optional)* Identifies the type of data in **File**. Certain Job Ticket Processors may fail if **FileType** is not specified.<br><br>**FileType** must be a MIME file type as recorded by the Internet Assigned Numbers Authority (IANA). IANA has procedures for registering new file types if needed. A list of MIME types for some common FileTypes is included in this document as Appendix E.<br><br>The value of this key shall be honored, even if the device recognizes the content as a different type. For example, a PostScript file could have a text file type, resulting in the device processing the file as a text file, and not as PostScript.<br><br>Only one **FileType** is supported for each **JTFile** object; therefore, all files referenced in the **FilesDictionary** must be of the same type. |
| **FI** (Filters) | name or array | *(Optional)* A name or an array of names. Filter(s) to be applied in processing the file. Specify multiple filters in the order they should be applied to decode the data. Indicates the sequence of file conversions that must be performed to put the file into the format given by **FileType**.<br><br>Allowed values are the standard PDF filters, **ASCIIHexDecode**, **ASCII85Decode**, **LZWDecode**, **RunLengthDecode**, **CCITTFaxDecode**, **DCTDecode**, and **FlateDecode**, plus **SIT** (Stuffit), **CPT** (Compact Pro), **DD** (Disk Doubler), **ZIP**, **GZIP**, **BinHex**, **UUEncode**, and **MacBinary**.<br><br>The job shall fail if filters are specified which are unsupported or unrecognized by the Job Ticket Processor.<br><br>Only one set of **Filters** is supported for each **JTFile** object; therefore, all files referenced in the **FilesDictionary** must be used the same set of **Filters** for decoding purposes. |
| **Pf** (Preflight) | dictionary | *(Optional)* A **Preflight** object (page 91).This object specifies the constraints used to check the file and the results of preflight operations. It may also provide a complete inventory of the characteristics of the file. PJTF 1.1 |

*Table 14 JTFile attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **Pw** (Password) | string | *(Optional)* Specifies a password to use when decrypting an encrypted file. Decryption is required before a Job Ticket Processor can process the content of an encrypted file. Note that the encryption method is not specified or controlled by the job ticket; the Job Ticket Processor is expected to know what encryption method is in use, and how to apply the password. |
| **PO** (PlaneOrder) | array | *(Optional – disallowed if **FilesDictionary** is present)* An array of **PlaneOrder** objects (page 57) which specifies the order of colorant planes which occur in the file referenced by the **File** key. Presence of this key indicates that the file referenced is pre-separated. Omission of this key indicates that the file is not pre-separated, unless the **JTFile**::**FilesDictionary** key is present.<br><br>In order to identify all colorant planes for a specific virtual page, the Job Ticket Processor must examine the objects in this array in sequence.<br><br>Note that the colorant names in this dictionary are subject to colorant aliasing as specified in the **JobTicketContents** object (page 27) for the job. PJTF 1.1 |
| **RD** (RevisionDate) | date | *(Optional)* The date and time the file was last changed. |

*Table 14 JTFile attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **TR** (Trapped) | name | (*Optional*) Indicates whether the file has been trapped. Allowed values are: *True* The file has been trapped; *False* The file has not been trapped; *Unknown* It is not known whether the file is trapped. When this key is *True*, a Job Ticket Processor shall not trap the referenced file unless the processor can identify the existing traps and remove or update them. When the **FileType** is application/pdf, this key is defined in accordance with the PDF 1.3 language specification, and it is an error if the value of this key is inconsistent with the value of the **Trapped** key in the file. |

PJTF 1.1

## 5.17 ColorantControl Objects

**ColorantControl** objects identify how the colors in the job are to be rendered.

Keys in this object define whether separations will be produced and what colorant planes are marked.

**ColorantControl** objects may occur in content objects (**JobTicketContents, Document, PageRange**) or in layout objects (**PrintLayout**, **Layout**, **Signature**, **Sheet**, **Surface**). In the case where there are **ColorantControl** objects in both content and layout objects, the settings in the **ColorantControl** objects from the layout object hierarchy override those in the content objects.

For more information on the use of color in Extreme™ and PostScript 3™, see Technical Note #5606, "Printer Systems-Based Separations".

Also, see Section 6.2.5, "Color Support," in the *PostScript Language Reference, Third Edition.*

**ColorantControl** objects can occur as the value for the **ColorantControl** key for **JobTicketContents, Document, PageRange**, **PrintLayout**, **Layout**, **Signature**, **Sheet** or **Surface** objects.

*Table 15 ColorantControl attributes*

| Key | Type | Semantics |
|---|---|---|
| **CA** (ColorantAliases) | array | *(Optional)* An array of **ColorantAlias** objects (page 56). PJTF 1.1 |
| **CO** (ColorantOrder) | array | *(Optional)* Array of names that identifies which colorant planes shall be rendered, and in what order they shall be output when separations are printed.<br><br>When **Separations** is *true* and **ColorantOrder** is not provided, the printing system assumes the set of colorants is the union of the colorants implied by the color model specified in **ProcessColorModel** and the colorants listed in **ColorantParams**; the order being the colorants defined by the **ProcessColorModel** followed by those listed in **ColorantParams**.<br><br>When **Separations** is *false*, **ColorantOrder** specifies the colorants which will be marked; other colorants are unmarked. Composite devices apply colorants in a device specified order that cannot be controlled by a job ticket.<br><br>When **Separations** is *false* and **ColorantOrder** is not provided, the planes represented by the union of colorants specified in **ColorantParams** and implied by the **ProcessColorModel** are rendered in a device specified order. |

*Table 15 ColorantControl attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **CP** (ColorantParams) | array | *(Optional; required if* **ProcessColorModel** *is* **DeviceN***)* Array of names that indicate the colorants available on the printing press for which colorant planes are being produced. At one extreme, the contents of **ColorantParams** may specify all colorants available on the target printing press, including process colorants; at the other extreme, the contents may specify only non-process colorants. Printing systems should assume that the colorants available on the target printing press is the union of the colorants explicitly provided in **ColorantParams** and those implied by the color model specified in **ProcessColorModel**. <br><br> When **Separations** is *false*, non-process colorants named in **ColorantParams** will be marked. A request for a non-process colorant(s) not supported by the device may result in failure, or the colorant may be rendered as a process color. |
| **CSS** (ColorSpaceSubstitutes) | array | *(Optional)* An array of **ColorSpaceSubstitute** objects (page 56). |

PJTF 1.1

*Table 15 ColorantControl attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **PCM** (ProcessColorModel) | name | *(Optional)* Specifies the model to be used for rendering colorants specified in color spaces into process colorants. **ProcessColorModel** affects rendering for all color spaces except **Separation** and **DeviceN** colorspaces. (**Separation** and **DeviceN** color spaces mark directly into named colorant when these named colorants are included in the **ColorantParams** array.) **ProcessColorModel** does not affect the interpretation of color values in any color space; rather, it controls the rendering method.<br><br>If **Separations** is *true*, **ProcessColorModel** specifies the color model (typically **DeviceCMYK**) of the press on which the separations will be printed. If **Separations** is *false*, this color model should reflect the color model of the device on which the resulting proofs will be printed.<br><br>Legal values are **DeviceGray**, **DeviceRGB**, **DeviceRGBK**, **DeviceCMY**, **DeviceCMYK** and **DeviceN**. If no value is provided, **ProcessColorModel** depends on the value of **Separations**.<br><br>When **Separations** is *true*, **ProcessColorModel** defaults to *DeviceCMYK*.<br><br>When **Separations** is *false*, the default **ProcessColorModel** is device-dependent.<br><br>When **Separations** is *false*, **ProcessColorModel** is absent, and the content is pre-separated, **ProcessColorModel** defaults to *DeviceCMYK*. |
| **S** (Separations) | boolean | *(Optional)* If *true*, directs the device to produce each page by creating multiple color separations, one for each colorant specified by **ColorantOrder**. If *false*, directs the device to produce each document page as a single composite page with all colors combined on the same page.<br><br>The default value for **Separations** is *false*. |

PJTF 1.1

## 5.18 ColorantAlias Objects

PJTF 1.1

**ColorantAlias** objects are used to replace named colorants in **Separation** and **DeviceN** color spaces in PDL files. Emulations in the existing color spaces are not affected.

Use of **ColorantAlias** objects allow the job ticket to refer to colorant names consistently. Colorant names are replaced within **Separation** or **DeviceN** color spaces, and within Type 5 Halftone dictionaries.

Implicit process colorant names requested using language features which paint using a process color model are not replaced (e.g., there is no way to replace the implicit request for, say, Cyan when the PDF **K** operator is used.)

**ColorantAlias** objects are not used to replace any colorant names in the job ticket. All colorant names in the job ticket refer to the uniform name space for named colorants which results from the application of **ColorantAlias** objects.

**ColorantAlias** objects can occur as the elements in the **ColorantAliases** array in **ColorantControl** objects.

*Table 16 ColorantAlias attributes*

| Key | Type | Semantics |
|---|---|---|
| **A** (Aliases) | array of names | *(Required)* Each entry in the array is the name of a colorant which, if found in a color space or Type 5 Halftone dictionary, is to be replaced by the colorant specified by the **ReplacementColorantName**. |
| **RCN** (ReplacementColorantName) | name | *(Required)* The name of the colorant which is to be substituted for the colorants named in the **Aliases** array. |

## 5.19 ColorSpaceSubstitute Objects

PJTF 1.1

**ColorSpaceSubstitute** objects provide a mechanism for the job ticket to specify replacements for color spaces which occur in the input files for the job ticket.

A **DeviceN** color space is replaced if its **names** array is identical to the **TargetColorantNames** array. A **Separation** color space is replaced if its **name** value is identical to the contents of the **TargetColorantNames** array.

The replacement color space is identified by **ColorSpace** and must have the same number of color components as the color space it is replacing.

**ColorSpaceSubstitute** objects occur as the elements in the **ColorSpaceSubstitutes** array in **ColorantControl** objects.

*Table 17 ColorSpaceSubstitute attributes*

| Key | Type | Semantics |
|---|---|---|
| **CS** (ColorSpace) | name | *(Required)* The name used in the **ResourceAlias** object (page 25) which specifies the color space. |
| **TCN** (TargetColorantNames) | array of names | (Required) An array of colorant names. These colorant names describe the color spaces which are to be replaced by the color space resource specified by the **ColorSpace** key. |

## 5.20 PlaneOrder Objects

PJTF 1.1

**PlaneOrder** objects identify a sequence of colorant planes which occur within a **JTFile** which has been pre-separated, with all color planes represented in a single file.

Multiple **PlaneOrder** objects may be needed to fully describe the sequence of color planes which make up a pre-separated file. This will generally be the case when one or more blank separations were suppressed when the file was generated.

The **Planes** array lists the names of the colorant planes in the order in which they occur in the file as separate PDL pages.

The **Count** key identifies the number of times the sequence of colorant planes defined by **Planes** repeats for the sequence described by this **PlaneOrder** object.

The total number of virtual pages in a **JTFile** which has a **PlaneOrder** array is the sum of the **Count** values for all **PlaneOrder** objects in the array.

The total number of PDL pages in a **JTFile** which has a **PlaneOrder** array is the sum of the product, for each **PlaneOrder** object, of the **Count** value times the number of entries in the **Planes** array, for all **PlaneOrder** objects in the array.

The value *All* indicates the presence of composite pages in the **File**. Each instance of the value *All* indicates a single composite page.

Only the last **PlaneOrder** object in the **PlaneOrder** array for a **JTFile** object may omit the **Count** key.

In the case where the Count key is not present in the last **PlaneOrder** object in the **PlaneOrder** array for a **JTFile** object, neither the number of virtual pages in the **File** referenced, nor the total number of PDL pages in the **File** referenced, can be determined by parsing the Job Ticket.

**PlaneOrder** objects can occur as the values in the **PlaneOrder** key for **JTFile** objects.

*Table 18 PlaneOrder attributes*

| Key | Type | Semantics |
|---|---|---|
| **Co** (Count) | integer | *(Optional)* A positive integer which indicates the number of times the sequence of **Planes** occurs in the file. Absence of this key indicates that the sequence of **Planes** specified repeats to the end of the file. |
| **Pl** (Planes) | array | *(Required)* An array of names which identifies the colorant planes which occur in the file.<br><br>The value *All* indicates the presence of composite pages in the **File**. Each instance of the value *All* indicates a single composite page.<br><br>Typically, other entries in the **Planes** array will correspond to entries in the **ColorantParams** array of some **ColorantControl** object.<br><br>Note that the colorant names in this array are subject to colorant aliasing as specified in the **ColorantControl** object for the job. |

## 5.21 MediaSource Objects

**MediaSource** objects enable a device to select from available media sources. Job Ticket Processors can select a medium that best matches the values of keys in the object.

