

Application Note

Use of standard Print Product Metadata (PPM) with PDF

1st Edition
2024-05



Print Product
Metadata LWG

© 2024 PDF Association – pdfa.org

This work is licensed under CC-BY-4.0 © ⓘ

Copyright © 2024 PDF Association or its licensors.

This work is licensed under the Creative Commons Attribution 4.0 International License.

To view a copy of this license, visit
<http://creativecommons.org/licenses/by/4.0/>

or send a letter to
Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

PDF Association
Friedenstr. 2A · 16321 Bernau bei Berlin · Germany

E-mail: copyright@pdfa.org
Web: www.pdfa.org

Published in Germany and the United States of America

Vendors own their respective copyrights wherever they are mentioned. The information contained in this document is provided only as general information, which may be incomplete or outdated. Users of this document are responsible for independently verifying any and all information. Any mention of companies or products does not imply endorsement or support of any of the mentioned information, services, products, or providers.

Table of Contents

1. Overview.....	1
2. Audience.....	1
3. Introduction.....	1
4. References.....	3
5. Definitions.....	4
6. Describing Print Products.....	5
6.1. Product Type.....	5
6.2. Layout Intent.....	6
6.3. Media Intent.....	6
6.4. Hole Making Intent.....	7
6.5. Binding Intent.....	7
6.6. Folding Intent.....	8
6.7. Colour Intent.....	8
6.8. Assembling Intent.....	8
7. Encoding PPM.....	9
7.1. DPart structure and DPM overview.....	9
7.2. Encoding PPM within DPM.....	10
7.3. Providing Contact Information.....	12
8. Print Product Appearance Rules.....	14
8.1. Concepts.....	14
8.2. Product Type.....	15
8.3. Layout Intent.....	17
8.4. Media Intent.....	18
8.5. Folding Intent.....	18
8.6. Hole making Intent.....	20
8.7. Binding Intent.....	21
8.8. Assembling Intent.....	22
8.9. Extensions.....	24
Appendix A: Romantic Koosbania example.....	25
Final print products.....	25
Page Content.....	26
PPM.....	27

1. Overview

This document provides an overview and technical introduction to the Print Product Metadata (PPM) ISO Standard (ISO 21812-1) which makes use of PDF features introduced in ISO PDF/VT and natively supported in PDF 2.0.

PPM is structured machine readable information stored within a PDF file that is used to describe the characteristics of a finished print product in relation to the PDF pages. PPM can be used to describe finished print products to the level of detail necessary to convey print product requirements for common use cases.

Using PDF with PPM to describe the finished print product along with its page content in a single PDF file, provides a concrete way to communicate print product design requirements among the various stake-holders involved in the print ecosystem — from print product designers, print buyers through to prepress and the various print production processes.

In the case of variable data printing (e.g. ISO PDF/VT), PPM can be used to describe both uniform or varied finished print product requirements per recipient, along with the varied page content.

It is expected that graphic design, MIS, prepress and production workflow tools that already support PDF, or make use of proprietary formats for describing print products for various purposes, will be extended to support this ISO standard and interoperable PPM format.

2. Audience

- Print providers — Commercial and Variable data printers
- Print Stakeholders — Brand owners, publishers, print buyers and print brokers
- Print product originators — Content creators, Marketers
- Solution developers, integrators
- Educational Institutions

3. Introduction

This Application Note is intended to provide guidance to software solution developers and stakeholders interested in understanding the purpose and high level technical details of the ISO PPM standard. This document is not intended as a substitute for the ISO PPM standard.

PDF is commonly used today for specifying final form page content for print applications and products. The format's ability to represent pages from simple black and white graphical content to fully color-managed page designs is well known and supported by most page design and layout applications, prepress workflow and production printing systems available today.

The expressiveness, flexibility and reliability of PDF, especially ISO's PDF/X, has led to its use as the preferred and universal file format for communicating graphical page content independent of proprietary formats supported by page layout applications.

Although standards, such as PDF and PDF/X, and related best practices have been in common use for exchanging color page content among print stakeholders, PDF and PDF/X alone cannot represent the detailed requirements of finished print products due to limitations of the format.

Typically such details would be stored outside of the PDF file using JDF or XJDF, or some proprietary and application-specific method.

ISO 32000-2 (PDF 2.0) introduced a standard way to specify structured relationships among the pages of a PDF file as well as a way to embed specialized metadata within that structure.

Based on CIP4's XJDF the PPM standard, defines a set of machine readable metadata definitions that can be embedded within a PDF 2.0, (including PDF/X-6) or PDF/VT file to specify the details of the finished print product(s) in relation to its PDF pages.

Prior to PPM's introduction, the common way to communicate finished print product requirements was either through ad-hoc methods such as a written description within an email, a Customer Service Representative (CSR), or perhaps through the use of an online portal's print product catalog or order entry form.

The use of PPM in PDF creates an opportunity for the print industry to modernize and improve communication of finished print product details among participants in the print ecosystem, with the potential to reduce effort and costs.

Conforming PDF files that include PPM will be readable, viewable and editable by any supporting graphic design applications and, of course, usable by downstream preflight, prepress and automated print production workflow software.

For example, a finished print product, such as a bound book, may be represented using PPM as a hierarchical assembly of print product parts such as its cover(s), body section(s), folded pages, etc. Specific content pages of the PDF file are mapped to the finished pages of each well defined part. The finished pages of a part might have specific paper type requirements, and the overall print product would have a defined binding style.

It is important to note that PPM is only used in the definition of the finished print product and makes no assumptions about any specific print manufacturing process or printing method. It is normally the responsibility of the print provider to determine an appropriate print manufacturing process based on their available print production capability.

The PPM standard was created as a non-proprietary means of specifying finished print product definitions that are portable and interoperable among software systems from different vendors.

The PPM standard is based on the subset of the CIP4 XJDF specification known as Product Intent, an existing print industry specification capable of describing the requirements of finished print products to various degrees of detail. More specifically, the PPM standard specifies the use of a subset of XJDF Product Intent properties and defines the technical details of how they are encoded for use within the internal document part structure of a PDF 2.0 or PDF/VT conformant file.

4. References

- XJDF Specification, Release 2.0, 2018, CIP4 Organization, Available from the internet at: <https://www.CIP4.org>
- ISO 21812-1 Graphic technology — Print product metadata for PDF files — Part 1: Architecture and core requirements for metadata
- ISO 16612-2 Graphic technology — Variable data exchange — Part 2: Using PDF/X-4 and PDF/X-5 (PDF/VT-1 and PDF/VT-2)
- ISO 16612-3 Graphic technology — Variable data exchange — Part 3: Using PDF/X-6 (PDF/VT-3)
- ISO 32000-2:2020, Document Management — Portable Document Format — Part 2, PDF 2.0
- ISO 19593-1 Graphic technology — Use of PDF to associate processing steps and content data — Part 1: Processing steps for packaging and labels

5. Definitions

- **Logical page** — individual page definition of the PDF file (not to be confused with “*logical order*” as used in *Tagged PDF*).
- **Finished page** — physical page of the print product (from the print product user’s perspective) that is a trimmed sheet of paper or print substrate that has two sides.
- **Spread** — single logical page that has content typically representing two finished pages, e.g. a center spread would represent a pair of facing pages.
- **Part (Print Product Part)** — represents one or more finished pages that are part of a print product such as a book cover, book block or folded finished pages.
- **Print product** — sequence of one or more print product parts where each part can itself be subdivided into a sequence of one or more parts.
- **Product Intent** — specifies the creator’s view of a finished print product.
- **Product Assembly** — defines a print product where one or more parts are inserted into another part, e.g. a letter into an envelope.
- **Document Part Metadata (DPM)** — set of PDF key value pairs defining the metadata associated with a Document Part.
- **Document Part** — set of related consecutive pages and/or related sets of pages of a PDF.
- **Document Part Hierarchy** — hierarchical PDF data structure that specifies the organization of document parts. The Document Part (DPart) hierarchy consists of a tree of nodes. Each page of the PDF refers to exactly one leaf node.
- **Print Product Metadata (PPM)** — Document Part metadata whose key value pairs are defined in the ISO standard 21812-1.
- **Root node** — parent node for all DPart nodes in the Document Part Hierarchy which also declares compliance to the PPM standard.
- **Print Product Metadata (PPM)** — metadata model for use in DPM as specified in the PPM standard.
- **Leaf node** — node in the Document Part Hierarchy that has no child nodes. Multiple consecutive PDF pages may refer to a single leaf node.
- **Intermediate node** — node of the Document Part Hierarchy that is neither the root node nor a leaf node.
- **Cover** — finished page visible from the outside of a finished print product, e.g. a book.
- **Body** — those pages of a finished print product that are not the cover.
- **Front Cover** — cover that is read before the body pages.
- **Back Cover** — cover that is read after the body pages.
- **Wraparound Cover** — single folded sheet of media into which the finished pages of the corresponding body part are inserted. Since in PDF it is defined by a single page it allows for printing on the side of the finished print product, e.g. the spine of a book.

