PODi

PPML 2.2

November 2006

Personalized Print Markup Language

Functional Specification

PODi: the Digital Printing Initiative

1240 Jefferson Road, Rochester, New York 14623, USA

Tel: (585) 239-6014

Internet: **http://www.podi.org**

PODi the Digital Printing Initiative

Approval of a PODi standard requires acceptance by the members of PODi.

PODi is a not for profit industry consortium formed in 1996. Its charter is to foster the growth of the digital printing industry through market and standards development activities. PODi constantly monitors market and technology trends in the industry, and shares information through seminars, independent research, white papers, articles, and the web. PODi promotes interoperability through the PPML suite of open, XML based standards, test suites and certification.

PODi welcomes feedback on this specification, and offers the following services to support widespread adoption of the specification:

Specification Updates

> The PPML Functional specification is distributed free of charge. Developers who are implementing the PPML standard are invited to subscribe to free PPML updates and the technical note service.

Developer Support web site

> Software and hardware developers interested in supporting PPML are invited to register for the PPML Developers discussion group.

To participate in the PPML initiative, send an email to ppmlinfo@podi.org.

Contents

Foreword

The Personalized Print Markup Language (PPML) standard was introduced in May 2000 by PODi to foster market growth in high-volume, full-color variable data printing. Key concepts include the ability to leverage existing standards and to ensure interoperability between and among hardware and software vendors. PPML promotes the development of highly efficient print streams through object-level addressability and reusability for page components in a print workflow. PPML encourages the use of reusable objects, which optimizes file sizes for print jobs with high graphical content. PPML-based workflows are intended to be described using existing job ticketing formats such as the Job Definition Format (JDF) sponsored by CIP4 and are not a part of the PPML Functional specification. However, PPML does provide meta information that can guide such workflows.

PPML is an open industry standard that uses an XML grammar to define how to compose digital assets into objects, pages, documents, and sets. A wide range of XML software tools is available today that can convert a stream of data into a ready-to-print stream of PPML documents. Some XML tools are free, including some that are open source.

PPML allows digital assets in multiple content formats to be included in the PPML file or retrieved from local or remote storage during processing. Those digital assets are clipped, transformed, and combined into objects. Those objects are organized into a hierarchy of pages, documents and sets, which enables advanced processing such as impositioning and finishing.

PPML also allows objects, pages, documents, and sets to be classified so that each class may be assigned different properties in a job ticket. The ability to assign classes to objects allows for more granular assignment of job processing parameters. Class-based selection of properties also allows job tickets to be smaller and reused as a whole.

PPML optimizes digital print workflows by allowing objects that are placed on many pages to be defined once, marked as reusable, and used multiple times.

NOTE Some of the elements of this standard may be the subject of patent rights. PODi is not responsible for identifying any or all such patent rights.

PODi does not guarantee the suitability of any of PPML or any of the conformance subsets for any specific purpose.

Send suggestions for improving this standard to PODi, 1240 Jefferson Road, Rochester, NY 14623, USA; e-mail: ppmlinfo@podi.org.

Introduction

The Personalized Print Markup Language (PPML) specification defines an XML grammar for specifying graphical page content for both monochrome and full color variable data jobs. The PPML format describes how to combine existing digital assets using clipping and transformations into pages, documents, and sets. PPML provides only meta information that can be used to guide PPML-based workflows. PODi recommends the use of JDF to describe such workflows. For information on the use of PPML with JDF, see the PODi *Digital Print Ticket (DPT) Specification*.

This specification defines the syntax and semantics of PPML datasets using a formalized imaging model. This specification also defines the conformance requirements for Producers and Consumers of PPML datasets. The PPML Functional specification allows conformance subsets to be defined to enhance interoperability. A registry of conformance subsets may be found at the PODi website.

PPML based impositioning was an integral part of previous versions of this specification. As part of the effort to separate content from workflow, the PPML impositioning instructions have been moved into a separate document, the *PPML Impositioning Specification*, without affecting the current PPML syntax. PODi now recommends using JDF based impositioning instead of PPML based impositioning, which may be deprecated in the future.

The purpose of PPML is to:

- optimize print speed
- organize page content for flexible processing and finishing
- allow flexible access to digital assets, which may be stored internally, locally, or remotely
- leverage existing standards, infrastructures and digital assets
- enable interoperability

PPML Producers might be document composition and page layout programs that include both design and merge functions. Other producers may compose photo albums or personalized marketing collateral. **PPML Consumers** include digital front ends (DFE), print engines, converters, and viewers.