**MediaSource** objects can occur as the value for the **MediaSource** key for **JobTicketContents**, **Layout, Signature, Sheet, Tile** or **SlipSheet** objects.

*Table 19 MediaSource attributes*

| Key | Type | Semantics |
|---|---|---|
| **Cl** (Class) | string | *(Optional)* Product-specific classification of media, which may influence rendering. For example, transparent or glossy media may affect the selection of a color rendering method or a post-rendering technique specific to the device. |
| **LE** (LeadingEdge) | number | *(Optional)* Specifies the size, in points (1/72 in), of the media's edge that represents the scanline direction. If this key is absent, the scanline direction is assumed to be along the X axis (of **Dimensions** parameter). Use of **LeadingEdge** can accommodate different media orientations while avoiding changes to the **CTM** of each **PlacedObject** when placing page content onto a **Surface** for a given **Media**::**Dimensions** array. |
| **MF** (ManualFeed) | boolean | *(Optional)* When *true*, indicates that the device shall wait for input from an operator. |
| **Me** (Media) | dictionary | *(Optional)* A **Media** object (page 59). Describes the type of media to use. If **Position** has been specified, the value of **Media** indicates the type of media which should be loaded in the designated media source. |
| **Po** (Position) | integer | *(Optional)* In a device that has numbered input sources, identifies which source to use. |

## 5.22 Media Objects

**Media** objects specify attributes of the media itself.

**Media** objects can occur as the value for the **Media** key for **MediaSource** objects.

*Table 20 Media attributes*

| Key | Type | Semantics |
|---|---|---|
| **Ct** (Category) | text | *(Optional)* A string identifying a user- or site-specific type of media, such as *LetterHead*, *3-hole*, or *Transparency.* |

*Table 20 Media attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **Dm** (Dimensions) | array | *(Optional)* An array of four non-negative numbers [*minX minY maxX maxY*] describing an acceptable range of widths (*between minX and maxX inclusive*) and heights (*between minY and maxY inclusive*) for the medium, expressed in points (1/72 inch). A medium of fixed width X and fixed height *Y* is specified as [*X Y X Y*].<br><br>In simple printing, the X,Y values imply the content orientation.<br><br>For roll-fed media, the device shall advance the media at least the minimum value (either minX or minY) for the direction determined by **MediaSource**::**LeadingEdge**. |
| **MC** (MediaColor) | string | *(Optional)* An arbitrary string identifying the color of the medium. |
| **We** Weight) | number | *(Optional)* The weight of the medium, in grams per square meter. |

## 5.23 MediaUsage Objects

**MediaUsage** objects specify roll-fed media will be advanced or cut.

**MediaUsage** objects can occur as the value for the **MediaUsage** key for **JobTicketContents, Signature, Sheet** or **Tile** objects.

*Table 21 MediaUsage attributes*

| Key | Type | Semantics |
|---|---|---|
| **AD** (AdvanceDistance) | number | *(Optional)* Non-negative number indicating the distance in points (1/72 inch) to advance the media (beyond the distance determined by the device based on **Media**::**Dimensions** and **MediaSource**::**LeadingEdge**.)<br><br>If **CutMedia** is *true*, the media shall be advanced by at least **AdvanceDistance** prior to cutting. |
| **CM** (CutMedia) | boolean | *(Optional)* Indicates whether or not to cut the media. |

*Table 21 MediaUsage attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **MP** (MirrorPrint) | boolean | *Obsolete in version 1.1.* **MirrorPrint** *now occurs in* **Rendering** *objects (page 64).* <br><br> *(Optional)* If *true*, the device produces a page image that is reflected along one of the axes of device space. This is accomplished by device-dependent means that are independent of the graphics state (the CTM in particular). **MirrorPrint** is always applied before any transformation is applied during imaging. | PJTF 1.0 |
| **NP** (NegativePrint) | boolean | *Obsolete in version 1.1.* **NegativePrint** *now occurs in* **Rendering** *objects (page 64).* <br><br> *(Optional)* If *true*, the device produces a negative image of the page. This is accomplished by device-dependent means that are independent of the graphics state (the transfer function in particular). | PJTF 1.0 |

## 5.24 InsertPage Objects

An InsertPage object specifies whether to image the page content onto Even or Odd value of **Ord** of a **PrintLayout Signature**. Only one page image may be inserted to ensure the document set is printed at the specified position. This inserted page image will be blank when there is no **PageRange** key in this object.

Odd or Even refers to the value of **Ord** (of a **PlacedObject**). The first page of the **Signature** (**Ord** 0) will be designated as Even.

**InsertPage** objects can occur as the value for the **InsertPage** key for **JobTicketContents**, **Document** or **PageRange** objects. This object is only applied when **PrintLayout** is the printing method.

*Table 22 InsertPage attributes*

| Key | Type | Semantics | |
|---|---|---|---|
| **IS** (InsertSheet) | dictionary | *Obsolete in version 1.1.* <br><br> *(Optional)* An **InsertSheet** object (page 62). Inserts a sheet after each copy of the job. | PJTF 1.0 |

*Table 22 InsertPage attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **NP** (NewPage) | name | (Optional) One of the following: | |
| | | *Odd*, *Even* | Indicate that up to 1 page image will be inserted, so that the next page from the **Documents** structure will have the requested reader order. If the next **Ord** is consistent with **NewPage**, this request will have no effect (no page image will be inserted). |
| | | ~~*FillSheet*~~ | Obsolete in version 1.1. ~~Indicates that up to N-1 page images will be inserted on the current sheet, so that the next page from the **Documents** structure will be the first page on a new sheet.~~ |
| **PR** (PageRange) | dictionary | *(Optional)* A **PageRange** (page 43**)** object. Specifies the contents of the inserted page when not blank. Only the first page of this object will be inserted. This object must always use the dictionary form of its **JTFile** object. | |

PJTF 1.0

## 5.25 InsertSheet Objects

**InsertSheet** specifies how to complete a sheet currently being imaged so that a new sheet may be used for subsequent page images. It may indicate if new media will be inserted prior to imaging of the **Document** pages. Any new media inserted may be blank or have page content. If imaged, its page content may come from the object where it is used (**Documents** structure) or the **InsertContent** itself. Applies only to **PrintLayout** or simple printing method.

**InsertSheet** objects can occur as the value of **NewSheet** or **Trailer** keys for **JobTicketContents**, **Document**, and **PageRange** objects. A **Trailer** must have **InsertContent** defined when **BlankSheet** is *false*.

*Table 23 InsertSheet attributes*

| Key | Type | Semantics |
|---|---|---|
| **BS** (BlankSheet) | boolean | *(Optional)* Indicates whether the next **Sheet** shall be blank or not. If this key is absent, it is assumed to be *true*. This key must be present when the other keys are absent. |

*Table 23 InsertSheet attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **FC**<br>(FillContent) | dictionary | *(Optional)* A **PageRange** (page 43**)** object. Specifies the page image source used to complete a sheet currently being imaged (by **PrintLayout** method). The first and subsequent pages of this **PageRange** will be used until the current sheet is completely imaged. This object must always use the dictionary form of its **JTFile** object. If absent, the current sheet will be completed with blank content. | PJTF 1.1 |
| **IC**<br>(InsertContent) | dictionary | *(Optional)* A **PageRange** (page 43**)** object. Specifies the page image source used on the inserted sheet when **BlankSheet** is *false*. This data source will only be used if **Sheet** is present in this object. This object must always use the dictionary form of its **JTFile** object. If absent, the next sheet will either be blank or be imaged with content from **Documents** hierarchy, depending on the value of **BlankSheet**. | PJTF 1.1 |
| **PR**<br>(PageRange) | dictionary | *Obsolete in version 1.1*<br><br>*(Optional)* A **PageRange** (page 43**)** object. Specifies the contents of the page to be imaged on the sheet.<br><br>The **PageRange** will be ignored if BlankSheet is *true*. | PJTF 1.0 |
| **Sh**<br>(Sheet) | dictionary | *(Optional)* A **Sheet** (page 83**)** object. Specifies the attributes of the **Sheet** to be inserted. If this key is absent, the next sheet of the **Signature** will be used (or the first **Sheet** of the **Signature** if current sheet being imaged is the last **Sheet** of the **Signature**). If absent, no alternate media will be inserted (only **Sheet**(s) of the **Signature** will be used). | |

## 5.26 SlipSheet Objects

PJTF 1.1

**SlipSheet** objects specify media to be inserted after a **Finishing Operation** is complete.

**SlipSheet** objects occur as the value for **SlipSheet** key for **Finishing** objects.

*Table 24 SlipSheet attributes*

| Key | Type | Semantics |
|---|---|---|
| **A** (Alignment) | array | *(Optional)* An array of one integer and two numbers [Rot X Y] indicating how to align this sheet for **Finishing** operations (page 36). Rot is limited to 0, 90, 180, or 270 and indicates the clockwise rotation (in degrees) of this sheet relative to the coordinates used for the **Finishing** operations. X and Y indicate how to position the origin of the rotated sheet in the coordinate system for the **Front Surface**.<br><br>In the case of simple printing where the **Finishing** object is specified as part of a **JobTicketContents** object, the coordinate system which X and Y position the origin of the sheet onto is assumed to have its origin at its lower-left corner, with X increasing to the right and Y increasing upward. |
| **MS** (MediaSource) | dictionary | *(Optional)* A **MediaSource** object (page 58). |

## 5.27 Rendering Objects

**Rendering** objects provide device-specific rendering information.

While information provided within each of the dictionaries which may occur within a **Rendering** object (**DeviceRenderingInfo**, **PostRenderingEnhanceDetails**, **PreRenderingEnhanceDetails**) is device-specific, the contents of these dictionaries are not OEM extensions. Rather, each of these dictionaries is expected to have a **Type** key to identify itself, and devices will only honor one of these dictionaries when they support the **Type** specified. Other keys in these dictionaries need not reflect the PJTF extensions convention (i.e., such keys should not begin with OEM prefixes.)

**Rendering** objects can occur as the value for the **Rendering** key for **JobTicketContents, Document** or **PageRange** objects.

*Table 25 Rendering attributes*

| Key | Type | Semantics |
|---|---|---|
| **DRI** (DeviceRenderingInfo) | dictionary | *(Optional)* A dictionary that specifies device rendering parameters. |
| **MP** (MirrorPrint) | boolean | *(Optional)* If *true*, a page image is produced which is reflected around the scanline direction of the device. This is accomplished by device-dependent means. PJTF 1.1 |
| **NP** (NegativePrint) | boolean | *(Optional)* If *true*, a page is produced which is the negative image of the page. This is accomplished by device-dependent means. PJTF 1.1 |
| **Po** (PostRenderingEnhance) | boolean | *(Optional unless* **PostRenderingEnhanceDetails** *is present.)* If *true*, any enhancements available on the device will be invoked. |
| **PoD** (PostRenderingEnhanceDetails) | dictionary | *(Optional unless* **PostRenderingEnhance** *is present.)* Describes product-specific details related to post-rendering image enhancement. |
| **Pr** (PreRenderingEnhance) | boolean | *(Optional unless* **PreRenderingEnhanceDetails** *is present.)* If *true*, any enhancements available on the device will be invoked. |
| **PrD** (PreRenderingEnhanceDetails) | dictionary | *(Optional unless* **PreRenderingEnhance** *is present.)* Describes product-specific details related to pre-rendering image enhancement. |
| **R** (Resolution) | array | *(Optional)* Array of two numbers. **Resolution** indicates the resolution for the physical device to apply, expressed in pixels per inch. |
| **V** (ValuesPerColorComponent) | integer | (Optional) A positive integer indicating the number of values each color component may have, or in the monochrome case, the number of gray levels. |

## 5.28 Trapping Objects

PJTF 1.1

**Trapping** objects indicate whether trapping is requested, and, through the **TrappingDetails** object, specify colorant information for the trapping process.

**Trapping** objects can occur as the value for the **Trapping** key for **JobTicketContents, Document** or **PageRange** or in **Layout** and **PrintLayout** objects.