6. Describing Print Products

This section provides a conceptual overview of the basic information that can be used in describing finished print products and their parts. The actual encoding and semantics of PPM are specified in sections [7](#) and [8](#).

PPM describes a print product as a sequence of one or more print product part definitions. Each print product part may itself be further described as a sequence of one or more part definitions. Each part describes one or more finished pages whose content is provided by the associated PDF pages (logical pages). The appearance of each part is specified using document part metadata associated with those logical pages.

PPM can be used to describe a complete or partial print product. PPM allows for a varying degree of specificity in terms of the completeness of the description of the print product. For example, specific attributes of the print product description may be left unspecified, allowing their values to be chosen later during the production preparation phase, perhaps based on availability of materials or equipment as well as additional out-of-band communication with stakeholders. In all cases, such out-of-band communication with stakeholders may influence the requirements of the print product that will ultimately be produced. In the absence of additional communication, the specified attributes in the print product description are expected to be honored. It is worth noting that the obligation to fully honor the print product definition as presented using PPM depends on the agreements between participating stakeholders.

6.1. Product Type

The product type of a part identifies or classifies the intended use of the print product (e.g. a book) or part of a print product (e.g. a front cover, body pages, or back cover, etc...). The following common product types are included in the PPM standard:

- | | |
|-----------------------|--------------------------|
| ■ <i>BackCover</i> | ■ <i>Jacket</i> |
| ■ <i>Body</i> | ■ <i>Label</i> |
| ■ <i>Book</i> | ■ <i>Leaflet</i> |
| ■ <i>BookBlock</i> | ■ <i>Letter</i> |
| ■ <i>BookCase</i> | ■ <i>Map</i> |
| ■ <i>Booklet</i> | ■ <i>Newspaper</i> |
| ■ <i>Box</i> | ■ <i>Notebook</i> |
| ■ <i>Brochure</i> | ■ <i>Postcard</i> |
| ■ <i>BusinessCard</i> | ■ <i>Poster</i> |
| ■ <i>Cover</i> | ■ <i>ResponseCard</i> |
| ■ <i>CoverLetter</i> | ■ <i>Section</i> |
| ■ <i>Envelope</i> | ■ <i>SelfMailer</i> |
| ■ <i>FrontCover</i> | ■ <i>Spine</i> |
| ■ <i>Insert</i> | ■ <i>WrapAroundCover</i> |

Note: The list of product types defined by the PPM standard should be used where possible even if this implies using a more generic product type (e.g. use “Letter” even when producing an “invoice”).

Although specifying the product type is optional, including this information in the PPM definition is strongly recommended.

6.2. Layout Intent

The layout intent is used to specify how pages are imaged onto the sides of the finished pages of a part. It specifies whether the finished pages will have content on the front side only, back side only or on both sides.

If both sides of the finished pages are to be imaged, the layout intent can also specify whether the content on the back side should be rotated.

The layout intent can also be used to specify that the PDF page content is for a spread, where spreads are used to specify the content for a wrap-around cover, a center page of a book, a fold-out page, or an envelope (prior to cutting, folding and gluing into the final form).

The layout intent can also specify the size of the finished part typically derived from or accommodates the dimensions defined by the pdf page.

In the case of folding, the size of the unfolded finished page is larger than that of the folded finished page. In this case, the size of the folded finished page is specified in the layout intent while the size of the unfolded finished page is taken from the definition of the PDF page.

Layout intent can also specify the thickness of the print product part using the Z dimension of the finished dimensions property. The Z dimension can be set to zero if a specific thickness of the print product is not required.

6.3. Media Intent

The media intent is used to specify the properties of the media (e.g. paper) of the finished pages of a part.

Properties of the media that can be specified such as:

- Weight
- Color
- Grade
- Front and/or back coating
- Media type details (e.g. pre-cut tabs, envelope, labels, ...)
- Media quality

The use of pre-printed media is considered as an aspect of a print production process, and is therefore out of scope for PPM.

The PDF content used for producing pre-printed media may be included in the PDF and marked (e.g. as optional content / layers representing the static page content background) in such a way that the production process can either print it inline or omit it when the content has already been printed and included as pre-printed media.

Note 1: Any holes in the printed product are specified by hole making intent; when and how the holes are made — such as pre-punched paper or drilling — is part of the production process, and is not defined by PPM.

Note 2: The dimensions of the media used during production must (of course) accommodate the print product dimensions as specified in layout intent but are not specified in PPM, and depend on the print production process.

6.4. Hole Making Intent

Hole making intent specifies the hole pattern and placement of holes for a finished part. Whether the holes are implemented in production through using pre-drilled media or added by drilling holes during the production process is up to the print service provider.

It is possible to specify hole making requirements for the overall finished print product, one or more finished parts, or just a single finished page. If hole making is specified on a child part as well as its parent part, both the holes specified on the parent part and the child part will apply to the child part.

PPM allows for specifying hole patterns for common use cases:

- No holes
- Ring binding — 2, 3, 4, 5, 6, 7 and 11 holes
- Plastic comb binding
- Wire comb binding
- Coil binding
- Spiral binding

In some cases of binding intent (e.g. ring binding, spiral/comb binding), hole making is implied, in which case, no explicit hole making intent is necessary.

6.5. Binding Intent

The binding intent is used to specify how and what part(s) should be bound together and in what order. This specification includes the binding side and method.

The following binding methods can be specified:

- *AdhesiveNote*
- *ChannelBinding*
- *CoilBinding*
- *CornerStitch*
- *EdgeGluing*
- *HardCover*
- *LooseBinding*
- *None*
- *PlasticComb*
- *RingBinding*
- *SaddleStitch*
- *SideStitch*
- *SoftCover*
- *StripBind*
- *Tape*
- *Wirecomb*

If the binding involves stitching or stapling (e.g. *SaddleStitch* or *SideStitch*) then the number of stitches or staples can be specified. The exact location of the stitches or staples along the binding side cannot be specified as this is considered a production decision. Similarly, for other forms of binding, such as coil binding, the exact number of holes cannot be specified.

6.6. Folding Intent

The folding intent can specify that a part should be folded. Folds implied by the binding method are not specified as they are part of the production process (e.g. folds required for a given binding style, such as folding of gathered signatures for saddle stitch binding, do not apply here). The types of folds that can be specified are:

- *F2-1*: No fold
- *F4-1*: Single fold
- *F6-1*: Zigzag fold
- *F6-3*: Alter fold
- *F6-4*: Tri fold
- *F6-7*: Z-fold
- *F8-2*: Parallel fold
- *F8-4*: Gate fold

Production oriented folds used with impositioning and trimming are not included as they do not influence the definition of the intended final product.

The fold illustrations shown in Table 17 of the PPM standard depict single product folds and do not necessarily represent that part's orientation when combined with other parts, e.g. as a fold out page bound into a book. To achieve the folded sheet orientation as depicted in the illustrations in Table 17, the **CIP4_Orientation** property requires a non-default setting. See section [8.5](#) of this document.

6.7. Colour Intent

Colour intent in PPM does not replace or supplement the use of conventional colour-management profiles present in the PDF file. Thus PPM is only used to specify coatings to be applied to identified finished pages.

Note 1: The spectral properties of the coating may require the colour management process to compensate for unwanted colour shifts.

Note 2: As stated in the PPM standard, "Information about printing colour printing conditions is out of scope of the PPM standard. This information shall be provided using the standard methods defined for PDF, e.g. output profiles." Accordingly, PPM cannot be used to specify the colour space in which PDF page content is defined. Everything related to color management can only be specified in the PDF data, not in the PPM.

6.8. Assembling Intent

The assembly intent is used to define a print product that is an assembly of related print product parts.