PPML producers and consumers may participate in workflows defined by a job ticket such as JDF. PPML provides only meta information that can be used to guide such workflows, for example, selecting media, altering color processing or selecting finishing options. When it is used with a job ticket such as JDF, the PPML data must be organized and tagged in a way that permits association of PPML pages with specific consumer process control parameter sets.

A workflow might include some digital assets provided by design bureaus in static form, while others will be generated from data stored in a database and merged with a template. The PPML producer combines the static and dynamically-generated assets into a PPML dataset and may include meta information to guide the workflow. The PPML producer may choose to include some digital assets directly in the PPML dataset and reference others.

The PPML dataset may be accompanied by a referencing job ticket and sent to a workflow controller or be directly submitted to a consumer. The consumer, in this case a DFE, processes the PPML dataset as described by the job ticket, which may be guided by the meta information contained in the PPML dataset. Remote digital assets referenced by the PPML dataset are retrieved by the DFE during processing. The printed output may be finished inline, nearline, or offline under guidance of the workflow controller.

The purpose of this revision of the PPML Functional specification is to:

- improve interoperability with job ticketing languages, primarily JDF
- complete the separation of content from workflow
- formalize the imaging model and semantics of PPML to eliminate ambiguities in interpretation of the PPML Functional specification

Personalized Print Markup Language

1 Scope

This specification defines the syntax and semantics of PPML, an XML based page layout format designed to enable optimization of both streaming and non-streaming variable data print applications. This format allows for the definition of pages in a way that is independent of the graphical content formats used to specify graphical content elements that may occur on a PPML defined page.

This specification does not identify particular content formats that must be used with PPML data. The definition of complete PPML based datasets suitable for print applications depends upon other specifications or standards to define the use of PPML subsets with specific graphical content formats.

This specification defines conformance for **PPML Producers** and **Consumers**. This specification delegates part of the conformance requirements for Producers and Consumers to registered conformance subsets. This specification defines the requirements for conformance statements that define such conformance subsets.

NOTE Due to the dependencies on other specifications, conformance to this specification alone does not guarantee open interoperability among conforming producing and consuming systems.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including amendments) applies.

PPML is specified both in a Document Type Definition (DTD, http://www.w3.org/TR/1998/REC-xml-19980210#dt-doctype) and in an XML Schema (http://www.w3.org/XML/Schema). All versions of the DTD and Schema are available at http://www.podi.org/ppml.

Adobe Systems. *Postscript Language Reference Manual* (Third Edition). February 1999. Addison-Wesley Professional.

Adobe Systems Incorporated. *PDF Reference* (Second Edition). 2000. Addison-Wesley Professional.

ICC Version 4.2.0.0: ICC Profile Format Specification, version 4.2.0.0. October 2004. http://www.color.org/ICC1V42.pdf

Internet Assigned Numbers Authority. *Official Names for Character Sets*, ed. Keld Simonsen et al. http://www.iana.org/assignments/character-sets

Internet Assigned Numbers Authority (IANA) Assignments http://www.iana.org/assignmentssi.edu/in-notes/iana/assignments

Internet Assigned Numbers Authority (IANA). MIME Media Types. http://www.isi.edu/in-notes/iana/assignments/media-types/media-types.

IETF (Internet Engineering Task Force). *RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax*. T. Berners-Lee, R. Fielding, L. Masinter. 1998. http://www.ietf.org/rfc/rfc2396.txt

ISO. *ISO 8601: Data elements and Interchange formats – Information interchange – Representation of dates and times.* 1988. Also described in the W3C's http://www.w3.org/TR/NOTE-datetime

MIT Laboratory for Computer Science and RSA Data Security, Inc. The MD5 Message-Digest Algorithm. April 1992. http://www.ietf.org/rfc/rfc1321.txt

Network Working Group. Uniform Resource Identifiers (URI): Generic Syntax. August 1998. http://www.ietf.org/rfc/rfc2396.txt

Network Working Group. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. November 1996. http://www.ietf.org/rfc/rfc2045.txt

PKWARE, inc. ZIP File Format Specification Version 6.2.1. April 1, 2005. http://www.pkware.com/business_and_developers/developer/appnote/appnote.txt

World Wide Web Consortium. Tim Bray, Dave Hollander, and Andrew Layman, editors. *Namespaces in XML*, 1999. http://www.w3.org/TR/REC-xml-names/

World Wide Web. Date and Time Formats. 1997. http://www.w3.org/TR/NOTE-datetime

World Wide Web Consortium. *Extensible Markup Language (XML) 1.0* (Third Edition) (February 2004) http://www.w3.org/TR/REC-xml/

2.1 Other references

American National Standards Institute. Graphic technology - Variable printing data exchange using PPML and PDF (PPML/VDX). (CGATS.20-2002). July 2002. http://www.npes.org .