*Table 26 Trapping attributes*

| Key | Type | Semantics |
|---|---|---|
| **D** (Details) | dictionary | *(Optional)* A **TrappingDetails** object (page 66) containing global trapping parameters. |
| **T** (Trapping) | boolean | *(Optional)* If *true*, pages are trapped within defined areas according to trapping parameters for each area. The **Details** entry must be present and its parameters must be valid.<br><br>If *false*, pages are not trapped. The content of the **Details** entry is ignored and is not validated. |

## 5.29 TrappingDetails Objects

PJTF 1.1

**TrappingDetails** objects provide global parameters for processes which perform trapping.

**TrappingDetails** objects can occur as the value for the **Details** key for **Trapping** objects.

*Table 27 TrappingDetails attributes*

| Key | Type | Semantics |
|---|---|---|
| **CD** (ColorantDetails) | dictionary | *(Optional)* A **ColorantDetails** object (page 67). Defines trapping-related parameter values for individual device colorants specified in the job ticket. |
| **CSN** (ColorantSetName) | text | *(Optional)* Used by the user to identify the named colorant parameter set. |

*Table 27 TrappingDetails attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **TO** (TrappingOrder) | array of names | *(Optional)* Array of colorant names. The trapping engine will trap colorants as if they are laid down on the media in the order specified in **TrappingOrder**. The colorant order may affect which colors to spread, especially when opaque inks are used. This order can be different from the actual printing order on press or the order specified by **ColorantOrder**.<br><br>The first entry in the array identifies the first color on paper. After all colors listed in the array have been laid down, or if the array is empty, or if this key is not present, any additional colors are laid down in the order defined by **ProcessColorModel** and **ColorantParams**. Colors listed in the array but not part of **ProcessColorModel** and not found in **ColorantParams** are ignored. |
| **TT** (TrappingType) | integer | *(Optional)* Identifies the trapping method. The number identifies the minor (last three digits) and major (any digits prior to the last three) version of the trapping type requested.<br><br>Keys for the **TrappingParameters** object are subject to the trapping method identified here. Currently 1001 is the only recognized value and the **TrappingParameters** object description corresponds to that value. |

## 5.30 ColorantDetails Objects

**ColorantDetails** objects provide colorant information for specific colorant sets. PJTF 1.1

No extensions may be added to this dictionary. The only allowed keys are the names of colorants. The names which occur in this dictionary reflect the results of any colorant aliasing specified by **ColorantAlias** objects in the job.

**ColorantDetails** objects can occur as the value for the **ColorantDetails** key for **TrappingDetails** objects.

*Table 28 ColorantDetails attributes*

| Key | Type | Semantics |
|---|---|---|
| *name of colorant* | dictionary | *(Optional)* A **DeviceColorant** object (page 68). Contains parameters for a colorant. |

## 5.31 DeviceColorant Objects

PJTF 1.1

**DeviceColorant** objects provide information about each named colorant (ink) used in the device.

**DeviceColorant** objects occur as values for the keys in **ColorantDetails** dictionaries (page 67).

*Table 29 DeviceColorant attributes*

| Key | Type | Semantics |
|---|---|---|
| **CN** (ColorantName) | name | *(Optional)* May be used to record the commercial name or identifier of the colorant (ink), such as the vendor name and the part number. All instances of this colorant are identified by the key in the **ColorantDetails** dictionary for which this **DeviceColorant** object is the value. |

*Table 29 DeviceColorant attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **CT** (ColorantType) | name | *(Optional)* Controls how trapping is done for this colorant. If no value is specified, the Job Ticket Processor provides a default value. Recognized values are: *Normal* — Marks made with this colorant, marks covered by this colorant, and marks on top of this colorant, are trapped. *Transparent* — Marks made with this colorant are ignored for trapping. The trap engine need not generate a color plane for this colorant. **Transparent** can be used for varnish. *Opaque* — Marks covered by this colorant are ignored for trapping. **Opaque** can be used for metallic inks. *OpaqueIgnore* — Marks made with this colorant, and marks covered by this colorant, are ignored for trapping. **OpaqueIgnore** can be used for metallic inks. |
| **ND** (NeutralDensity) | number | *(Optional)* A number in the range of 0.001 to 10. The neutral density of this colorant. If no value is specified, the Job Ticket Processor provides a default. |

## 5.32 TrappingParameters Objects

**TrappingParameters** objects specify the desired trapping behavior for each trap zone. PJTF 1.1

**TrappingParameters** dictionaries are referenced by name. Names are resolved up the hierarchy from the object where the reference occurs. Thus, if a **TrapRegion** object for a **PageRange** object refers to **TrappingParameters** set by name, the name is looked up, first in the **PageRange**, then, if not found, in the **Document** object above the **PageRange**, and finally, if not found, in the **JobTicketContents** object that is the parent of the **Document** and **PageRange**.

Note that a **TrapRegion** object which occurs in **PlacedObject** object can only refer to **TrappingParameters** objects which occur in a **Layout** or **PrintLayout** object, when the value of **JobTicketContents**::**TrappingSourceSelector** is either *Layout* or *PrintLayout*, respectively.

All Job Ticket Processors which support **TrappingType** of 1001 must provide a default set of trapping parameters, and the values for keys omitted from **TrappingParameters** objects are inherited from these defaults.

**TrappingParameters** objects can occur as values in the **TrappingParameters** dictionary in the **JobTicketContents, Document, PageRange, Layout** or **PrintLayout** objects.

*Table 30 TrappingParameters attributes*

| Key | Type | Semantics |
|---|---|---|
| **BCL** (BlackColorLimit) | number | *(Optional)* A number between 0 and 1 which specifies the lowest color value required for trapping a colorant according to the black trapping rule. This entry uses the subtractive notion of color, where 0 is white, or no colorant, and 1 is full colorant. |
| **BDL** (BlackDensityLimit) | number | *(Optional)* A positive number which specifies the lowest neutral density of a colorant for trapping according to the black trapping rule. |
| **BW** (BlackWidth) | number | *(Optional)* A positive number which specifies the trap width for trapping according to the black trapping rule. **BlackWidth** is specified in **TrapWidth** units; a value of 1 means that the black trap width is one **TrapWidth** wide. The resulting black trap width is subject to the same device limits as **TrapWidth**. |

*Table 30 TrappingParameters attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **CZD** (ColorantZoneDetails) | dictionary | *(Optional)* A dictionary where the keys are named colorants and the values are **ColorantZoneDetails** objects (page 75). As with the entries in the **TrappingDetails**::**ColorantDetails** dictionary, entries in this dictionary reflect the results of any named colorant aliasing specified.<br><br>Each entry defines parameters specific for one named colorant. If omitted for a specific colorant, the relevant parameters in the **TrappingDetails**::**ColorantDetails** dictionary are used.<br><br>If the colorant named is neither listed in the **ColorantParams** array, nor implied by the **ProcessColorModel**, for the **ColorantControl** object in effect when these **TrappingParameters** are applied, the entry is not used for trapping. |
| **E** (Enabled) | boolean | *(Optional)* Indicates whether trapping is enabled for zones which are defined with this parameter set. |
| **HN** (HalftoneName) | name or string | *(Optional)* A name which identifies a halftone object to be used when marking traps.<br><br>The name is the **ResourceName** key of some **ResourceAliases** object in the **ResourceAliases** array in the **JobTicketContents** (page 23) object. PJTF 1.1<br><br>If absent, the halftone in effect just before traps are marked will be used, which may cause unexpected results. |
| **IIT** (ImageInternalTrapping) | boolean | *(Optional)* If *true*, the planes of color images are trapped against each other.<br><br>If *false*, the planes of color images are not trapped against each other. |

*Table 30 TrappingParameters attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **IR** (ImageResolution) | integer | *(Optional)* A positive integer indicating the minimum resolution in dots per inch for downsampled images. Images can be downsampled by a power of 2 before traps are calculated. The downsampled image is used only for calculating traps, while the original image is used when printing the image. |

*Table 30 TrappingParameters attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **IT** (ImagemaskTrapping) | boolean | *(Optional)* Controls trapping when the **TrapZone** contains a stencil mask.<br><br>A stencil mask is a monochrome image in which each sample is represented by a single bit. The stencil mask is used to paint in the current color: image samples with a value of 1 are marked; samples with a value of 0 are not marked.<br><br>When is *false*, none of the objects covered by the clipped bounding box of the stencil mask are trapped. No traps are generated between the stencil mask and objects that the stencil mask overlays. No traps are generated between objects that overlay the stencil mask and the stencil mask. For all other objects, normal trapping rules are followed. Two objects on top of the stencil mask that overlap each other, may generate a trap, regardless of the value of this parameter.<br><br>When is *true*, objects are trapped to the stencil mask, and to each other. |
| **ITO** (ImageToObjectTrapping) | boolean | *(Optional)* Controls whether images are trapped to other objects. |
| **ITP** (ImageTrapPlacement) | name | *(Optional)* Controls the placement of traps for images. Recognized values are:<br>*Center* Trap is centered on the edge between the image and the adjacent object.<br>*Choke* Trap is placed in the image.<br>*Normal* Trap is placed in the adjacent object.<br>*Spread* Trap is based on the colors of the areas. |

*Table 30 TrappingParameters attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **STL** (SlidingTrapLimit) | number | *(Optional)* A number between 0 and 1. Specifies when to slide traps towards a center position. If the neutral density of the lighter area is greater than the neutral density of the darker area multiplied by the **SlidingTrapLimit**, then the trap slides. This applies to vignettes and non-vignettes. No slide occurs at 1. |
| **SL** (StepLimit) | number | *(Optional)* A number between 0 and 1. Specifies the smallest step required in the color value of a colorant to trigger trapping at a given boundary.<br><br>If the higher color value at the boundary exceeds the lower value by an amount that is equal or greater than the larger of 0.05 or **StepLimit** times the lower value (low + max (**StepLimit** * low, 0.05)), then the edge is a candidate for trapping. The value 0.05 is set to avoid trapping light areas in vignettes.<br><br>This entry is used when not specified by a **ColorantZoneDetails** dictionary explicitly for a colorant. |
| **TCS** (TrapColorScaling) | number | *(Optional)* A number between 0 and 1. Specifies a scaling of the amount of color applied in traps towards the neutral density of the dark area. 1 means the trap has the combined color values of the darker and the lighter area. 0 means the trap colors are reduced so that the trap has the neutral density of the darker area.<br><br>This entry is used when not specified by a **ColorantZoneDetails** dictionary explicitly for a colorant. |
| **TW** (TrapWidth) | number | *(Optional)* A positive number. Specifies the trap width in points (1/72 inch). Also defines the unit used in trap width specifications for certain types or objects, such as **BlackWidth**.<br><br>The valid range is usually at least 1 to 40 pixels. |

## 5.33 ColorantZoneDetails Objects

PJTF 1.1

ColorantZoneDetails objects specify overrides to trapping parameters for specific device colorants.

ColorantZoneDetails objects can occur as the values for keys in the **ColorantZoneDetails** dictionary in **TrappingParameters** objects.

*Table 31 ColorantZoneDetails attributes*

| Key | Type | Semantics |
|---|---|---|
| **SL** (StepLimit) | number | *(Optional)* A number between 0 and 1. Specifies the smallest step required in the color value of a colorant to trigger trapping at a given boundary.<br><br>If the higher color value at the boundary exceeds the lower value by an amount that is equal or greater than the larger of 0.05 or **StepLimit** times the lower value (low + max (**StepLimit** * low, 0.05)), then the edge is a candidate for trapping. The value 0.05 is set to avoid trapping light areas in vignettes.<br><br>If omitted, the **StepLimit** in the **TrappingParameters** objects is used. |
| **TCS** (TrapColorScaling) | number | *(Optional)* A number between 0 and 1. Specifies a scaling of the amount of color applied in traps towards the neutral density of the dark area. 1 means the trap has the combined color values of the darker and the lighter area. 0 means the trap colors are reduced so that the trap has the neutral density of the darker area.<br><br>If omitted, the **TrapColorScaling** in the **TrappingParameters** objects is used. |

## 5.34 TrapRegion Objects

PJTF 1.1

TrapRegion objects specify zones of pages to be trapped, as well as the **TrappingParameters** to be used when trapping the zone.

TrapRegion objects always occur in arrays. The order in which these objects occur in the array is significant. Specifically, when the geometry specified in one TrapRegion object overlaps that of another, the later TrapRegion object takes precedence, and its **TrappingParameters** are applied to the overlapped area.

The polygons specified in the **TrapZone** arrays may extend beyond the PDL page contents. For example, it may be desirable to create trap networks for the area between pages which are imposed onto a **Surface**.

TrapRegion objects can occur as elements in the **TrapRegions** array in **JobTicketContents, Document**, **PageRange** or **PlacedObject** objects.