PPM defines three different types of assembling methods to specify how certain other part(s) may be inserted into or attached to a specified container:

- *BindIn* — each part is glued into the container,
- *BlowIn* — each part is loosely inserted into the container,
- *StickOn* — each part is glued onto the container, such as a label.

Assembling intent can be used to describe a complete mail piece consisting of a letter part that is folded and inserted into an envelope part. Assembly implied by binding need not be specified.

7. Encoding PPM

PPM is encoded within a PDF file using Document Part Metadata (DPM) as defined in PDF/VT and PDF 2.0 (clause 14.12). The definition of DPM in PDF 2.0 is less prescriptive than its use in PDF/VT.

7.1. DPart structure and DPM overview

The DPM associates metadata PPM to the various pages of the PDF document using the document part (DPart) node hierarchy. The DPart node hierarchy subdivides the PDF document into a set of (nested) page ranges for which DPM can be supplied.

The root of the DPart node hierarchy is referenced from the catalog dictionary of a PDF file as the dictionary value of the *DPartRoot* key using the following keys:

- **DPartRootNode** — an indirect reference to the DPart dictionary that is the root of the document part node hierarchy.
- **NodeNameList** — encodes an array of names for each level in the document part node hierarchy. The first entry is for the root node, the second entry for the direct children of the root node, etc.
- **RecordLevel** — optionally encodes the level in the document part hierarchy that corresponds to the pages for an individual recipient.

Each DPart dictionary represents a node in the document part node hierarchy and includes the following keys:

- **Parent** — an indirect reference to the dictionary that references this *DPart* dictionary. For the root node this is an indirect reference for the *DPartRoot* dictionary.
- **DPM** — a dictionary value with metadata that applies to the pages covered by this node.

For each intermediate DPart node, the DPart dictionary includes a **DParts** key with a value that is an array of arrays containing one or more indirect references to the DPart dictionaries of child DPart nodes. The DPM specified within an intermediate DPart node is associated with the sequence of pages referenced by its child leaf DPart node(s).

Note: The value of the **DParts** key is an array of arrays used to work around the array length limits of some PDF processors. For compatibility with ISO 16612-2 conforming PDF/VT-2 processors, all but the last entry in the **DParts** array are required to be an array containing exactly 8192 elements. The last entry in the **DParts** array is required to be an array containing at least one element. If more than 64 million children are needed an extra level of DPart nodes is therefore required to work around these array size limitations. This may occur, for example, when creating output from VDP jobs with a very large number of recipients.

For each leaf DPart node, the **DPart** dictionary provides additional data to explicitly identify its unique range of related pages using the following keys:

- **Start** — an indirect reference to the page object of the first page belonging to this leaf node.

- **End** — an indirect reference to the page object of the last page belonging to this leaf node. The **End** key is omitted for single page leaf nodes.

The document part node hierarchy must reference all of the pages in the PDF file and must reference them in the same order as does the PDF page tree. Each page object must have a **DPart** key whose value is an indirect reference to the DPart dictionary leaf node that references that page. The **DPart** and **Parent** keys therefore provide two ways to retrieve DPM related to a given page; either relative to the DPart hierarchy or the Page dictionary.

The interpretation of the key-value pairs in the DPM is determined by other specifications (e.g., the PPM standard). As required by PDF 2.0, all DPM metadata keys must use a registered second class name as a prefix so that the metadata relevant to a particular specification can be easily identified.

7.2. Encoding PPM within DPM

PPM is encoded as the values of metadata keys present in the DPM using the “CIP4_” prefix. The PPM that applies to a particular DPart is encoded as a dictionary under the **CIP4_Root** metadata key within the **DPM** dictionary of that DPart.

[Appendix A](#) contains an example of a complex print product with all the relevant PPM.

The core of PPM uses a subset of CIP4 XJDF 2.1 product intent definitions encoded in the DPM as the value of the **CIP4_Intent** metadata key. Each product intent definition is encoded as the value of a metadata key named after the XJDF element name of the product intent prefixed with the CIP4 second class name. For example, the XJDF product intent element named *BindingIntent* is encoded in PPM as **CIP4_BindingIntent**. The value of a PPM product intent metadata key is a PDF dictionary encoding of the XJDF attributes of the product intent. Each attribute is encoded using a key with the attribute name prefixed with the “CIP4_” second class name prefix. The attribute value is represented as a PDF value based on the XJDF schema type of the attribute value. Detailed encoding rules are described in section 6 of the PPM specification. The following rules apply:

- Basic type values (integer, float, boolean, string) are converted to their corresponding PDF primitive object values.
- Enumeration values, NMTOKEN and ID are encoded as a PDF name beginning with a slash ‘/’ character.
- List type values (anything encoded in XJDF as a whitespace separated array) are encoded as an array of values.
- Enumerations and NMTOKENS are encoded as an array of PDF names.
- Range type values are encoded as an array with two elements (begin value, end value).
- Date type values (date, dateTime) are encoded as a PDF date string (see section 7.9.4 of the PDF 2.0 specification).
- If none of the above rules apply then the value is encoded as a PDF text string (encoded as specified in the respective PDF specification, e.g. in “7.9.2.2 Text string type” of PDF 2.0).

Product intent definitions that contain other XJDF elements are encoded as the value of an additional metadata key with the element name prefixed with the CIP4 second class name prefix. The value of that key is a single dictionary if that element may occur at most once as a

child (a maximum cardinality of 1) or is an array of dictionaries otherwise. The content of each dictionary follows the same encoding rules for attributes and nested elements.

All dictionaries defined within PPM require the inclusion of a **Type** key. In general the value of the **Type** key matches the key name associated with the dictionary unless otherwise noted (e.g. the dictionary value of the **CIP4_Root** key has *CIP4_Root* as the value for its **Type** key).

Product intent definitions should include a **CIP4_ProductType** key that is a machine readable identifier of what the document part is intended to represent.

The product intent definitions and product type information together define the intended appearance of the print product represented by each DPart node; see [Print Product Appearance Rules](#).

PPM can also provide additional metadata for each individual document part:

- **CIP4_DescriptiveName** — a human readable description that includes more details of what is represented by the machine readable **CIP4_ProductType** identifier. For example, noting more specifically that the print product is an invoice when the **CIP4_ProductType** identifier is *Letter*.
- **CIP4_ExternalID** — a reference value to link the document part to a definition present on an external system. For example, the ID of the document template originally used in the creation of the document part.

Additional metadata for the document as a whole is encoded into the top level document part node (the DPart node referenced from **DPartRootNode**) as the value of the **CIP4_Metadata** key. The value of the **CIP4_Metadata** key is encoded as a PDF dictionary with the following required information:

- **CIP4_Conformance** — a list of names identifying the conformance levels the PPM adheres to. The value *CIP4_IntentBase_2.0* can be used if no other more restrictive conformance level applies.
- **CIP4_Creator** — a string value identifying the software application that generated the PPM.
- **CIP4_ModificationDate** — a PDF date string indicating the date and time the metadata was modified (if any). The value of this key can be used to detect when a software application that does not understand PPM has modified the PDF and may have invalidated the metadata. The typical best practice in such a case is for the receiver of the file to contact the document originator for clarification.

PPM permits inclusion of the following optional information:

- **CIP4_Accounting** — a CIP4_Contact dictionary containing contact information for billing purposes.
- **CIP4_Administrator** — a CIP4_Contact dictionary containing contact information for the individual(s) responsible for the PDF document data.
- **CIP4_Author** -a CIP4_Contact dictionary containing contact information for the author of the PDF document.
- **CIP4_Sender** — a CIP4_Contact dictionary containing contact information for the sender or originator of the PDF document.
- **CIP4_JobID** — a reference to the job as used in the external system.

- **CIP4_ProjectID** — a reference to the project as used in an external system.

PPM also provides a mechanism to identify the intended recipient of the print product (or part of the print product). The DPart nodes for which recipient information may be specified depends on the **RecordLevel** key in the **DPartRoot** dictionary of the PDF. If the **RecordLevel** value is zero then only a single recipient can be specified. If the **RecordLevel** value is greater than zero then each DPart node at the indicated level may specify a separate recipient. The **CIP4_Recipient** key specifies the intended recipient using a dictionary where:

- **CIP4_ExternalID** — provides an external unique reference to the recipient.
- **CIP4_Contact** — provides the contact information of the recipient

PPM encodes contact information as a **CIP4_Contact** dictionary, see [Providing Contact Information](#) section for more details.