International Cooperation for the Integration of Processes in Prepress, Press and Postpress (CIP4). September 2005. *JDF Specification, Release 1.3*, http://www.cip4.org/

International Cooperation for the Integration of Processes in Prepress, Press and Postpress (CIP4): System Behaviour and Interoperability WG. December 22, 2004. *Base ICS, Version: 1.0*.

ISO 16612-1:(2005) Graphic technology - Variable data printing exchange - Part 1: Using PPML 2.1 and PDF 1.4 (PPML/VDX-2005). May 2005.

PODi: the Digital Printing Initiative. *Digital Print Ticket* (all versions, 2.0/2.2). http://www.podi.org

PODi: the Digital Printing Initiative. *Graphic Arts Conformance Subset Version 2.2*. http://www.podi.org

PODi: the Digital Printing Initiative. *PPML Impositioning Specification Version 2.2.* http://www.podi.org

PODi: the Digital Printing Initiative. *PPML Conformance Subset Registry.* http://www.podi.org

3 Definitions

For the purposes of this specification, the following definitions apply.

3.1
PPML Document
Refer to the definition in section 5.1 Valid PPML document.

3.2
PPML Dataset
Refer to the definition in section 5.2 Conforming PPML datasets.

3.3
PPML Producer
Refer to the definition in section 5.4 Conforming PPML Producer.

3.4
PPML Consumer
Refer to the definition in section 5.5 Conforming PPML Consumer.

3.5
RIP — Raster Image Processor
A hardware device or software application that processes content descriptions in some format and produces raster images.

3.6
DFE — Digital Front End
A hardware device or software application that manages print jobs.

4 Terms, symbols, notations, and abbreviations

Element names are written in a bold sans serif font; for example **DOCUMENT_SET.** Attributes are written in an italic sans serif font with initial capitalization, for example *Label*..

Many PPML element names are common English words; therefore, it is often convenient and accurate to use them conversationally. In this specification, when an element name appears in text *not* in a bold sans serif font, but with Initial Capitals, it is specifically referring to the PPML item that bears that name. When it appears with no capitalization, the word is being used with no special PPML significance. For example:

> The **SOURCE** element contains one or more component files.
>
> In an **OBJECT** element, the Source may contain data in any of several formats.
>
> Customers may submit image data that was gathered from a number of different sources.

The following is an example of a **DOCUMENT_SET** tag with attributes *Name* and *DocumentCount*:

```
<DOCUMENT_SET Label="MyJob"  DocumentCount="150">
```

4.1 Notation for specifying syntax requirements

This PPML Functional specification uses the syntax for Document Type Definitions as defined in the XML specification.

PPML elements shall contain characters as defined in the XML Specification.

5 Conformance

This section defines conformance requirements for PPML datasets, PPML Producers, PPML Consumers, and conformance subsets.

5.1 Valid PPML document

A valid PPML document is a well-formed XML document that adheres to the PPML syntax and semantics as defined in *Section 7 Syntax and semantics, Section 8 Resources*, and *Section 9 Impositioning* except for those semantics that require interpretation of content data or examination of definitions in the Global static scope.

5.2 Conforming PPML datasets

A conforming PPML Dataset shall:

- be a valid PPML document;
- only contain valid references to data in accordance with the PPML semantics defined in *Section 7 Syntax and semantics, Section 8 Resources*, and *Section 9 Impositioning* of this specification;

All referenced data shall adhere to the requirements set forth in *Section 7 Syntax and semantics, Section 8 Resources*, and *Section 9 Impositioning* of this specification.

All references to external data shall exactly match (including case) the names of the referenced files on the system where they are stored.

5.3 PPML Conformance subsets

A PPML conformance subset defines a collection of conforming PPML datasets. See *Annex A* for a list of registered conformance specification documents. A PPML conformance subset shall have:

- a unique name registered with PODi, for example PPML/GA and PPML/VDX;
- conformance requirements described in a publicly available conformation specification document. See *Registered Conformance Statements* for a list of registered conformation specification documents available as of the publication date of this document.

Conformance specification shall specify:

- which PPML elements may be used;
- which PPML attributes and attribute values may be used;
- a non-empty set of content file formats or subsets thereof that may be used to describe content;

- which URL schemes may be used to reference content data.