*Table 32 TrapRegion attributes*

| Key | Type | Semantics |
|---|---|---|
| **TP** (TrappingParameters) | name | *(Required)* The name of a **TrappingParameters** object. Must be one of the names in a **TrappingParameters** dictionary. Depending on where the **TrapRegion** object exists, the **TrappingParameters** object referenced may exist in a **JobTicketContents**, **Document**, **PageRange**, **Layout** or **PrintLayout** object.<br><br>It is an error if the named **TrappingParameters** object cannot be found. |
| **TZ** (TrapZone) | array of arrays of numbers | *(Optional)* The value of this key defines a complex path, and each sub-array is one subpath. Each sub-array consists of at least 3 ordered pairs of numbers. Each pair of numbers is interpreted as a point which is one vertex of a closed polygon. There is an implicit **moveto** operation at the beginning of each sub-array and an implicit **closepath** operation at the end of each sub-array.<br><br>The **TrapZone** is the area which results when the path specified by the polygons is filled using the non-zero winding rule.<br><br>The vertices are specified in page coordinate space.<br><br>When absent:<br><br>If the **TrapRegion** object is attached to a **JobTicketContents**, **Document** or **PageRange** object, the **MediaBox** array defines the **TrapZone**.<br><br>If the **TrapRegion** object is attached to a **PlacedObject** object, the **ClipBox** array defines the **TrapZone**. |

## 5.35 Layout Objects

PJTF 1.1

**Layout** objects map the sequence of pages to a user-specified signature. The sequence of pages and rendering operations applied to them are defined by means of the **Documents** array and the corresponding set of **PageRange** objects.

The **Layout** mechanism provides the means for specifying printing on devices that image page contents to some intermediate medium, and then print to final media in a separate operation, (e.g., imagesetters).

In **Layout**, each page image to be printed is explicitly positioned on a **Surface** via a **PlacedObject** which references the page. And only pages which are explicitly referenced from **PlacedObjects** are imaged.

A **Layout** is executed only once.

**Layout** objects can occur as the value for the **Layout** key for **JobTicketContents** objects.

*Table 33 Layout attributes*

| Key | Type | Semantics |
|---|---|---|
| **Co** (ColorantControl) | dictionary | *(Optional)* A **ColorantControl** object (page 52). A **ColorantControl** object describes how to control output color rendering. |
| **Cp** (Copies) | integer | *(Optional)* A positive integer specifying the number of copies to produce at the **Layout** level. |
| **F** (Finishing) | array | *(Optional)* An array of **Finishing** objects (page 36). Each **Finishing** operation is specified in a separate object, and the array determines the order of operations. |
| **Si** (Signatures) | array | *(Optional; Required if there is no* **SubType** *key)* An array of **Signature** objects (page 81). The array defines the order in which the signatures are needed. |

*Table 33 Layout attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **St** (SubType) | string | *(Optional; Required if there is no* **Signatures** *key)* The name given to this layout by the imposition creator. If a **Layout** object (page 77) has a **SubType** key but no **Signatures** key, it is the responsibility of the Job Ticket Processor to create the **Layout** that the name represents. This key is ignored if the **Signatures** key is present. |
| **SCB** (SurfaceContentsBox) | rectangle | *(Optional)* This rectangle defines the area for all **Surfaces** in this **Layout** into which all marking, including printers marks, will occur. **PlacedObject**::**CTMs** transform page contents into the coordinate space defined by this box.<br><br>**SurfaceContentsBox** defaults to [0 0 X Y], where X and Y are the minX and minY values for the **Media**::**Dimensions** array for the **Sheet**. PJTF 1.1 |
| **T** (Trapping) | dictionary | *(Optional)* A **Trapping** object (page 66). The trapping settings described here are used to control trapping when **TrappingSourceSelector** is *Layout*. PJTF 1.1 |
| **TP** (TrappingParameters) | dictionary | *(Optional)* Each key is the name of a **TrappingParameters** set and each value is a **TrappingParameters** object (page 69). These objects specify the sets of trapping parameters which will be used to create trap networks for **PlacedObjects** in this **Layout**. |

## 5.36 PrintLayout Objects

**PrintLayout** provides a printing mechanism which differs from printing via **Layout**. See section 5.35, "Layout Objects" for a description of printing via **Layout**, and see section 3.2, "Layout and PrintLayout" for a comparison of **Layout** and **PrintLayout**.

The **PrintLayout** mechanism provides the means for specifying printing on devices that image and print to final media in a single operation.

In **PrintLayout**, **PlacedObjects** do not refer to specific pages; rather, the arrangement of **PlacedObjects** on **Surfaces** provides a template for imaging the PDL page contents onto media. And, when printing via **PrintLayout**, the entire sequence of pages defined by the **JobTicketContents**::**Documents** array is printed.

The **PrintLayout** template is executed (filled) repeatedly, until all the pages from the page sequence have been consumed.

Each time the **PrintLayout** is executed, the same number of pages are consumed from the sequence of pages for the job. The number of pages consumed is **MaxOrd** +1. When **MaxOrd** is absent, the number of pages consumed is determined by the values of **PlacedObject**::**Ord** for the **Surfaces** in the **PrintLayout**. Specifically, the largest value of **Ord** for a **PlacedObject** in the **PrintLayout**, which does not have a **MarkDoc** key, provides the default for **MaxOrd**, and this value +1 is the number of pages consumed.

Multiple **PlacedObjects** in the **PrintLayout** may reference the same page (have the same **Ord** value.)

Alternate media may be introduced as the result of **InsertSheet** objects which occur for the **NewSheet** or **Trailer** keys in the in **JobTicketContents, Document** or **PageRange** objects.

Refer to Appendix B, “PrintLayout Details” for more information on how **PrintLayout** objects are used to control printing.

**PrintLayout** objects can occur as the value for the **PrintLayout** key for **JobTicketContents** objects.

*Table 34 PrintLayout attributes*

| Key | Type | Semantics |
|---|---|---|
| **Co** (ColorantControl) | dictionary | *(Optional)* A **ColorantControl** object (page 52). A **ColorantControl** object describes how to control output color rendering. |
| **Cp** (Copies) | integer | *(Optional)* A positive integer specifying the number of copies to produce at the **PrintLayout** level. |

*Table 34 PrintLayout attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **F** (Finishing) | array | *(Optional)* An array of **Finishing** objects (page 36). Each **Finishing** operation is specified in a separate object, and the array determines the order of operations. | |
| | | When a **PrintLayout** specifies **SubType** but not **Signature**, finishing can be applied to the **PrintLayout**. | PJTF 1.0 |
| **IS** (InsertSheet) | dictionary | *Obsolete in version 1.1.* | PJTF 1.0 |
| | | *(Optional)* An **InsertSheet** object (page 62). Inserts a sheet after each **Signature** or **SubType**. | |
| **MO** (MaxOrd) | integer | *(Optional) Identifies* the number of pages consumed each time the **PrintLayout** is repeated (**MaxOrd** + 1), and specifies the largest allowable **Ord** value for a **PlacedObject** in the **PrintLayout** which does not reference a **MarkDoc**. | PJTF 1.1 |
| | | The default value for **MaxOrd** is the largest **Ord** value for any **PlacedObject** in the **PrintLayout**::**Signature** which does not reference a **MarkDoc**. | |
| **Si** (Signature) | dictionary | *(Optional; Required if there is no* **SubType** *key)* A **Signature** object (page 81). | |
| **St** (SubType) | string | *(Optional; Required if there is no* **Signature** *key)* The name given to this **PrintLayout** by the imposition creator. If a **PrintLayout** object has a **SubType** key but no **Signature** key, it is the responsibility of the Job Ticket Processor to create the **Signature** that the name represents. This key is ignored if the **Signature** key is present. | |

Table 34 PrintLayout attributes (Continued)

| Key | Type | Semantics | |
|---|---|---|---|
| **SCB** (SurfaceContentsBox) | rectangle | *(Optional)* This rectangle defines the area for all **Surfaces** in this **PrintLayout** into which all marking, including printers marks, will occur. **PlacedObject**::**CTMs** transform page contents into the coordinate space defined by this box.<br><br>**SurfaceContentsBox** defaults to [0 0 X Y], where X and Y are the minX and minY values for the **Media**::**Dimensions** array for the **Sheet**. | PJTF 1.1 |
| **T** (Trapping) | dictionary | *(Optional)* A **Trapping** object (page 66). The trapping settings described here are used to control trapping when **TrappingSourceSelector** is *PrintLayout*. | PJTF 1.1 |
| **TP** (TrappingParameters) | dictionary | *(Optional)* Each key is the name of a **TrappingParameters** set and each value is a **TrappingParameters** object (page 69). These objects specify the sets of trapping parameters which will be used to create trap networks for **PlacedObjects** in this **PrintLayout**. | PJTF 1.1 |

## 5.37 Signature Objects

**Signature** objects describe finishing and media management operations for signatures.

A signature is the prescribed method for imposing images (of the document objects) onto actual media.

**Signature** objects occur as the value for the **Signature** key in **PrintLayout** objects or as elements in the **Signatures** array in **Layout** objects.

Table 35 Signature attributes

| Key | Type | Semantics |
|---|---|---|
| **Co** (ColorantControl) | dictionary | *(Optional)* A **ColorantControl** object (page 52). A **ColorantControl** object describes how to control output color rendering. |

*Table 35 Signature attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **Cp** (Copies) | integer | *(Optional)* A non-negative integer specifying the number of copies of this **Signature** to produce. |
| **F** (Finishing) | array | *(Optional)* An array of **Finishing** objects (page 36). Each **Finishing** operation is specified in a separate object, and the array determines the order of operations. **Signature** object **Finishing** operations are performed on all of the sheets that make up a signature. For example, a **Fold** instruction here calls for the sheets to be folded together. |
| **IS** (InsertSheet) | dictionary | *Obsolete in version 1.1.* PJTF 1.0<br>*(Optional)* An **InsertSheet** object (page 62). Inserts a sheet after each copy of the job. |
| **MS** (MediaSource) | dictionary | *(Optional)* A **MediaSource** object (page 58) for this **Signature**. The medium described here is used when individual **Sheets** do not specify a **MediaSource**. |
| **MU** (MediaUsage) | dictionary | *(Optional)* A **MediaUsage** object (page 60) for this **Signature**. Indicates whether roll-fed media will be advanced and/or cut; the media usage specified here applies only when individual **Sheets** do not specify a **MediaUsage** object. |
| **S** (Sheets) | array | *(Optional)* Required if there is no **SubType** key. An array of **Sheet** objects (page 83). |
| **St** (SubType) | string | *(Optional)* Required if there is no **Sheets** key. The name given to this signature style. If a **Signature** object has a **SubType** key but no **Sheets** key, it is the responsibility of the Job Ticket Processor to create the signature that the name represents. |

*Table 35 Signature attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **SCB** (SurfaceContentsBox) | rectangle | *(Optional)* This rectangle defines the area, in surface coordinate space for all **Surfaces** of this **Signature**, into which all marking, including printers marks, will occur. **PlacedObject**::**CTMs** transform page contents into this rectangle. **SurfaceContentsBox** defaults to [0 0 X Y], where X and Y are the minX and minY values for the **Media**::**Dimensions** array for the **Sheet**. |

PJTF 1.1

## 5.38 Sheet Objects

**Sheet** objects describe finishing and media management operations for sheets in a signature.

**Sheets** may comprise one or two **Surfaces** (**Front** and **Back**). The **Sheet**::**LockOrigins** key, **SurfaceContentsBoxes** and **PlacedObject**::**CTMs** for the **Sheet** interact to control the placement of PDL page contents onto each **Surface** for the **Sheet**.

The coordinate space of a **Front Surface** is always set up with its origin at its lower left corner, and its **SurfaceContentsBox** is defined in the first quadrant of the coordinate space.

The coordinate space of a **Back Surface** varies, depending on the value of **Sheet**::**LockOrigins**: when *false*, the coordinate space is set up the same as for a **Front Surface** (origin at lower left, **SurfaceContentsBox** in the first quadrant); when *true*, the coordinate space is set up with its origin at lower right and its **SurfaceContentsBox** in the second quadrant.

In all cases, **PlacedObject**::**CTMs** position page contents into the **SurfaceContentsBox** for the **Surface**.

See section 3.4, “Front/Back Alignment” for more information on placing page contents onto **Front** and **Back Surfaces**.

A **Sheet** which does not specify a **Front** or **Back Surface** will be a blank sheet (most likely to occur for **PrintLayout** use).

**Sheet** objects occur as elements in the **Sheets** array in **Signature** objects, or as the value for the **Sheet** key in **InsertSheet** objects.