7.3. Providing Contact Information

The **CIP4_Contact** dictionary provides contact information for various purposes via these keys:

- **CIP4_ContactTypes** — an array of names that identify the purpose of the contact. Each name identifies a specific purpose such as *Recipient*, *Editor*, *Owner*, etc. The full list of purposes can be found in Table 25 of the PPM specification.
- **CIP4_DescriptiveName** — a human readable string representing the role of the contact. This should be used in the case that a standard purpose listed in the PPM specification is not applicable, or additional information is required.
- **CIP4_ComChannel** — an array of dictionaries representing information on the method that may be used to communicate with the contact.
- **CIP4_Person** — a dictionary representing information regarding the person (such as name) to contact.
- **CIP4_Company** — a dictionary representing information regarding the contact's organization.
- **CIP4_Address** — a dictionary representing the contact's address.

The following keys are available in a **CIP4_ComChannel** dictionary:

- **CIP4_ChannelType** — a name that identifies the method of communication such as *Email*, *Fax*, *Phone*, *Mobile* and *WWW* (for a web URL).
- **CIP4_DescriptiveName** — a human readable string representing the method and/or location if the standard values do not apply or need additional clarification.
- **CIP4_Locator** — a string that provides the locator of the channel such as the phone number or email address to use.
- **CIP4_ChannelUsage** — a name that defines the purpose of the channel, such as *Business*, *Private*, *DayTime*, *NightTime*, or *Weekend*.

The following keys can be used in a **CIP4_Person** dictionary:

- **CIP4_FirstName** — a string representing the person's first name.
- **CIP4_FamilyName** — a string representing the person's family name.
- **CIP4_AdditionalNames** — a string representing the person's middle names if any.
- **CIP4_FullName** — a string representing the person's full name.
- **CIP4_DescriptiveName** — a human readable string representing additional information about the person that is not already covered by the other keys.
- **CIP4_JobTitle** — a string representing the person's organizational role.
- **CIP4_NamePrefix** — a string representing a prefix to the person's name such as Mr., Ms., Dr., etc.
- **CIP4_NameSuffix** — a string representing a suffix to the person's name such as Jr., Sr., etc.

Where possible the above information about a person should be provided in the most specific keys that are applicable. However, sometimes this breakdown of information is not available then at least the **CIP4_FullName** should be provided.

The following keys can be used in a **CIP4_Company** dictionary:

- **CIP4_OrganizationName** — a string representing the name of the organization to which a contact belongs.
- **CIP4_DescriptiveName** — a human readable string representing the contact's organization.
- **CIP4_OrganizationalUnit** — an array of strings representing the name(s) of the organizational unit to which the contact belongs.

The following keys can be used in a **CIP4_Address** dictionary:

- **CIP4_AddressUsage** — a name identifying the intended usage of the address. Predefined usages are *Business* and *Residential*.
- **CIP4_AddressLines** — an array of strings representing the complete formatted address, with one string per line.
- **CIP4_City** — a string representing the name of the city in which the address is located.
- **CIP4_CivicNumber** — a string representing the civic number of the address.
- **CIP4_Country** — a string representing the country in which the address is located.
- **CIP4_CountryCode** — a string representing the ISO 3166-1 alpha-2 code of the country in which the address is located.
- **CIP4_PostalCode** — a string representing the postal code of the address.
- **CIP4_PostBox** — a string representing the postal box identifier of the address.
- **CIP4_Region** — a string representing the region (state or province) in which the address is located.
- **CIP4_Street** — a string representing the complete street address which is a combination of the values of **CIP4_StreetName** and **CIP4_CivicNumber** when they are not available separately.
- **CIP4_StreetName** — a string representing the street name for the address.

The address for a contact is not always available in its individual components as described above. The **CIP4_AddressLines** key should always be filled to ensure that the whole address is available. Where possible the more detailed fields should be included to simplify post-processing

tasks such as postal sorting. Many postal sorting tools have the capability to use a pre-formatted address but this process is generally less reliable.

8. Print Product Appearance Rules

8.1. Concepts

A print product is defined using PPM as an arrangement of one or more related parts. An example of a simple print product would be a book with a two-sided cover and body parts.

Other print products, however, have more complex features requiring greater definition. Such print products can utilize a larger variety of PPM intent definitions, such as a case bound book with a fold-out page and sections requiring different paper types.

Most print products can be represented using some combination of PPM intent definitions positioned within a DPart hierarchy, with one or more leaf DPart nodes mapping logical pages to finished pages. The purpose of this structure is to express the relationship of the parts of the finished print product.

A given DPart node along with any child DPart nodes typically defines a complete or part of a print product.

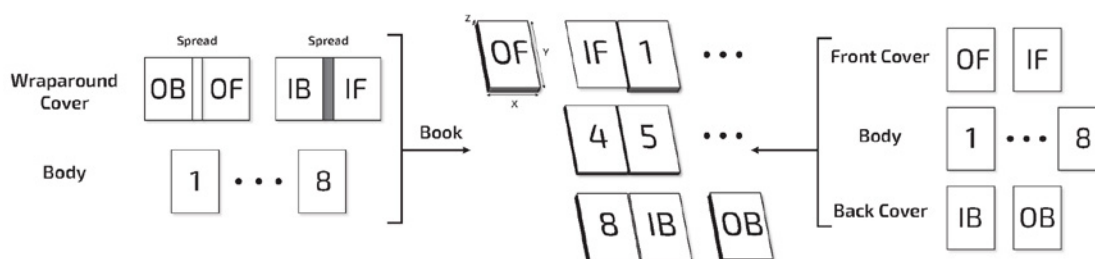
The sequence of *logical pages* referenced by a leaf DPart node defines the content to be imaged onto the sides of a sequence of one or more finished pages of a part.

A finished page:

- is produced on a single media,
- has a front and back side,
- each side may either be empty or shows the content of all or part of a PDF page,
- may be bound to other finished pages,
- may be folded,
- may have holes.

Note: A single physical sheet may contain multiple finished pages.

Example: A wraparound cover consists of two finished pages. One of those finished pages from the front cover and the other from the back cover:



The purpose of **CIP4_LayoutIntent** is to define how a sequence of one or more logical pages is mapped to specific sides of a sequence of one or more finished pages.

For example, where the **CIP4_Sides** key has a value of *OneSided* or *OneSidedBack* specified for a logical page, this implies a single finished page is to be imaged with the content from that logical page but for one side only. Accordingly, a sequence of four such logical pages implies a sequence of four one-sided finished pages.

If the **CIP4_Sides** key has a value of *TwoSidedHeadToHead* or *TwoSidedHeadToFoot* a logical page is expected for each side of the implied two-sided finished page. If an expected logical page is missing then the associated finished page side is presumed blank.

If two consecutive logical pages within a single DPart have different page sizes (explicit or implied **TrimBox**) then each of them are expected to be imaged onto a side of separate finished pages.

Example: A two logical page DPart node of a Letter print product definition that includes a *LayoutIntent* with **CIP4_Sides** set to the value of *TwoSidedHeadToHead* where each page has a different orientation such as:

- Page 1 A4 Portrait
- Page 2 A4 Landscape

In this case the two incompatible logical page sizes implies that the print product must have two finished pages where:

- Finished page 1 front gets logical page 1,
- Finished page 1 back gets an implied blank logical page,
- Finished page 2 front gets logical page 2,
- Finished page 2 back gets an implied blank logical page.

Any product intent that implies an operation be applied to a DPart node only applies to the part defined by that DPart node as a whole. In general, this rule implies that the finished pages of that part are first gathered together and then the specified operation is performed on the gathered pages as a whole.

Example: A folding intent on the root DPart node implies that all the finished pages are first gathered together and then folded as a whole. If each finished page should first be folded and then gathered together then a DPart node for each finished page would be needed with a folding intent for each DPart node.