NOTE It is recommended that all PPML datasets in a PPML conformance subset include a **CONFORMANCE** element whose *Subse*t and *Level* attributes identify adherence to a particular version of that subset.

NOTE A list of defined conformance subsets can be found at the conformance registry at the PODi website.

5.4 Conforming PPML Producer

A conforming PPML Producer shall:

- produce conforming PPML datasets;
- state for which PPML conformance subsets and subset versions datasets can be produced;

5.5 Conforming PPML Consumer

A conforming PPML Consumer shall:

- state which PPML conformance subsets and subset versions are supported;
- either reject the dataset or warn the end-user when asked to process an unsupported PPML dataset;
- convert supported PPML datasets into content tagged as pages, documents and sets in accordance with the semantics defined in sections 6 through 10 of this specification;
- provide a method to allow for the visual inspection of the converted result to confirm that the result has an appearance in conformance with the semantic model described in sections 6 through 10 of this specification;
- ensure that the processing of content data shall not influence future processing of other content data.

NOTE This implies that the processing of content data by the processor should not leave a residual effect in the state of that processor that may otherwise influence the processing of other content data. In some cases, it may be necessary to restart the content data processor between the processing of multiple content data streams.

A PPML Consumer supports a particular PPML dataset if:

- it supports all of the PPML elements, attributes and attribute values in that PPML dataset;
- it supports all of the content data files referenced by or contained in that PPML dataset.

A PPML Consumer supports a particular content format if:

- it supports all or a subset of the content file format identified by the mimetype specified for that content data;
- it processes the content data in accordance with the specification for the content file format of that content data. Constructs in the content data falling outside the subset supported by the PPML Consumer shall either be ignored or rejected.

NOTE It is recommended that a PPML Consumer notify the end-user when ignoring parts of the content data.

NOTE The above definition of supporting a PPML dataset implies that a conforming PPML Consumer cannot ignore parts of a PPML dataset. It must either support a PPML dataset completely or not at all.

NOTE The behavior of a PPML Consumer in response to invalid or non-conforming content data is not defined by this specification. Some PPML Consumers may halt the job or simply issue a warning or error message and may continue processing that data as best they can.

6 Semantic model

This section defines the imaging model for PPML and the rules for handling data definitions. The syntax of PPML is described using DTD syntax as defined in the XML specification.

6.1 The coordinate system

The PPML **coordinate system** is unbounded in all directions. Each unit is 1/72 inch long for each axis. A **point** in the PPML coordinate system is device and resolution independent and is *not* the same as a device pixel. Each point has a location specified by a pair of coordinates. A coordinate is represented by a floating-point number, which can be fractional.

In an implementation, pixels have spatial dimensions; the concept of a point in this semantic model is dimensionless to allow a device independent model. An actual implementation shall approximate the conceptual behaviour as best it can.

6.2 Content

A **compound element** is a single appearance entity that is a composite of one or more content-types including text, graphics, line art or image data. In this specification, the term **content** is defined as a set of marked points and unmarked points, which represents the appearance of a collection of compound elements.

Each **marked point** has a color value in a color space.

A color space may be a standardized color space such as sRGB, SWOP, or CIE L*a*b*. Such standardized color spaces have a prescribed interpretation of color information. Colors specified in standardized color spaces shall have an appearance that closely matches the prescribed colors within the capabilities of the PPML Consumer.

NOTE A rendering intent specification may be used to control which aspects of color fidelity should be optimized when rendering the prescribed colors.

The interpretation of color information specified in device dependent color spaces, such as RGB and CMYK, may result in different colors on different devices.

Since PPML is device independent, the methods of rendering marked points onto the physical medium is out of scope of this specification. To facilitate imaging, each marked point has a mark type. The mark type takes one of the values: smooth shade, vector graphics, text, and image data.

NOTE Depending on the physical rendering technique employed by the rendering device, certain colors may be produced or approximated such that the underlying medium remains visible. For example, on a device using a CMYK process model, for points marked with zero colorant density, no colorant may be laid down, thus showing the color of the underlying medium.

6.3 Source data rendering

Source data describes content. Source data is rendered into a collection of marked and unmarked points by a rendering process. The rendering process may require supplementary source data such as Fonts and ProcSets to render the source data.

PPML provides an explicit rectangular extent for source data used to define content. The dimensions for that rectangular extent are specified by the *Dimensions* attribute of the **SOURCE** element.