*Table 36 Sheet attributes*

| Key | Type | Semantics |
|---|---|---|
| **A** (Alignment) | array | *(Optional)* An array of one integer and two numbers [Rot X Y] indicating how to align this sheet for **Finishing** operations (page 36). Rot is limited to 0, 90, 180, or 270 and indicates the clockwise rotation (in degrees) of this sheet relative to the coordinates used for the **Finishing** operations. X and Y indicate how to position the origin of the rotated sheet in the signature coordinate system. |
| **B** (Back) | dictionary | *(Optional)* Required if there is no **Front** key. The **Surface** object (page 85) that represents the back of the sheet. |
| **Co** (ColorantControl) | dictionary | *(Optional)* A **ColorantControl** object (page 52). A **ColorantControl** object describes how to control output color rendering. |
| **Cp** (Copies) | integer | *(Optional)* A non-negative integer specifying the number of copies of this **Sheet** to produce. |
| **F** (Finishing) | array | *(Optional)* An array of **Finishing** objects (page 36). Each **Finishing** operation is specified in a separate object, and the array determines the order of operations. |
| **Fr** (Front) | dictionary | *(Optional)* Required if there is no **Back** key. The **Surface** object (page 85) that represents the front of the sheet. |

*Table 36 Sheet attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **LO** (LockOrigins) | boolean | (*Optional*) Indicates whether the origin of the **Back Surface** shall be aligned with the origin of the **Front Surface**.<br><br>When **LockOrigins** is *true*, the **PlacedObject**::**CTMs** for the **Back Surface** image page contents into the second quadrant, with the origin at the lower right corner of **Surface**.<br><br>When **LockOrigins** is *false*, the **PlacedObject**::**CTMs** for the **Back Surface** image page contents into the first quadrant, with the origin at the lower left corner of **Surface**.<br><br>The default value for **LockOrigins** is *false*. |
| **MS** (MediaSource) | dictionary | *(Optional)* A **MediaSource** object (page 58) for this **Sheet**. |
| **MU** (MediaUsage) | dictionary | *(Optional)* A **MediaUsage** object (page 60) for this **Sheet**. Indicates whether roll-fed media will be advanced and/or cut. |
| **SCB** (SurfaceContentsBox) | rectangle | *(Optional)* This rectangle defines the area, in surface coordinate space for all **Surfaces** in this **Sheet**, into which all marking, including printers marks, will occur. **PlacedObject**::**CTMs** transform page contents into this rectangle.<br><br>**SurfaceContentsBox** defaults to [0 0 X Y], where X and Y are the minX and minY values for the **Media**::**Dimensions** array for the **Sheet**. |

PJTF 1.1

## 5.39 Surface Objects

**Surface** objects specify the manner of positioning pages of a **Signature** onto **Sheets**.

When both a **Front** and **Back Surface** occur for a **Sheet**, the **Front Surface** precedes the **Back Surface**.

See section 5.38, "Sheet Objects" for a discussion of the coordinate space of **Surfaces**.

**Surface** objects occur as the values of the **Front** and **Back** keys in **Sheet** objects.

*Table 37 Surface attributes*

| Key | Type | Semantics |
|---|---|---|
| **Co** (ColorantControl) | dictionary | *(Optional)* A **ColorantControl** object (page 52). A **ColorantControl** object describes how to control output color rendering. |
| **PO** (PlacedObjects) | array | *(Optional)* An array of **PlacedObject** objects (page 86). The array specifies the order in which to place the PDL page images onto the **Surface**. |
| **SCB** (SurfaceContentsBox) | rectangle | *(Optional; Required if Sheet::LockOrigins is true and this is a Back Surface)* This rectangle defines the area, in surface coordinate space for this **Surface**, into which all marking, including printers marks, will occur. **PlacedObject**::**CTMs** transform page contents into this rectangle.<br><br>When this is a **Back Surface** and **Sheet**::**LockOrigins** is *true* the **SurfaceContentsBox** must occur in the second quadrant of surface coordinate space. Otherwise, **SurfaceContentsBox** must occur in the first quadrant of surface coordinate space.<br><br>**SurfaceContentsBox** defaults to [0 0 X Y], where X and Y are the minX and minY values for the **Media**::**Dimensions** array for the **Sheet**. PJTF 1.1 |
| **T** (Tiling) | array | *(Optional)* An array of **Tile** objects (page 90). Each tile is a portion of the surface to be imaged as a separate sheet (or piece of film). These separate sheets are typically assembled after the job finishes. The array specifies the order in which to produce the tiles.<br><br>If the **Surface** object is part of a **PrintLayout** hierarchy, this key is ignored. PJTF 1.1 |

## 5.40 PlacedObject Objects

**PlacedObject** objects describe the attributes of images placed on the **Surface**.

**PlacedObject** objects occur in **Layout** or **PrintLayout** hierarchies. In either case, they define the placement of 'virtual' pages drawn from **JTFile**s onto **Surface**s. For pre-separated files, the mechanisms provided by the **FilesDictionary** and **PlaneOrder** keys in the **JTFile** object provide for the placement of individual colorant planes onto the **Surface**.

There are three steps needed to image page contents onto a **Surface**. First the contents are clipped using **PlacedObject**::**SourceClipPath**. Next, they are transformed into the coordinate space of the **Surface**::**SurfaceContentsBox** using **PlacedObject**::**CTM**. Finally, the transformed page image is clipped using **PlacedObject**::**ClipBox**. If both **PlacedObject**::**ClipBox** and **PlacedObject**::**SourceClipPath** are absent, no clipping of the page contents will occur.

PlacedObject objects occur as elements in the **PlacedObjects** array in **Surface** objects.

*Table 38 PlacedObject attributes*

| Key | Type | Semantics |
|---|---|---|
| **Cl** (ClipBox) | rectangle | *(Optional)* A rectangle defining a clip path in the coordinates of the **Surface**::**SurfaceContentsBox** that the device sets up before imaging the page. |
| **CTM** | array or dictionary | *(Required)* When an array, an array of six numbers. A coordinate transformation matrix mapping the coordinates of the PDL page referred to by the **PlacedObject**'s **Doc** and **Ord** keys to **Surface**::**SurfaceContentsBox** coordinates.<br><br>When a dictionary, the keys are the names of the colorants specified by the **ColorantControl** object in the **Surface**, **Sheet**, **Signature**, **Layout** or **PrintLayout** and the values are arrays of six numbers defining the transformation matrix for that colorant. |

*Table 38 PlacedObject attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **D** (Doc) | integer | *(Required if used within a* ***Layout*** *hierarchy and* ***MarkDoc*** *is absent.)* An index into the **Documents** array in the **JobTicketContents** object (page 27). Indicates the document in which the page is located. When this key is present, **Ord** refers to a virtual page in the specified document.<br><br>It is an error if both **MarkDoc** and **Doc** are present. It is an error if **Doc** is present and this object is within the **PrintLayout** hierarchy. PJTF 1.1 |
| **MD** (MarkDoc) | integer | *(Required if used within a* ***Layout*** *hierarchy and* ***Doc*** *is absent.)* An index into the **MarkDocuments** array in the **JobTicketContents** object (page 27). Indicates the document in which the mark is located. When this key is present, **Ord** refers to a virtual page in the specified mark document.<br><br>May be used when the **PlacedObject** is part of a **PrintLayout** (page 78) hierarchy, since this allows marks which are not part of the full sequence of pages defined by the **Documents** array to be imaged. PJTF 1.1 |

*Table 38 PlacedObject attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **O** (Ord) | integer | *(Required)* A non-negative integer *N* as the document page reference. The first page of a document has the value 0.<br><br>When neither **Doc** or **MarkDoc** keys are present, **Ord** reflects a relative page reference of reader pages used in a **PrintLayout**::**Signature**. **Ord** must be less than the value of **MaxOrd** when this object occurs in a **PrintLayout** hierarchy. It is an error if neither **Doc** or **MarkDoc** are present and this **PlacedObject** is used in a **Layout** hierarchy.<br><br>When the **Doc** key is present, **Ord** refers to the (N+1)th page listed in the ordered sequence defined by the **Pages** arrays in the **Document** objects specified in **JobTicketContents**::**Documents** array.<br><br>When the **MarkDoc** key is present, **Ord** refers to the (N+1)th page listed in the ordered sequence defined by the **Pages** arrays in the **Document** objects specified in **JobTicketContents**::**MarkDocuments** array.<br><br>For a **PlacedObject** within a **Layout** hierarchy, it is an error if this integer specifies a page which falls outside the range of total pages in the file. |
| **SC** (SourceClipPath) | array of arrays of numbers | *(Optional)* Each array consists of ordered pairs of numbers. Each pair of numbers is interpreted as a point which is one vertex of a closed polygon. There is an implicit **moveto** operation at the beginning of each array and an implicit **closepath** at the end of each array. PJTF 1.1<br><br>The **SourceClipPath** is the area which results when the path specified by the polygons is filled using the non-zero winding rule.<br><br>**SourceClipPath** defines a clipping path in the page coordinate space units of the page referenced by **Ord**, which is applied to the contents before they are transformed by the **CTM**. |

*Table 38 PlacedObject attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **TD** (TrappingDescription) | text | *(Optional)* A descriptive name to apply to the trap network which will be produced by a trapping application as a result of the **TrapRegion** objects for this **PlacedObject**. | PJTF 1.1 |
| **TR** (TrapRegions) | array | *(Optional)* An array of **TrapRegion** objects (page 75). These objects specify the trapping regions and trapping parameters which will be used to create trap networks for this **PlacedObject**.<br><br>Note that trap networks are created for a page only when there is at least one **TrapRegion** object for that page. | PJTF 1.1 |

## 5.41 Tile Object

**Tile** objects describe the position, extent, and media used for tiles. PJTF 1.1

Tiling occurs in some production environments when pages are imaged on to an intermediate medium and the resulting image of the **Surface** is larger than the media. In this case, instructions are needed to determine how the intermediate media (tiles) will be assembled to achieve the desired output (i.e., a single plate for the **Surface**). For example, a device might require that 4 pieces of film be assembled to create the image for the plate.

**Tile** objects occur as elements in the **Tiling** array in **Surface** objects.

*Table 39 Tile attributes*

| Key | Type | Semantics |
|---|---|---|
| **Cl** (ClipBox) | rectangle | *(Required)* A rectangle that defines a bounding box describing this tile in the **Surface**::**SurfaceContentsBox** coordinate system. Only marks lying within the bounding box are imaged. |
| **CTM** | array | *(Required)* An array of six numbers. A coordinate transformation matrix mapping the **ClipBox** (which is defined in the **Surface**::**SurfaceContentsBox** coordinate system) to the rectangle [*0 0 X Y*], where X and Y are the minX and minY values for the **MediaSource**::**Media**::**Dimensions** for the **Tile**. |

*Table 39 Tile attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **F** (Finishing) | array | *(Optional)* An array of **Finishing** objects (page 36). Each **Finishing** operation is specified in a separate object, and the array determines the order of operations. |
| **MR** (MarkRefs) | array | *(Optional)* An array of non-negative integers. Each entry in the array specifies an element from the **Surface** object's **PlacedObjects** array (page 86) to image on this tile. The first **PlacedObject** has the value 0.<br><br>If present, process only the specified placed objects. If absent, all placed objects may be processed to determine whether or not they intersect the tile's **ClipBox**. |
| **MS** (MediaSource) | dictionary | *(Optional)* A **MediaSource** object (page 58). |
| **MU** (MediaUsage) | dictionary | *(Optional)* A **MediaUsage** object (page 60). Indicates whether roll-fed media will be advanced and/or cut. |

## 5.42 Preflight Object

PJTF 1.1

**Preflight** objects describe the specification and results of preflight operations performed on documents. Preflighting is the process of determining the resources and capabilities needed to print the file.

Preflighting must be performed by an application which can parse the documents which are referenced by the job ticket.

See section 3.9, "Preflight Information" on page 19 for more information on the use of **Preflight** objects.

**Preflight** objects occur as the value of the **JTFile**::**Preflight** key.

*Table 40 Preflight attributes*

| Key | Type | Semantics |
|---|---|---|
| **A** (Analysis) | array | *(Optional)* An array of **Analysis** objects (page 94). Each **Analysis** object indicates the results of one preflight inspection. |

*Table 40 Preflight attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **I** (Inventory) | dictionary | *(Optional)* An **Inventory** object (page 92). The entries in this dictionary group information about the file. |
| **P** (Profile) | array | *(Required)* An array of **Profile** objects (page 93). Each **Profile** describes one set of constraints which could be used to perform an analysis. |

## 5.43 Inventory Objects

PJTF 1.1

An **Inventory** object exhaustively describes a file. The information included in the **Inventory** is determined by the preflight application, and may vary by **FileType**.