8.2. Product Type

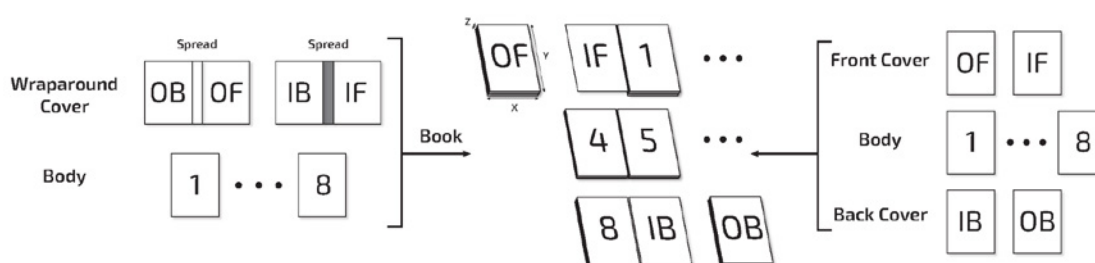
The **CIP4_ProductType** key specifies the product type of a DPart node. The product type defines the purpose of the part defined by that DPart node. Although optional, providing such metadata is strongly recommended as the product type may help in making production decisions.

Some of the defined product types also imply specific handling conventions:

- A *Body* part is implicitly inserted into a *Cover* part. Thus the finished pages of the *Body* part are intended to be inserted between the finished pages of the *Cover* part.
- A *FrontCover* part defines a finished page that comes before the finished pages of a corresponding *Body* part. If a *BackCover* is defined but the *FrontCover* is not explicitly defined then an empty finished page is presumed for the *FrontCover*.

- A *BackCover* part defines a finished page that comes after the finished pages of a corresponding *Body* part. If a *FrontCover* is defined but the *BackCover* is not explicitly defined then an empty finished page is presumed for the *BackCover*.
- A *WrapAroundCover* is a single folded sheet of media into which the finished pages of the corresponding *Body* part are inserted. If the DPart node also includes a **CIP4_LayoutIntent** resource with a **CIP4_SpreadType** key with a value of *Spread* then the logical page content for the front side specifies the outside of the cover as a whole and the back side specifies the inside of the cover. Otherwise the *WrapAroundCover* implies the same behavior as a *Cover* product type.
- A *Book*, *Booklet* or *Notebook* part implies that a binding is expected. The **CIP4_BindingIntent** resource is used to specify the exact details of the type of binding.
- A *Jacket* part implies a *WrapAroundCover* and the correct folding is implied.
- An *Envelope* part is typically used as the container for a finished mail piece.

Example: Book with/without wrap-around cover:

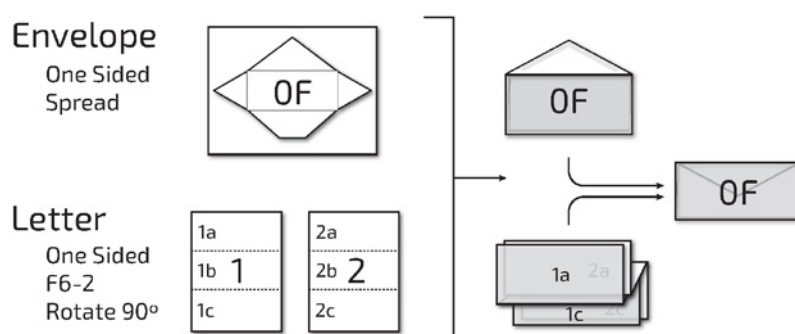


A part with a product type that implies a limit on the number of finished pages (e.g. *FrontCover*, *BackCover*, *WrapAroundCover*) that references logical pages that imply more finished pages than the part's limit will lead to an undefined result.

For a part with **CIP4_ProductType** with a value of *Envelope* the **CIP4_LayoutIntent** determines how the referenced PDF page content is used:

If **CIP4_SpreadType** has a value of *SinglePage* (i.e. folding not required) then the size of the PDF page (i.e. its **TrimBox**) determines the size of the finished envelope. Any PDF page content is printed on the front and back of the finished envelope (depending on the **CIP4_Sides** setting) and the inside is blank.

Example: Envelope with *Spread*:



If **CIP4_SpreadType** has a value of *Spread* then it is presumed that a single sheet will be cut/folded into the envelope form. The PDF page content intended for the front of the sheet therefore specifies the content for the outside of the envelope, and the PDF content intended for the back of the sheet specifies the content on the inside of the envelope (similar to a wraparound cover). The size of the envelope must be specified using **CIP4_FinishedDimensions**. The center of the PDF page content for the part will be imaged such that it ends up at the center of the finished envelope.

Any special features of the envelope such as the exact envelope contour (size, location, and form of the flaps) and window location cannot be specified in PPM, and must be specified for instance using the Processing Steps specification (ISO ref 19593-1). Typically the opening of an envelope can be presumed to be at the top. In practice the location of the opening is important, for example, for the physical insertion process that places printed matter inside the envelope.

8.3. Layout Intent

The **CIP4_LayoutIntent** key specifies the layout intent of a print product part using a set of keys.

The **CIP4_FinishedDimensions** key specifies the finished dimensions (width, height, and thickness) of the finished part. The finished dimensions specify the intended size of the part after any folding, trimming and binding operations have been applied. The thickness of the specified finished dimensions (Z dimension) is set to 0 if the exact thickness is unknown.

If **CIP4_FinishedDimensions** is not explicitly specified then the finished dimensions for a part is the aggregate of the finished dimensions of the part's finished pages taking into account the relative alignment of those finished pages. If the part's **CIP4_FinishedDimensions** are smaller than the calculated finished dimensions then trimming of the part to the specified size is implied.

The definition of layout intent requires a specification of the finished dimensions of each finished page of a part in order to determine the intended behavior for a part. If the finished dimensions for a finished page are not explicitly specified then the finished dimensions are presumed equal to the dimensions specified by the **TrimBox** of each of the PDF pages intended to be imaged onto that finished page. As the **TrimBox** only specifies width and height, the thickness (Z dimension) of the finished dimensions is left unspecified, and is therefore set to 0.

If the PDF page's **TrimBox** and the specified finished dimensions are not the same then the **TrimBox** defines the dimensions of the finished page prior to folding. The specified finished dimensions for a finished page should match the finished page size after folding. For spreads, it is necessary to explicitly specify finished dimensions using **CIP4_FinishedDimensions**.

The **CIP4_Sides** key in the layout intent specifies which sides of a finished page will have content imaged onto it. The following values are available:

- *OneSided* — implies that only the front side of the finished page has content.
- *OneSidedBack* — implies that only the back side of the finished page has content.
- *TwoSidedHeadToHead* — implies that both sides of the finished page have content.
- *TwoSidedHeadToFoot* — implies that both sides of the finished page have content. The content for the back side of the finished page is rotated by 180 degrees before being imaged. The binding edge is typically expected to be either the top or bottom. However

there are less common uses for combining top/bottom with a left or right binding edge, for example in dual language text books.

The **CIP4_SpreadType** key in the layout intent specifies whether the content is a *Spread* or *SinglePage*. A layout intent should only specify a **CIP4_SpreadType** value of *Spread* for logical content that will be used for a cover, an envelope, a folded page or a center page of a book. The results are undefined in all other cases.

Note 1: For wraparound covers and center spreads, the content of each logical page defines the content used for two finished pages.

Note 2: The actual size of the print media used in production may be larger than the specified finished dimension in order to accommodate production specific needs such as sheet marks, quality control marks and printing device limitations.

8.4. Media Intent

The **CIP4_MediaIntent** key specifies the Media Intent which is used to provide the characteristics of the media to be used for a print product part using keys in a PDF dictionary. A complete list of characteristics that can be specified can be found in Table 21 of the PPM Standard.

Since a **CIP4_MediaIntent** definition does not imply an operation to be performed, the scope of the specified **CIP4_MediaIntent** definition covers all finished pages contained within the print product part. This includes all child part definitions unless an overriding **CIP4_MediaIntent** definition is specified for a child part.

The PPM standard does not specify partial inheritance of a product intent definition, therefore each product intent definition fully defines all the characteristics. Accordingly, when specifying the same product intent resource on a part and a child part, only the characteristics specified on the child part apply; any characteristics specified on the parent part are ignored.

Example: A job with a root DPart node specifying the default MediaIntent with an exception for a particular finished page.

- Root DPart node for all pages: */CIP4_MediaColor (Blue) /CIP4_Weight 80*
- Leaf DPart node for page 1: */CIP4_Weight 100*

This implies that all but the first page will be Blue while the first page has an undefined (presumably white) color, as the **CIP4_MediaColor** from the root DPart is not inherited.

Note 1: **CIP4_MediaIntent** does not include a key specifying the size of the media to be used as the size is derived from the PDF page size and the associated **CIP4_LayoutIntent**.