The **Colors**, **Document**, **FileType**, **Fonts**, **Images** and **Pages** keys in the **Inventory** object categorize the results. The values are arrays of **PreflightResults** objects.

**Inventory** objects occur as the value for the **Inventory** key in **Preflight** objects.

*Table 41 Inventory attributes*

| Key | Type | Semantics |
|---|---|---|
| **C** (Colors) | dictionary | *(Optional)* A **PreflightResults** object (page 97). Within the **PreflightResults** object, **PreflightDetail** and **PreflightInstance** objects describe characteristics of the colorants used. |
| **D** (Document) | dictionary | *(Optional)* A **PreflightResults** object (page 97). Within the **PreflightResults** object, **PreflightDetail** and **PreflightInstance** objects identify characteristics of the document as a whole. |
| **FT** (FileType) | dictionary | *(Optional)* A **PreflightDetail** object (page 98). The **Value** key for this object must be a string object which identifies the type of data in the file as determined by the preflighting process.<br><br>The string in the **Value** array must be a MIME file type as recorded by the Internet Assigned Numbers Authority (IANA). IANA has procedures for registering new file types if needed. |
| **F** (Fonts) | dictionary | *(Optional)* A **PreflightResults** object (page 97). Each element in this array groups information about fonts which occurs in the file. |

*Table 41 Inventory attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **I** (Images) | dictionary | *(Optional)* A **PreflightResults** object (page 97). Within the **PreflightResults** object, **PreflightDetail** and **PreflightInstance** objects group information about images which occurs in the file. |
| **ID** (InventoryDate) | date | *(Optional)* A date/time stamp indicating when the inventory operation was performed. |
| **P** (Pages) | dictionary | *(Optional)* A **PreflightResults** object (page 97). Within the **PreflightResults** object, **PreflightDetail** and **PreflightInstance** objects provide page-related information which the application collects. |

## 5.44 Profile Objects

PJTF 1.1

**Profile** objects specify a set of constraints against which the file may be tested. The semantics for constraints are specific to the **FileType** of the **JTFile** object. See Appendix F, “Preflight Semantics for PDF” for a language-specific example.

The **Colors**, **Document**, **FileType**, **Fonts**, **Images** and **Pages** keys in the **Profile** object categorize the constraints. The value for each of these keys is an array of **PreflightConstraint** (page 96) objects. Within the **PreflightConstraint** objects, the **ConstraintValue** key indicates allowable values and the **Status** key indicates the error level (if any) to be flagged when exceptions to the constraints are identified.

**Profile** objects occur as the value for the **Profile** key in **Preflight** objects.

*Table 42 Profile attributes*

| Key | Type | Semantics |
|---|---|---|
| **C** (Colors) | array | *(Optional)* An array of **PreflightConstraint** objects (page 96). Each element in this array identifies a specific **Colors** constraint against which to test the file. |
| **D** (Document) | array | *(Optional)* An array of **PreflightConstraint** objects (page 96). Each element in this array identifies a specific **Document** constraint against which to test the file |

*Table 42 Profile attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **F** (Fonts) | array | *(Optional)* An array of **PreflightConstraint** objects (page 96). Each element in this array identifies a specific **Fonts** constraint against which to test the file |
| **FT** (FileType) | dictionary | *(Optional)* A **PreflightConstraint** object (page 96). The **Value** key for this object must be of type array and must contain string objects which identify the allowable types of data in the file.<br><br>The strings in the **Value** array must be MIME file types as recorded by the Internet Assigned Numbers Authority (IANA). IANA has procedures for registering new file types if needed. |
| **I** (Images) | array | *(Optional)* An array of **PreflightConstraint** objects (page 96). Each element in this array identifies a specific **Images** constraint against which to test the file |
| **P** (Pages) | array | *(Optional)* An array of **PreflightConstraint** objects (page 96). Each element in this array identifies a specific **Pages** constraint against which to test the file |

## 5.45 Analysis Objects

PJTF 1.1

**Analysis** objects describe the results of a preflight operation. The constraints against which the file was tested are specified by the corresponding **Profile** object identified by **AnalysisProfile**.

The **Colors**, **Document**, **FileType**, **Fonts**, **Images** and **Pages** keys in the **Analysis** object categorize the results. The values are arrays of **PreflightResults** objects.

Within the **PreflightResults** object, elements of the **PreflightDetails** array describes the analysis result for one **PreflightConstraint**, indicating the value determined and the resulting error level.

Within the **PreflightResults** object, elements of the **PreflightInstances** array, each **PreflightInstance** object groups the analysis results for one instance of an object type or file characteristic. Results for all **PreflightConstraints** for that instance occur within the **PreflightInstance**::**Properties** dictionary.

**Analysis** objects occur as the values in the **Preflight**::**Analysis** array.

*Table 43 Analysis attributes*

| Key | Type | Semantics |
|---|---|---|
| **AD** (AnalysisDate) | date | *(Optional)* A date/time stamp indicating when the analysis operation was performed. |
| **AM** (AnalysisMode) | name | *(Optional)* Indicates how the analysis operation was performed. Recognized values are: **Exhaustive** All exceptions to constraints are identified and recorded in the **Analysis**. **FirstError** Only the first exception to constraints are identified and recorded in the **Analysis**. If this key is absent, the value **Exhaustive** is assumed. |
| **AP** (AnalysisProfile) | dictionary | *(Required)* The **Profile** object (page 93) which was used for this analysis operation. Must be one of the elements of the **Preflight**::**Profile** array. |
| **C** (Colors) | dictionary | *(Optional)* A **PreflightResults** object (page 97). Within the **PreflightResults** object, **PreflightDetail** and **PreflightInstance** objects array provide results for specific **Colors PreflightConstraint**. |
| **D** (Document) | dictionary | *(Optional)* A **PreflightResults** object (page 97). Within the **PreflightResults** object, **PreflightDetail** and **PreflightInstance** objects provide results for specific **Document PreflightConstraint**. |
| **FT** (FileType) | dictionary | *(Optional)* A **PreflightDetail** object (page 98). The **Value** key for this object must be a string object which identifies the type of data in the file as determined by the preflighting process. The string in the **Value** array must be a MIME file type as recorded by the Internet Assigned Numbers Authority (IANA). IANA has procedures for registering new file types if needed. |
| **F** (Fonts) | dictionary | *(Optional)* A **PreflightResults** object (page 97). Within the **PreflightResults** object, **PreflightDetail** and **PreflightInstance** objects provide results for **Fonts PreflightConstraints**. |

*Table 43 Analysis attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **Im** (Images) | dictionary | *(Optional)* A **PreflightResults** object (page 97). Within the **PreflightResults** object, **PreflightDetail** and **PreflightInstance** objects provide results for **Images PreflightConstraints**. |
| **P** (Pages) | dictionary | *(Optional)* A **PreflightResults** object (page 97). Within the **PreflightResults** object, **PreflightDetail** and **PreflightInstance** objects provide results for specific **Pages PreflightConstraints**. |

## 5.46 PreflightConstraint Objects

PJTF 1.1

**PreflightConstraint** objects provide the details for the constraints input of preflight operations.

When **PreflightConstraint** objects occur inside **Profile** objects they represent the constraints used (or to be used) by preflight operations. Constraints may be expressed in numerous ways. For example, constraints might be expressed as: one specific value, a enumeration of allowable values, or a pair values indicating a continuous range of allowable values.

**PreflightConstraint** objects occur as the value for the **Colors**, **Document**, **FileType, Fonts**, **Images** and **Pages** keys in **Profile** objects.

*Table 44 PreflightConstraint attributes*

| Key | Type | Semantics |
|---|---|---|
| **C** (Constraint) | any | *(Required)* The semantics of this key are specific to each constraint and to the page description language of the file being preflighted. |
| **CV** (ConstraintValue) | any | *(Required)* The semantics of this key are specific to each constraint and to the page description language of the file being preflighted. |

*Table 44 PreflightConstraint attributes (Continued)*

| Key | Type | Semantics | |
|---|---|---|---|
| **S** (Status) | name | *(Required)* Must be **Error**, **Warning**, **Ignore** or **IgnoreValue**. | |
| | | *Error* | If the preflight process discovers a result which violates the constraint, the result is to be flagged as an *Error*. |
| | | *Warning* | If the preflight process discovers a result which violates the constraint, the result is to be flagged as a *Warning*. |
| | | *Ignore* | The constraint is ignored – no **PreflightDetail** or **PreflightInstanceDetail** objects are created for this constraint. |
| | | *IgnoreValue* | No comparison is made against the **ConstraintValue** for this constraint. **PreflightDetail** or **PreflightInstanceDetail** objects which are created for this constraint provide information about the file and are flagged with **IgnoreValue Status** values. |

## 5.47 PreflightResults Objects

PreflightResults objects are used to separate results for Inventory and Analysis sub-objects into an array of PreflightDetails objects and an array of PreflightInstance objects.

PreflightResults objects occur as the value for the **Colors**, **Document**, **FileType, Fonts**, **Images** and **Pages** keys in **Inventory** and **Analysis** objects.

*Table 45 PreflightResults attributes*

| Key | Type | Semantics |
|---|---|---|
| **PD** (PreflightDetails) | array | *(Optional)* An array of **PreflightDetail** objects (page 98). |
| **PI** (PreflightInstances) | array | *(Optional)* An array of **PreflightInstance** objects (page 99). |

## 5.48 PreflightDetail Objects

PJTF 1.1

**PreflightDetail** objects provide the details for the results of preflight operations.

When **PreflightDetail** objects occur inside **Inventory** objects they provide information about one characteristic of the file and the **Status** key is absent.

When **PreflightDetail** objects occur inside **Analysis** objects they provide the results of the preflight operation. When **Status** is **Error** or **Warning**, they indicate that the **Value** violated the **PreflightConstraint**::**ConstraintValue**. When **Status** is **IgnoreValue**, no comparison was performed, and the **PreflightDetail** object merely provides information about the file characteristic (as in an **Inventory**).

**PreflightDetail** objects occur as the value for the **FileType** key and as elements in the **PreflightResults**::**PreflightDetails** array for the **Colors**, **Document**, **Fonts**, **Images** and **Pages** keys in **Inventory** and **Analysis** objects.

*Table 46 PreflightDetail attributes*

| Key | Type | Semantics |
|---|---|---|
| **P** (Property) | any | *(Required)* The semantics of this key are specific to each constraint and to the page description language of the file being preflighted. |
| **PR** (PageRef) | array | *(Optional)* An array of numbers indicating the pages where error or warning conditions occur in the file. When the **JTFile** references a pre-separated file (through use of the **PlaneOrder** key or the **FilesDictionary** key), the pages referenced are 'virtual composite' pages.<br><br>The value of the **Analysis**::**AnalysisMode** may indicate whether the list is exhaustive. |
| **S** (Status) | name | *(Optional; required if the PreflightDetail object is part of an Analysis.)* Must be *Error*, *Warning* or *IgnoreValue* if present.<br><br>*Error* — If the preflight process discovers a result which violates the constraint, the result is to be flagged as an *Error*.<br><br>*Warning* — If the preflight process discovers a result which violates the constraint, the result is to be flagged as a *Warning*.<br><br>*IgnoreValue* — The **Value** for this characteristic of the file was not compared against a **PreflightConstraint**::**ConstraintValue**. |

*Table 46 PreflightDetail attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **V** (Value) | any | *(Required)* The semantics of this key are specific to each constraint and to the page description language of the file being preflighted. |

## 5.49 PreflightInstance Objects

PJTF 1.1

**PreflightInstance** objects collect all details for a specific instance of some file characteristic. For example, one **PreflightInstance** object would provide all the requested information about each font used in a file.

When **PreflightInstance** objects occur inside **Analysis** objects they represent the results of the preflight operation. Like constraints, results may be expressed in various ways. Generally, an array of **PreflightInstance** objects will occur as the value for a single key inside a sub-dictionary of an **Analysis** object.

For example, the **ProhibitedNames** key is defined as a constraint within the **Fonts** sub-dictionary for PDF files, and a single **PreflightInstance** object provides the list of prohibited font names. However, an array of **PreflightInstance** objects would occur within the **Analysis** object (assuming that more than one prohibited font was identified by the preflighting tool), one for each prohibited font which was identified.

When **PreflightInstance** objects occur inside **Inventory** objects they represent the results of the preflight operation. Like constraints, results may be expressed in various ways. Generally, an array of **PreflightInstanceDetail** objects will occur as the value for a single key inside a sub-dictionary of an **Inventory** object.

**PreflightInstance** objects occur as elements in the **PreflightResults**::**PreflightInstances** array for the value of the **Colors**, **Document**, **FileType, Fonts**, **Images** and **Pages** keys in **Inventory** and **Analysis** objects.

*Table 47 PreflightInstance attributes*

| Key | Type | Semantics |
|---|---|---|
| **I** (Identifier) | string | *(Required)* This key identifies the specific instance of the object type or file characteristic which is being described by this **PreflightInstance** object and the **PreflightInstanceDetail** objects in its **Properties**. |

*Table 47 PreflightInstance attributes (Continued)*

| Key | Type | Semantics |
|---|---|---|
| **P** (Properties) | dictionary | *(Required)* A dictionary of **PreflightInstanceDetail** objects (page 98).<br><br>When the **PreflightInstance** object is part of an **Inventory**, each key in this dictionary identifies one property for the instance.<br><br>When the **PreflightInstance** object is part of an **Analysis**, each key in this dictionary identifies one **PreflightConstraint** which the file was checked against. |
| **PR** (PageRef) | array | (Optional) An array of numbers indicating the pages where the item described by this **PreflightInstance** object error or warning conditions occur in the file.<br><br>If **PageRef** is absent, then the item occurs on all pages in the file. |

## 5.50 PreflightInstanceDetail Objects

PJTF 1.1

**PreflightInstanceDetail** objects provide the details for the results of preflight operations for some instance of a type of object or file characteristic which occurs at least once in the file being preflighted.

When **PreflightInstanceDetail** objects occur inside **Inventory** objects they describe one property for the instance and the **Status** key is absent.

When **PreflightInstanceDetail** objects occur inside **Analysis** objects they represent the results of the preflight operation, and the **Status** key indicates whether or not a comparison was made, and if so, whether the result was an **Error** or **Warning**.

**PreflightInstanceDetail** objects occur as the values for the keys in **PreflightInstance**::**Properties** dictionaries.

*Table 48 PreflightInstanceDetail attributes*

| Key | Type | Semantics |
|---|---|---|
| **S** (Status) | name | *(Optional; required if the PreflightInstanceDetail object is part of an Analysis.)* Must be *Error*, *Warning* or *IgnoreValue* if present.<br>*Error* — If the preflight process discovers a result which violates the constraint, the result is flagged as an *Error*.<br>*Warning* — If the preflight process discovers a result which violates the constraint, the result is flagged as a *Warning*.<br>*IgnoreValue* — The **Value** for this characteristic of the instance was not compared against a **PreflightConstraint**::**ConstraintValue**. |
| **V** (Value) | any | *(Required)* The semantics of this key are specific to each constraint and to the page description language of the file being preflighted. |

APPENDIX A

# Extending Job Ticket Keys

This appendix illustrates the mechanism for extending job ticket keys. It first describes the goals that an OEM wishes to accomplish, then the method it chooses for reaching those goals. The name of the OEM is Yet Another Printer Company, represented by its initials, YAP.

## A.1 Background

In the example considered here, the OEM wants to specify the default font to use for font substitution when the **FontPolicy**::**UseDefaultFont** key is set to true and a font cannot be found.

The OEM decides to use the **FontPolicy** object to accomplish its goals.

## A.2 Method

A new key is added to the **FontPolicy** object, **YAP_DefaultFont**, using the corporate prefix YAP. This key provides the name of the font resource to use as the default font, using PostScript semantics.

APPENDIX B

# PrintLayout Details

This appendix describes the use of the PrintLayout object to control printing. Example PrintLayouts demonstrate how PrintLayout provide some forms of printing control that print drivers have provided for PostScript printing.

## B.1 Background

The PrintLayout object is used to specify printing for simple devices which image and print pages in one operation. Such devices ‘see’ the document as a stream of consecutive pages to be imaged onto the media.

The operations specified by the PrintLayout object hierarchy are applied to all the pages specified in the JobTicketContents::Documents objects. The sequence of pages specified represent the reader order of the resultant document

Signature, Sheet and PlacedObject objects are specified for a subset M of the pages. During printing, pages are imaged and printed, M at a time, until the entire document has been printed.

The following examples demonstrate how PrintLayout objects can be used to specify different printing operations.

## B.2 Example: Simplex

In this example, the pages are consumed one page at a time, and each page is printed on a separate piece of media.

Simplex printing is accomplished through the use of these objects and values:

MaxOrd = 0 (may be omitted)

1 Signature object

1 Sheet object

1 Surface object (Front or Back)

1 PlacedObject object with
- A single CTM on Surface
- An Ord value of 0

## B.3 Example: Simplex – only odd pages

In this example, pages are consumed two at a time, but only one page is printed for each set.

Simplex, odd-only printing is accomplished through the use of these objects and values:

MaxOrd = 1 (required to skip even pages)

1 Signature object

1 Sheet object

Front or Back Surface objects

1 PlacedObject object with:
- A single CTM to position the page contents on Surface
- An Ord value of 0

## B.4 Example: Duplex

In this example, pages are consumed two at a time and imaged onto both sides of a single piece of media. One piece of media is output for each two pages.

Duplex printing is accomplished through the use of these objects and values:

MaxOrd = 1 (may be omitted)

1 Signature object

1 Sheet object

Front and Back Surface objects

2 PlacedObject objects, 1 per Surface, with:
- Ord values of 0 and 1
- Differing CTMs to place page contents on each Surface

## B.5 Example: PseudoDuplex

This is a method for printing an original from a simplex printer, and then using a copier to create duplex copies (2 simplex sheets becomes 1 duplex sheet).

In this example, pages are consumed 2 at a time, and 2 printed pages are issued for each set.

PseudoDuplex printing is accomplished through the use of these objects and values:

MaxOrd = 1 (may be omitted)

1 Signature object

2 Sheet objects, each with:

1 Surface object

2 PlacedObject objects, 1 for each Surface with:

- Ord values of 0 and 1
- Different CTMs for the PlacedObject objects if the page image needs to be shifted for binding

## B.6 Example: Signature (N-up)

In this example, pages are consumed N (e.g., 2, 4, 8, 16 . . .) and imaged onto one side of a single piece of media. One piece of media is output for each N pages.

N-up printing is accomplished through the use of these objects and values:

MaxOrd = N–1 (may be omitted)

1 Signature object

1 Sheet object

1 Surface object (either Front or Back)

N PlacedObject objects with:

- Ord values of 0 thru N–1
- N CTMs to place each page image onto the Surface

APPENDIX C

# Page Imaging Mechanics

This appendix describes how pages are imaged onto media via **Layout** or **PrintLayout**.

## C.1 Background

The examples in this appendix demonstrate how page contents (as described by **TrimBox**) are translated by the **PlacedObject**::**CTM** into region of the **Surface** described by the **SurfaceContentsBox**. The examples show the effect of various values for **Sheet**::**LockOrigins** affect the specification of **SurfaceContentsBox**.

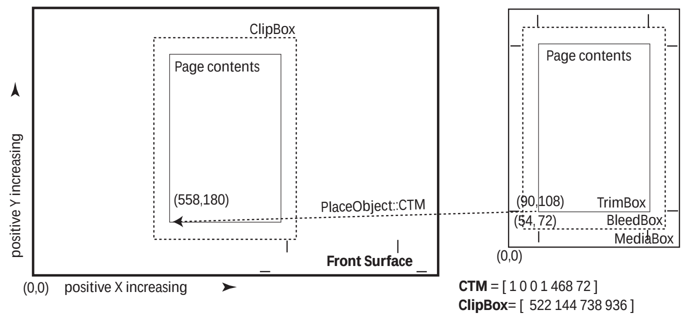
For simplicity, the examples demonstrate imaging an 8.5 by 11 inch page onto the **Front** and **Back Surfaces** for a 24 by 16 inch **Sheet**, except for the third example, in which the **Media** width is allowed to vary.

Each example is preceded by a table listing values for **PageRange**::**MediaBox**, **PageRange**::**TrimBox**, **Media**::**Dimensions**, **Surface**::**SurfaceContentsBox** and **PlacedObject**::**CTM**.

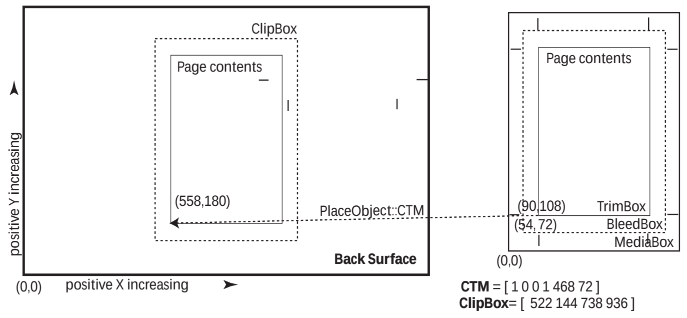
## C.2 Default case

In this case, **LockOrigins** is not provided (and defaults to *false*); the origin of each **Surface** is at the lower-left corner. **SurfaceContentsBox** is not provided, and the **CTMs** place the page images in the center of the **Surfaces**.

| OBJECT | KEY | VALUE |
|---|---|---|
| **PageRange** | **MediaBox** | *[0 0 792 1008]* |
| | **BleedBox** | *[54 72 738 936]* |
| | **TrimBox** | *[90 108 702 900]* |
| **Media** | **Dimensions** | *[1728 1152 1728 1152]* |
| **Surface** | **SurfaceContentsBox** | not provided – defaults to *[0 0 1728 1152]* for both **Surfaces** |



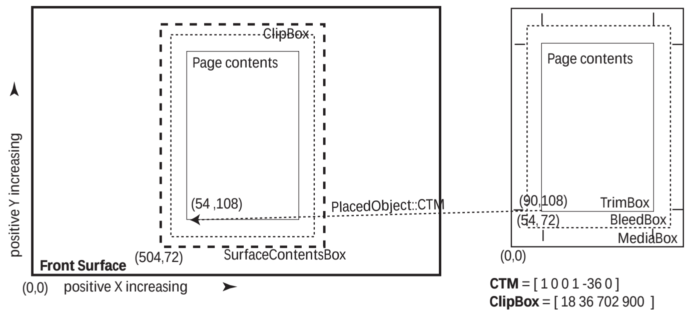
The **ClipBoxes** and the origins of the page contents (558, 180) are in the coordinate space of the **Surfaces**.

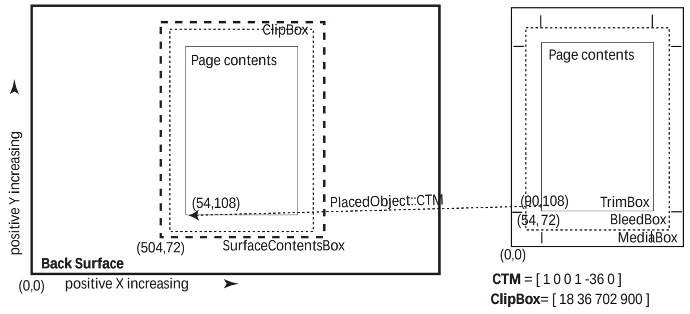


## C.3 LockOrigins false

In this case, **LockOrigins** is *false* and the origin of each **Surface** is at the lower-left corner. The **SurfaceContentsBox** is centered on each **Surface**, and each **CTM** places a page image in the center of the **SurfaceContentsBox**, which is 10 by 14 inches.

| OBJECT | KEY | VALUE |
|---|---|---|
| **PageRange** | **MediaBox** | *[0 0 792 1008]* |
| | **BleedBox** | *[54 72 738 936]* |
| | **TrimBox** | *[90 108 702 900]* |
| **Media** | **Dimensions** | *[1728 1152 1728 1152]* |
| **Surface** | **SurfaceContentsBox** | *[504 72 1224 1080]* for both **Surfaces** |

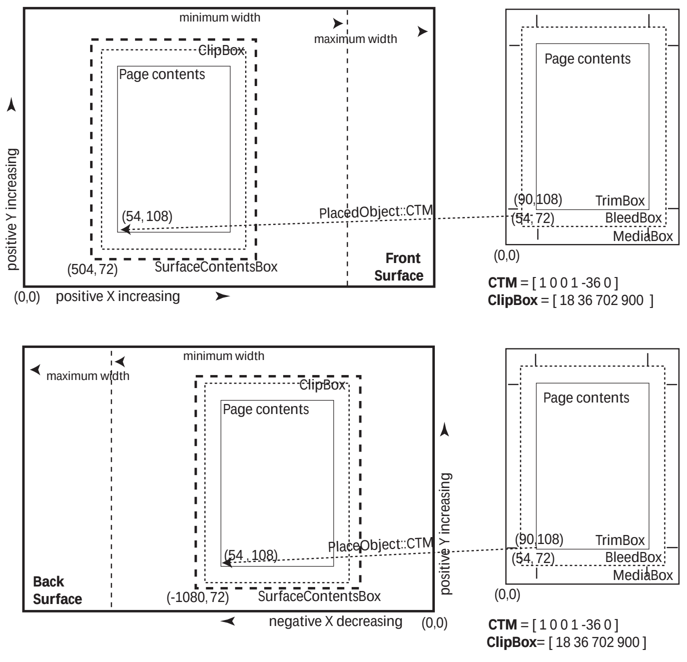




## C.4 LockOrigins true

In this case, **Sheet**::**LockOrigins** is *true*. The origin of the **Front Surface** is at it's lower-left corner; the origin of the **Back Surface** is at it's lower-right corner. The width of the Media is allowed to vary between 20 and 24 inches.

| OBJECT | KEY | VALUE |
|---|---|---|
| **PageRange** | **MediaBox** | *[0 0 792 1008]* |
| | **BleedBox** | *[54 72 738 936]* |
| | **TrimBox** | *[90 108 702 900]* |
| **Media** | **Dimensions** | *[1440 1152 1728 1152]* |
| **Surface** | **SurfaceContentsBox** | *[360 72 1080 1080]* for **Front Surface** |
| **Surface** | **SurfaceContentsBox** | *[-1080 72 -360 1080]* for **Back Surface** |



The **SurfaceContentsBoxes** are 10 by 14 inches. They are centered on the media, given the smallest allowable width.