Note 2: **CIP4_MediaIntent** does not include a key to specify pre-punched media as this is defined with the use of **CIP4_HoleMaking**.

8.5. Folding Intent

The **CIP4_FoldingIntent** key specifies the Folding Intent which is used to define the folding characteristics of a *print product part* using keys in a PDF dictionary.

Folding is applied to the part as a whole. This implies that the finished pages of the part are first gathered/collected and then folded rather than each finished page being folded and then gathered.

The **CIP4_FoldCatalog** key defines the folding pattern as a sequence of folds. The key's value is a name from the folding catalog as defined in the XJDF specification. Only a subset of folds are included, see section 7.6.6 of the PPM standard.

The folding pattern uses the bottom left corner of the finished page as the reference location called the *sheet lay*. Each fold catalog entry in XJDF specifies how much to advance the page in the feeding direction specified by the sheet lay (e.g. for bottom left the feeding direction is left and for upper left the feeding direction is up) and if the fold direction is up, the advanced portion is folded over the remaining portion. Otherwise the advanced portion is folded under the remaining portion.

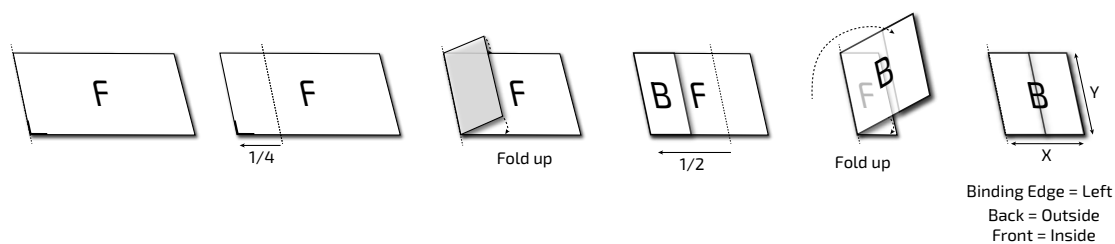
The F6-3, F6-4, F7-7, F8-2, F8-4 pictures in Table 17 of the PPM standard are shown in a manner to more clearly illustrate the folded print product, and do not appear with the default orientation of *Rotate0*.

The **CIP4_Orientation** key specifies the reference location and direction of the folds as defined in the fold catalog. This enables the specification of variants of the same basic fold patterns.

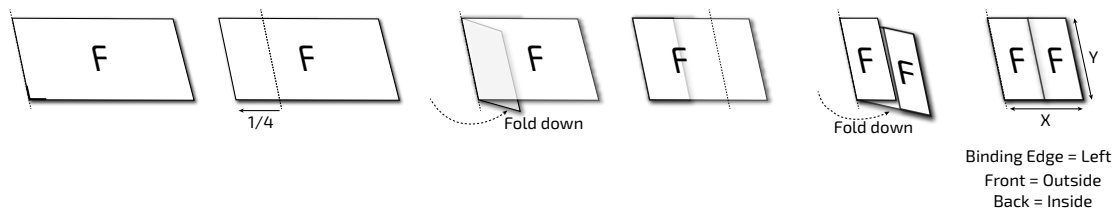
CIP4_Orientation value	Reference location	Folding location	Feeding direction
<i>Rotate0</i>	bottom left	default	left
<i>Rotate90</i>	upper left	default	up
<i>Rotate180</i>	upper right	default	right
<i>Rotate270</i>	bottom right	default	bottom
<i>Flip0</i>	bottom left	inverted	left
<i>Flip90</i>	upper left	inverted	up
<i>Flip180</i>	upper right	inverted	right
<i>Flip270</i>	bottom right	inverted	down

Changing the reference location and folding direction does not alter the physical orientation of the finished page, but only influences the location, orientation, and folding direction of the folds specified in the fold catalog.

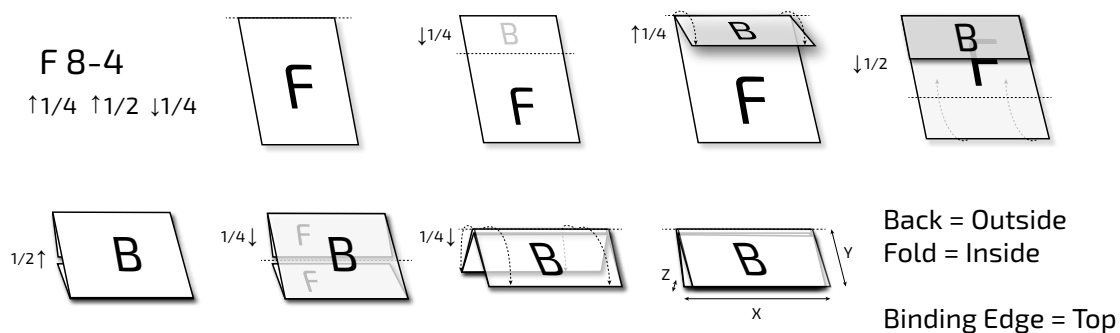
Example: F6-3 with *Rotate0*:



Example: F6-3 with Flip0:



Example: F8-4 with Rotate90:



If both binding and folding intent applies then the binding edge refers to the corresponding edge of the folded part after folding. The results are undefined for certain folding types when saddle or side stitching is requested due to the binding edge being physically connected to another finished page to form a single sheet where the folding would require a “flap” to be somehow attached. Similarly, the results are undefined when requesting folding on an axis perpendicular to the binding edge (e.g. binding edge left and using *Rotate90*, *Rotate270*, *Flip90*, *Flip270*).

8.6. Hole making Intent

The **CIP4_HoleMakingIntent** key specifies any holes that should be made in a print product part using keys in a PDF dictionary. A complete list of characteristics that can be specified can be found in Table 19 of the PPM Standard.

The **CIP4_HoleReferenceEdge** key specifies the edge of the print product where the holes are placed. The **CIP4_Pattern** key specifies a pattern id from the Hole Pattern Catalog in the XJDF specification. The pattern defines the number, shape and distribution of the holes along the reference edge. The center of the pattern of the specified physical hole locations is aligned with the center of the reference edge of the print product part. The specified axis offset of the pattern indicates the distance from the reference edge. The pattern geometry specifies the spacing between the center of the holes along the reference edge.

Logically, hole making is an operation applied to the physical print product part as a whole. Since only the desired end result can be specified by product intent, the actual production order of a hole making process is not specified. Using pre-drilled media that meets the intended hole requirements for the part is therefore considered equivalent to drilling holes after printing. If

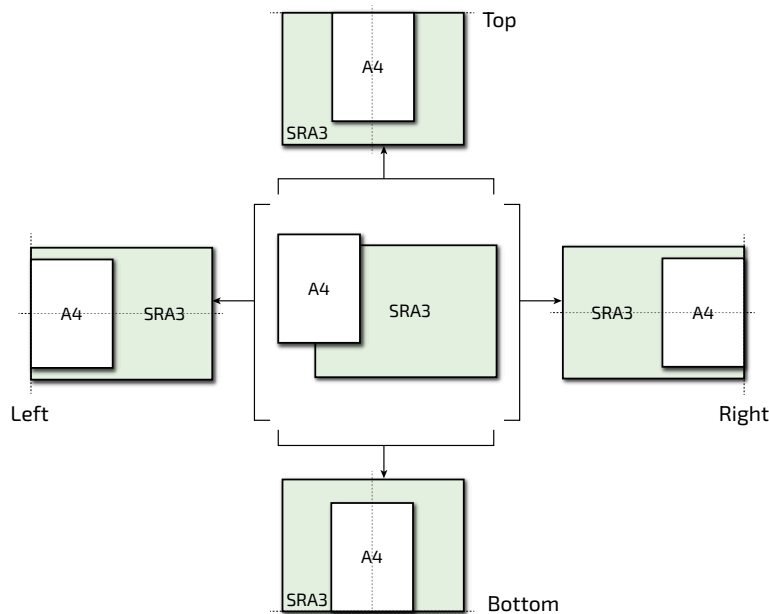
hole making is specified on a child part as well as its parent part, both the holes specified on the parent part and the child part will apply to the child part.

8.7. Binding Intent

The **CIP4_BindingIntent** key specifies the binding for a print product part using keys in a PDF dictionary. A complete list of characteristics that can be specified can be found in Table 12 of the PPM Standard.

Binding is applied to the part as a whole. This implies that the finished pages of the part are first gathered/collected and then bound.