This arrangement allows the **Front** and **Back Surface** content to be aligned, even though the **Media** width is not fixed.

APPENDIX D
# PostScript Equivalents

This appendix identifies the objects and keys in the Job Ticket specification which provide information which is directly analogous to information typically provided in PostScript page-description programs.

## D.1 setpagedevice Equivalencies

The PJTF object/keys listed in this table have direct analogs in PostScript **setpagedevice** calls.

| OBJECT | KEY | SETPAGEDEVICE PARAMETER(S) |
|---|---|---|
| Rendering | DeviceRenderingInfo | DeviceRenderingInfo |
|  | PostRenderingEnhance | PostRenderingEnhance |
|  | PostRenderingEnhanceDetail | PostRenderingEnhanceDetail |
|  | PreRenderingEnhance | PreRenderingEnhance |
|  | PreRenderingEnhanceDetails | PreRenderingEnhanceDetails |
|  | Resolution | HWResolution |
|  | ValuesPerColorComponent | DeviceRenderingInfo:: ValuesPerColorComponent |
| ColorantControl | ColorantOrder | SeparationOrder |
|  | ColorantParams | SeparationColorNames |

| OBJECT | KEY | SETPAGEDEVICE PARAMETER(S) |
|---|---|---|
| | ProcessColorModel | ProcessColorModel |
| | Separations | Separations |
| JobTicketContents, Document or PageRange | MediaBox | Approximates PageSize when there is no way to separate out the media from the intended imaging area |
| Finishing | Operation | This will encompass many pagedevice parameters as needed, e.g., Collate, Jog, Trim, Bind, Staple, Fold, Laminate |
| | Details | Details dictionaries for finishing pagedevice parameters (e.g., StapleDetails, BindDetails, FoldDetails.) |
| PlacedObject | CTM | CTM key will encompass several pagedevice parameters, as available, e.g., Orientation, Rotation, PageOffset, ImageShift, Margins |
| Media | Dimensions | PageSize<br>(Note that Dimensions allows a range of size, while PageSize is a specific size.) |
| | MediaColor | MediaColor |
| | Weight | MediaWeight |
| | Category | MediaType |

| OBJECT | KEY | SETPAGEDEVICE PARAMETER(S) |
|---|---|---|
| MediaSource | Class | MediaClass |
| | LeadingEdge – specifies axis of fast scan direction<br><br>(Note that LeadingEdge in Job Tickets only specifies the width and lets the CTM decide how the image is placed relative to this edge.) | Orientation – rotation to align to fast scan direction<br><br>LeadingEdge – selection of media that imposes fast scan direction |
| | ManualFeed | ManualFeed |
| | Position | MediaPosition |
| MediaUsage | AdvanceDistance | AdvanceMedia + AdvanceDistance |
| | CutMedia | CutMedia |
| Rendering | MirrorPrint | MirrorPrint |
| | NegativePrint | NegativePrint |
| Trapping | Trapping | Trapping |
| | Details | TrappingDetails |

The PJTF object/keys listed in this table have direct analogs in the TrappingDetails sub-dictionary for the PostScript **setpagedevice** calls.

| **OBJECT** | **KEY** | **SETPAGEDEVICE PARAMETER(S)** |
|---|---|---|
| TrappingDetails | ColorantDetails | TrappingDetails:: ColorantDetails |
| | ColorantSetName | TrappingDetails:: ColorantDetails:: ColorantSetName |
| | TrappingOrder | TrappingDetails:: TrappingOrder |
| | TrappingType | TrappingDetails::Type |

The PJTF object/keys listed in this table have direct analogs in the TrappingDetails::ColorantDetails sub-dictionary for the PostScript **setpagedevice** calls.

| **OBJECT** | **KEY** | **SETPAGEDEVICE PARAMETER(S)** |
|---|---|---|
| ColorantDetails | device colorant names | device colorant names in TrappingDetails:: ColorantDetails:: *(colorant name)* |
| DeviceColorant | ColorantName | TrappingDetails:: ColorantDetails:: *(colorant name)*:: ColorantName |
| | ColorantType | TrappingDetails:: ColorantDetails:: *(colorant name)*:: ColorantType |

| OBJECT | KEY | SETPAGEDEVICE PARAMETER(S) |
|---|---|---|
| | NeutralDensity | TrappingDetails:: ColorantDetails:: ColorantDetails:: *(colorant name)*:: NeutralDensity |

## D.2 settrapparams Equivalencies

The PJTF object/keys listed in this table have direct analogs among the arguments to the PostScript **settrapparams** operator.

| OBJECT | KEY | SETTRAPPARAMS PARAMETER(S) |
|---|---|---|
| TrappingParameters | All keys | Equivalent to keys in the PostScript TrappingParameters dictionary which is an argument to the settrapparams operator. |
| ColorantZoneDetails | StepLimit | TrappingParameters:: ColorantZoneDetails:: *colorantname::* StepLimit |
| | TrapColorScaling | TrappingParameters:: ColorantZoneDetails:: *colorantname::* TrapColorScaling |

APPENDIX E

# Mime Types for Some Common File Types

This appendix lists the MIME types (when available) for some common printer document formats.

The MIME types shown should be used as the value of **JTFile**::**FileType** objects which refer to files of the type shown. For example, the value of the **FileType** key in a **JTFile** object which refers to a PostScript file should be the string “application/postscript”.

## E.1 Mime Types

| TYPE OF FILE | MIME TYPE/SUBTYPE (for use in JTFile/FileType key) |
|---|---|
| Portable Document Format | application/pdf |
| PostScript | application/postscript |
| PCL | application/vnd.hp-PCL |
| PCL-XL | application/vnd.hp-PCLXL |
| DCA-RFT | application/DCA-RFT |
| TIFF-IT | None currently assigned |
| IPDS | None currently assigned |
| AFP | None currently assigned |

APPENDIX F

# Preflight Semantics for PDF

This appendix identifies the semantics for constraints which could be used to by a preflight tool to analyze pdf files. Note that this list of constraints provides examples of the various preflight categories only; it is not intended to be definitive, nor comprehensive.

## F.1 Document

Constraints relating to information about the file as a whole.

| DOCUMENT CONSTRAINT | TYPE | CONSTRAINT MEANING | RESULT MEANING |
|---|---|---|---|
| **Edited** | boolean | Determine whether the file has been edited since it was first created | A single **PreflightDetail** object indicates whether the file has been edited. |
| **MinVers** | number | Determine the minimum language version which must be supported by a application which will print or view this file | A single **PreflightDetail** object indicates the minimum version. |
| **Optimized** | boolean | Determine whether the file is optimized. | A single **PreflightDetail** object indicates whether the file is optimized. |
| **Producer** | text | Determine the application which produced the file | A single **PreflightDetail** object provides the name of the creator application. |

| DOCUMENT CONSTRAINT | TYPE | CONSTRAINT MEANING | RESULT MEANING |
|---|---|---|---|
| **Security** | array of names | Request security categories applied to the document | A **PreflightDetail** object contains an array of security categories applied to the document. |

## F.2 Colors

Constraints relating to color usage in the file.

| COLOR CONSTRAINT | TYPE | CONSTRAINT MEANING | RESULT MEANING |
|---|---|---|---|
| **Allowed** | array of names | An array of names which identifies the allowable color space families (e.g., **DeviceCMYK**, **DeviceN**) | A **PreflightInstance** object identifies each color space found in the file which does not match one of the allowed values. The **Identifier** key specifies the name of the color space family and the **PageRefs** array lists all pages where it is used.<br><br>Within the **Properties** dictionary for the **PreflightInstance** object, a **PreflightInstanceDetail** object occurs for the **Allowed** key – the **Value** key identifies the specific colorspace. |

## F.3 Fonts

Constraints relating to font usage in the file. Note that for all of these examples, results will be recorded in an array of **PreflightInstanceDetail** objects grouped within one **PreflightInstance** object for each font found in the file.

| FONT CONSTRAINT | TYPE | CONSTRAINT MEANING | RESULT MEANING |
|---|---|---|---|
| **Allowed** | array of names | Provides a complete set of names of fonts which are allowed in the document. | Within the **Properties** dictionary for a font, an **Allowed** entry identifies usage of each font not included in the ‘allowed’ list. |
| **Embedded** | boolean | Determine whether all fonts used in this file are embedded within it. | Within the **Properties** dictionary for a font, an **Embedded** entry indictates that the font is embedded. |
| **Prohibited Names** | array of names | Provides a set of names of fonts which should not occur in the document. | Within the **Properties** dictionary for a font, a **ProhibitedNames** entry indicates that the font is one of those which should not occur in the document. |
| **Subset** | boolean | Fonts which are subsetted in the file are identified.<br><br>When the value is false, fonts which are subsetted are flagged according to the value of **Status** in the **PreflightConstraint**.<br><br>When true, subsetting is expected. | Within the **Properties** dictionary for a font, a **Subset** entry indicates that the font is subsetted. |

## F.4 Images

Constraints relating to images in the file. Note that for all of these examples, results will be recorded in an array of **PreflightInstanceDetail** objects grouped within one **PreflightInstance** object for each image found in the file.

| IMAGE CONSTRAINT | TYPE | CONSTRAINT MEANING | RESULT MEANING |
|---|---|---|---|
| **ColorSpaces** | array | An array of names which identifies the allowable color space families (e.g., **DeviceCMYK**, **DeviceN**) | Within the **Properties** dictionary for the **PreflightInstance** object for an image, a **ColorSpaces** entry identifies a color space of a disallowed family used in the image. |
| **MaxRes** | number | Determine whether any image resolution exceeds a specified value | Within the **Properties** dictionary for the **PreflightInstance** object for an image, a **MaxRes** entry indicates that an image's resolution exceeds the specified maximum resolution. |
| **MinRes** | number | Determine whether any image resolution is lower than the specified value | Within the **Properties** dictionary for the **PreflightInstance** object for an image, a **MinRes** entry indicates that an image's resolution image's resolution is less than the specified minimum resolution. |

| IMAGE CONSTRAINT | TYPE | CONSTRAINT MEANING | RESULT MEANING |
|---|---|---|---|
| **Rotated** | boolean | Determine whether any image has been rotated. | Within the **Properties** dictionary for the **PreflightInstance** object for an image, a **Rotated** entry indicates that the image has been rotated. |

## F.5 Pages

Constraints relating to pages in the file.

| PAGE CONSTRAINT | TYPE | CONSTRAINT MEANING | RESULT MEANING |
|---|---|---|---|
| **AllowablePage-Sizes** | array of names | Determine whether any pages **MediaBox** specifies a disallowed paper size. | A **PreflightDetail** object identifies each page whose **MediaBox** is not allowed. |
| **MaxHorizontal** | number | Determine whether any pages exceed the specified maximum horizontal size (in default units for page coordinate space). | A **PreflightDetail** object identifies each page which exceed the specified dimension. |
| **MaxVertical** | number | Determine whether any pages exceed the specified maximum vertical size (in default units for page coordinate space). | A **PreflightDetail** object identifies each page which exceed the specified dimension. |

# Index