The **CIP4_BindingSide** key identifies the binding edge of the bound print product part. The binding edge determines how child parts or finished pages are aligned to form a stack with the binding edges. If the finished pages or parts are of different sizes along the binding edge then the parts are aligned along the center of each part:

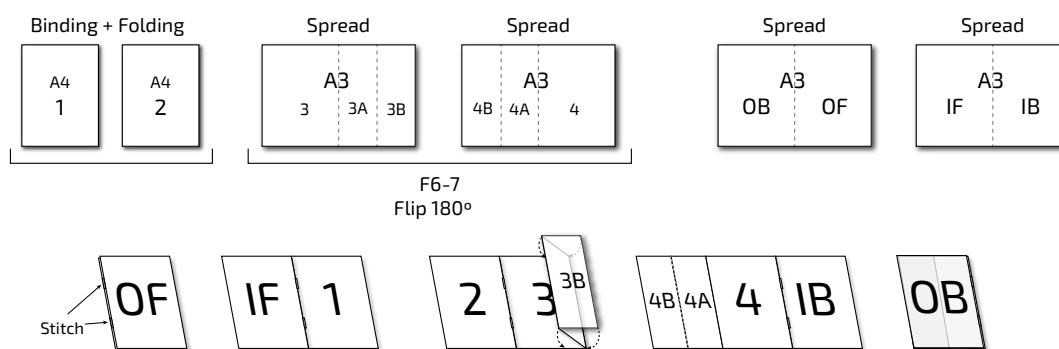


If no binding intent is specified then the child parts are aligned along the left edge. Alignment is always based on the child part after folding is applied.

The **CIP4_BindingType** key defines the binding method for the print product part. If the binding method is *SaddleStitch* then the **CIP4_SaddleStitching** key can be used to provide additional detail of the required stitching using keys in a PDF dictionary. If the binding method is *SideStitch* then the **CIP4_SideStitching** key can be used to provide additional details of the required stitching if necessary. In each case the **CIP4_StitchNumber** key defines the number of required stitches. In the absence of a **CIP4_StitchNumber** key any number of stitches are permitted. For the binding method *CornerStitch* the number of stitches cannot be explicitly specified.

A part for which a binding method of *SaddleStitch* or *SideStitch* is specified requires an even number of finished pages. If the number of finished pages is odd then an additional blank finished page is implicitly exists at the end of the part.

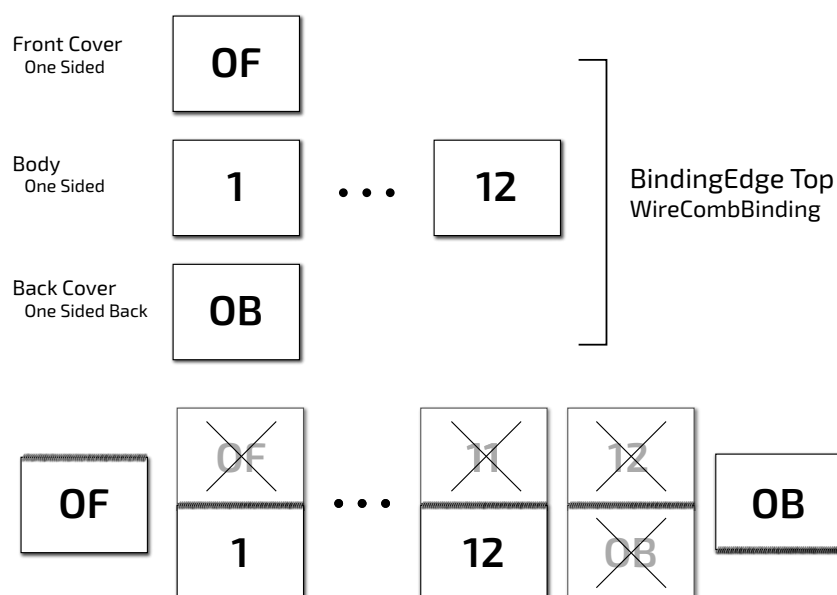
Example: *SaddleStitched* booklet with fold out:



Neither folding nor holemaking implied by the binding method should be specified.

For back covers with no inside back content the **CIP4_Sides** should specify *OneSidedBack* to ensure that the single logical page content is imaged on the outside.

Example: Calendar:



8.8. Assembling Intent

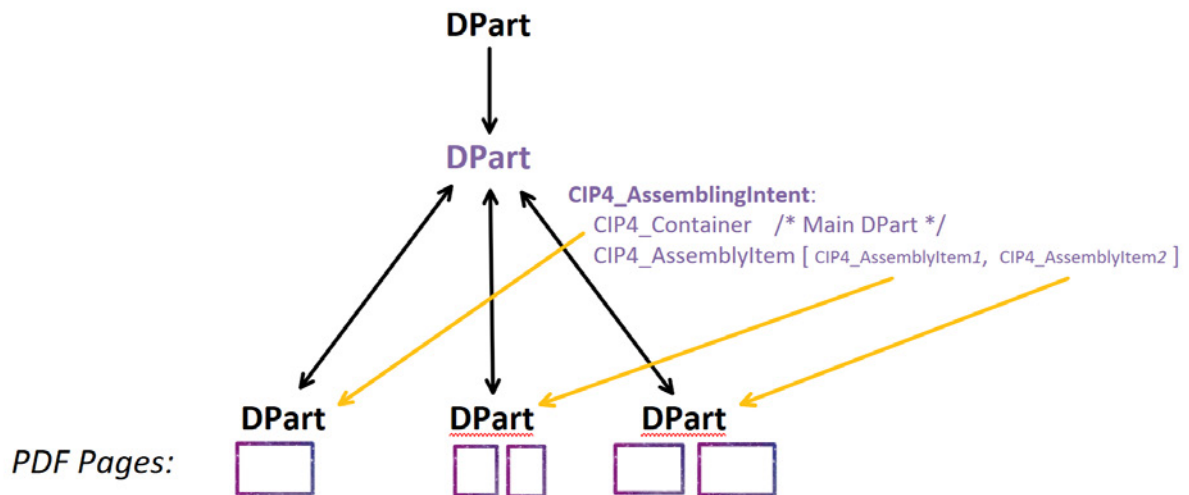
The **CIP4_AssemblingIntent** key specifies an assembly of product parts using keys in its PDF dictionary. The assembly intent specifies that in the assembled print product the child parts are placed inside a container without changing the orientation. The resulting part is the specified container.

If the finished dimensions of the part to be inserted into the container are equal to or exceed the finished dimensions of the container it is inserted into (i.e. doesn't fit) then the results are undefined.

Note 1: Some clearance on the inside will generally be necessary to allow the insertion process to succeed.

The **CIP4_Container** key specifies an indirect reference to the DPart dictionary that defines the container for the other referenced print product parts.

The **CIP4_AssemblyItem** key specifies an array of dictionaries, each of which define a print product part to be inserted into the container in the order specified.



The **CIP4_BindIn** key specifies an array of dictionaries, each of which refers to a print product part that is glued into the container in the order specified.

The **CIP4_BlowIn** key specifies an array of dictionaries, each of which refers to a print product part that is loosely inserted into the container in the order specified.

The **CIP4_StickOn** key specifies an array of dictionaries, each of which refers to a print product part that is glued onto the container in the order specified. The exact location of where a print product is stuck onto the container cannot be specified via the current PPM standard.

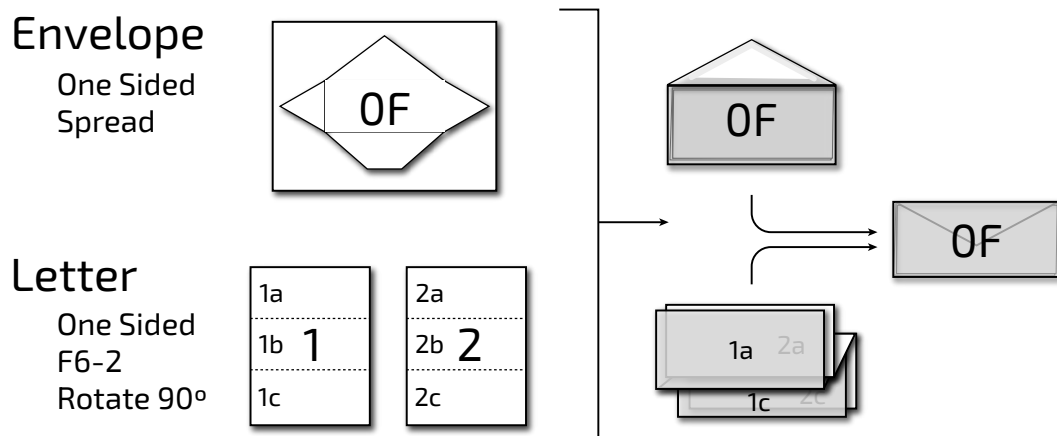
Each dictionary of the arrays contains a **CIP4_Child** key which is an indirect reference to the DPart dictionary that defines the print product part to be used in the assembly.

The DPart dictionary on which assembly intent is defined shall not reference one of its parent DPart dictionaries.

Note 2: The direct descendent DPart dictionaries of the DPart on which **CIP4_AssemblingIntent** is specified are typically used for defining the print product part for the container and any print product parts that are inserted or attached to the container.

A print product part with a **CIP4_ProductType** with value of *Envelope* can be referenced as a container and is used for describing the appearance of the envelope itself. The assembly intent may use an envelope as a container in its specification of a finished mail piece.

Example: complete mail piece:



8.9. Extensions

The PPM standard permits use of private extensions as allowed by the XJDF specification. The use of private extensions, however, will generally limit interoperability among products and solutions beyond a possible subset of cooperating vendors. As per Annex E of the PDF specification, private extensions require the use of a registered second class name prefix (see the PDF Names List at <https://github.com/adobe/pdf-names-list>).

Before using a private extension in PPM, vendors should check whether the XJDF specification already has a method of specifying the desired attributes of the print product. In such cases the encoding rules of section 6.2 to map XJDF to PDF must be applied and should use the second class name prefix "CIP4_". In situations where the XJDF specification does not have a way to express an equivalent intent property, it is recommended that a new intent be defined and proposed as an addition to the XJDF specification using CIP4's registration process.

Appendix A: Romantic Koosbania example

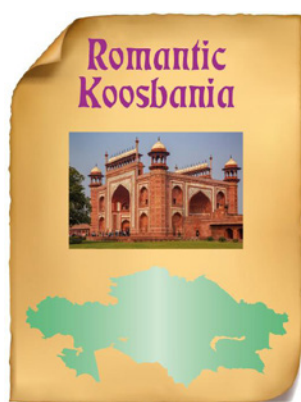
This appendix contains a description of final print products using PPM. The PDF snippets containing PPM in this appendix are PDF objects allowing pages with the same requirements to share the DPM information. The example consists of a complex bound book and an accompanying poster.

Final print products



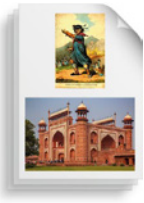


Complete book



Poster



Page Content

Content	Description
	<p>Book cover: A single page representing the imposed back, spine, and front of the book's cover as a spread to be printed on book cover cloth material.</p>
	<p>Text pages: Multiple pages representing text pages of this book, printed on white, coated text paper, monochrome only.</p>
	<p>Illustration pages: Multiple pages within the book containing high quality imagery printed on coated, heavy, glossy paper using process CMYK.</p>
	<p>Map pages: Multiple pages within the book containing maps on fold-in pages simplex printed on coated, heavy, glossy paper using process CMYK.</p>
	<p>Poster page: A single page representing the content to be printed as a poster on coated, very heavy, glossy paper.</p>

PPM

Complete book

```
1100 obj <<
  /CIP4_Root
  <<
    /Type /CIP4_Root
    /CIP4_DescriptiveName (Romantic Koosbania)
    /CIP4_Intent <<
      /Type /CIP4_Intent
      /CIP4_ProductType /Book
    >>
  >>
>> endobj
```

Book cover

```
1101 obj <<
  /CIP4_Root <<
    /Type /CIP4_Root
    /CIP4_DescriptiveName (book cover)
    /CIP4_Intent <<
      /Type /CIP4_Intent
      /CIP4_LayoutIntent 1283 0 R
      /CIP4_MediaIntent 1284 0 R
      /CIP4_ProductType /WrapAroundCover
    >>
  >>
>> endobj

1283 0 obj <<
  /Type /CIP4_LayoutIntent
  /CIP4_Sides /OneSided
  /CIP4_SpreadType /Spread
  /CIP4_FinishedDimensions [ 595 842 10 ]
>> endobj

1284 0 obj <<
  /Type /CIP4_MediaIntent
  /CIP4_MediaQuality (RuggedCloth)
  /CIP4_MediaTypeDetails /Cloth
>> endobj
```

Text pages

```
1101 obj <<
  /CIP4_Root <<
    /Type /CIP4_Root
    /CIP4_DescriptiveName (text pages)
    /CIP4_Intent <<
      /Type /CIP4_Intent
      /CIP4_LayoutIntent 1285 0 R
      /CIP4_MediaIntent 1286 0 R
      /CIP4_ProductType /Body
    >>
  >>
>> endobj

1285 obj <<
  /Type /CIP4_LayoutIntent
  /CIP4_Sides /TwoSidedHeadToHead
>> endobj

1286 obj <<
  /Type /CIP4_MediaIntent
  /CIP4_Coating /Matte
  /CIP4_ISOPaperSubstrate /PS2
  /CIP4_MediaQuality (Text100)
  /CIP4_Weight 100
>> endobj
```

Illustration pages

```
1103 obj <<
  /CIP4_Root
  <<
    /Type /CIP4_Root
    /CIP4_DescriptiveName (color text)
    /CIP4_Intent <<
      /Type /CIP4_Intent
      /CIP4_LayoutIntent 1287 0 R
      /CIP4_MediaIntent 1288 0 R
      /CIP4_ProductType /Body
    >>
  >>
>> endobj

1287 obj <<
  /Type /CIP4_LayoutIntent
  /CIP4_Sides /TwoSidedHeadToHead
>> endobj

1288 obj <<
  /Type /CIP4_MediaIntent
  /CIP4_Coating /Gloss
  /CIP4_ISOPaperSubstrate /PS1
  /CIP4_MediaQuality (Text120)
  /CIP4_Weight 120
>> endobj
```

Map pages

```
1104 obj <<
  /CIP4_Root <<
    /Type /CIP4_Root
    /CIP4_DescriptiveName (foldout map)
    /CIP4_Intent <<
      /Type /CIP4_Intent
      /CIP4_ColorIntent 1292 0 R
      /CIP4_FoldingIntent 1291 0 R
      /CIP4_LayoutIntent 1289 0 R
      /CIP4_MediaIntent 1290 0 R
      /CIP4_ProductType /Map
    >>
  >>
>> endobj

1292 obj <<
  /Type /CIP4_ColorIntent
  /CIP4_Coatings [/Varnish]
>> endobj

1291 obj <<
  /Type /CIP4_FoldingIntent
  /CIP4_FoldCatalog /F6-7
>> endobj

1289 obj <<
  /Type /CIP4_LayoutIntent
  /CIP4_Sides /OneSided
  /CIP4_SpreadType /Spread
  /CIP4_FinishedDimensions [ 595 842 0 ]
>> endobj

1290 obj <<
  /Type /CIP4_MediaIntent
  /CIP4_Coating /Gloss
  /CIP4_ISOPaperSubstrate /PS1
  /CIP4_MediaQuality (Special150)
  /CIP4_Weight 150
>> endobj
```

Poster page

```
1105 obj <<
  /CIP4_Root
  <<
    /CIP4_DescriptiveName(additional poster)
    /CIP4_Intent <<
      /Type /CIP4_Intent
      /CIP4_ColorIntent 846 0 R
      /CIP4_LayoutIntent 844 0 R
      /CIP4_MediaIntent 845 0 R
      /CIP4_ProductType /Poster
    >>
  /Type /CIP4_Root
>>
>> endobj

846 0 obj <<
  /Type /CIP4_ColorIntent
  /CIP4_Coatings [/Varnish]
>> endobj

844 0 obj <<
  /Type /CIP4_LayoutIntent
  /CIP4_Sides /OneSided
>> endobj

845 0 obj <<
  /Type /CIP4_MediaIntent
  /CIP4_Coating /Gloss
  /CIP4_ISOPaperSubstrate /PS1
  /CIP4_MediaQuality (Special140)
  /CIP4_Weight 150
>> endobj
```