The rendering process shall create a collection of marked and unmarked points within the confines of the bounding box (0,0), (*w,h)* given a dimension of width *w* and height *h*.

6.3.1 Coordinate system mapping

In this specification, a **virtual medium** is any representation of a dimensioned rectangular region onto which content is rendered. The dimensions and orientation of the virtual medium shall be equal to the dimensions and orientation of the printed output of that content data as defined by the content format specification.

Rendering some device-dependent source data (e.g. Adobe PostScript and HP PCL), requires knowledge of the dimensions of the virtual medium (*e.g.* as established by the Adobe PostScript `setpagedevice` operator). When the dimensions of the virtual medium cannot be inferred from the source data, the rendering process shall assume that the dimensions of that virtual medium are equal to the dimensions as specified by the *Dimensions* attribute of the PPML **SOURCE** element.

The lower left corner of the virtual medium used by the rendering process is placed at the origin of the PPML coordinate system. Spatial integrity shall be maintained.

For source data specified in a content format having no explicit spatial dimensions defined (for instance, certain image formats without resolution information), the spatial resolution of the source data shall be specified by the *Dimensions* attribute of the PPML **SOURCE** element. The content shall be scaled to match these dimensions, *i.e.* scale to fit. This scaling shall be performed as follows:

> When the source data specifies content of *x* units wide and *y* units high (of spatially undefined units) and for which PPML specifies a width *w* and height *h* in units of 1/72 inch, the spatial size of a one unit square in the source data is taken to be *w/(x*72)* inch wide by *h/(y*72)* inch high such that the content is rendered at the required size of *w* by *h*. The lower left corner of the content rectangle described by the source data shall be placed at the origin of the PPML coordinate system.

6.3.2 Marked and unmarked points

All of the **points** except those that are explicitly marked by the imaging operators in the source data ('the background') are considered unmarked and therefore fully transparent. An image operator using a fully transparent color shall not generate a marked point in PPML.

For images, each sample of the raster that is not fully transparent shall be rendered as a rectangular area of marked points with its own color. Rendering a sample with a fully transparent color shall not generate any marked points. If the image contains an image mask, only unmasked samples of the raster shall be rendered. For images with associated arbitrary clipping areas, only unclipped samples of the raster shall be rendered.

6.3.3 Mark Types

The **mark type** of a marked point is based on the imaging operator(s) that contributed to the color of that point. If a text drawing operator contributed to the color of a marked point, the mark type of that marked point shall be text. If a raster image operator is the sole contributor to the color of a marked point, then the mark type of that marked point shall be image data or continuous tone. The mark type raster data shall be used for image data for which the original image colors have been approximated using a small set of reduced colors; otherwise, the continuous tone mark type shall be used. For marked points, that are only part of a gradient fill or blend the mark type of smooth shade shall be used. In all other cases, a mark type of vector graphics shall be assigned to marked points.

NOTE A PPML Consumer may choose to ignore screening information embedded in the content data.

6.4 Manipulating content

PPML allows content from different sources to be manipulated and combined to create new content. This section describes the formal model for manipulating and combining content.

6.4.1 Content transformation

Content, as described in the PPML coordinate system, can be scaled, reflected, skewed, rotated, and translated by applying a transformation matrix [*a b c d e f*] to it. The point *(x,y)* in the original content is transformed into point *(x',y')* in the resulting content by the following equations:

$x' = a*x + c*y + e$

$y' = b*x + d*y + f$

Rotation by an angle *Φ* can be accomplished by using a transformation matrix of:

$[\cos(\Phi)\ \sin(\Phi)\ -\sin(\Phi)\ \cos(\Phi)\ 0\ 0]$

Translating over a distance (*x,y*) can be accomplished by using a transformation matrix of:

$[1\ 0\ 0\ 1\ x\ y]$

Scaling by a factor of *x* horizontally and *y* vertically can be accomplished by using a transformation matrix of:

$[\ x\ 0\ 0\ y\ 0\ 0]$

The identify transformation (*x' = x*, *y' = y*) is described by the transformation matrix:

$[1\ 0\ 0\ 1\ 0\ 0]$

6.4.2 Combining multiple transformations

Any sequence of rotation, skewing, reflection, scaling and translating transformations can be combined into a single homogeneous transformation matrix. First applying transformation matrix [*a b c d e f*] followed by [*g h i j k l*] creates a transformation of a point (*x,y*) into (*x',y'*) and then into (*x",y"*) as follows:

$x' = a*x + c*y + e,$

$y' = b*x + d*y + f$

$x'' = g*x' + i*y' + k$

$\quad = g*(a*x + c*y + e) + i*(b*x + d*y + f) + k$

$\quad = (g*a + i*b)*x + (g*c + i*d)*y + (g*e + i*f + k)$

$y'' = h*x' + j*y' + l$

$\quad = h*(a*x + c*y + e) + j*(b*x + d*y + f) + l$

$\quad = (h*a + j*b)*x + (h*c + j*d)*y + (h*e + j*f + l)$

The same transformation can be described using a single transformation matrix:

[(g*a + i*b) (h*a + j*b) (g*c + i*d) (h*c + j*d) (g*e + i*f + k) (h*e + j*f + l)]

By repeatedly applying the above rule, any sequence of transformations can be captured into a single transformation matrix.

6.4.3 Content clipping

Clipping unmarks all points (*x,y*) that lie outside a given bounding box defined by the points (*llx,lly)* and *(urx,ury)* satisfying the following relation:

$x < llx$ OR $x > urx$ OR $y < lly$ OR $y > ury$

All other points shall remain unchanged.

6.4.4 Content composition

Content is combined by overlaying some content B onto some other content A. The resulting content shall be the combination of the points in A and B. For each marked point (x,y) in A that is not in B, the resulting content shall contain that marked point. For each marked point (x,y) in B that is not in A, the resulting content shall contain that marked point. For each marked point (x.y) that is both in A and B, the resulting content shall contain the marked point from B (effectively obscuring the underlying content of A).

6.5 Scoping and identification of definitions

PPML is a hierarchical structure; **data definitions** can be introduced and referenced within the different levels of this structure. These data definitions include definitions of content, contexts, supplemental source data (e.g. Fonts and ProcSets), and imposition schemes. **Scoping** defines the availability of data definitions within the context of the PPML hierarchical structure(s) of one or more PPML datasets. See *6.6.1 Context* for information on how **contexts** are applied.

This semantic model uses a hierarchical tree of nodes, where each node is a static scope. The static scopes are named: *Global*, *PPML*, *DocSet*, *Document,* and *Page*. The Global scope is the highest and the Page scope is the lowest scope level. A static scope shall have a *begin* and *end* position associated with it. The *begin* and *end* positions of a lower static scope shall be within the *begin* and *end* positions of a higher static scope in the tree of static scopes.

A data definition has a dynamic scope. A data definition is only valid within its dynamic scope. The dynamic scope shall start at the point of definition within a static scope and shall end at the end of that static scope, except for Global scopes where the end of the dynamic scope shall be determined by the point of definition of the next data definition with the same name and type.

The dynamic scope in which the data definition is valid is known as the scope of the data definition. The name and type of a data definition shall be unique within the static scope in which the data definition is defined, with the exception of the Global scope.

A **reference** shall be an identifier that resolves to a data definition. The identifier for a reference to a data definition shall be the name and type of that data definition. All references to a data definition shall be made within the scope of the data definition to which the reference resolves. A reference resolves to the data definition with the lowest possible dynamic scope that includes the point of reference.

6.6 Semantic model for Ticket States

A PPML dataset may have an associated job ticket. A job ticket is data that conforms to a job ticket format. The semantics of a job ticket shall be defined by that job ticket's format. A job ticket processor interprets a job ticket following the semantics of the job ticket format.

6.6.1 Context

A ticket state is defined by a **context** where the semantics of that context shall be defined by the job ticket associated with the PPML dataset. This **context** shall be defined as a sequence of zero or more identifiers. The valid set of contexts shall be defined by the job ticket format and may be further constrained by the job ticket.

The concatenation of contexts *A* and *B* is the sequence of identifiers that consists of the sequence of identifiers of *A* followed by the sequence of identifiers in *B.*

For any given point in the PPML hierarchy, the ticket state shall be defined as the concatenation of all contexts for which the dynamic scope of each context includes that point in the PPML hierarchy. The contexts shall be concatenated in order of definition.

A context *A* is considered to be defined before a context *B* if the point of definition of *A* lies before the point of definition of *B*.

7 Syntax and semantics

A PPML dataset shall be a well-formed XML document in which all elements and attributes defined in this specification shall have a namespace of "urn://www.podi.org/ppml/ppml2".

NOTE In accordance with the *XML Namespace Recommendation*, it suffices to have an *xmlns* attribute with a value of "urn://www.podi.org/ppml/ppml2" on the PPML element to satisfy the above requirement.

NOTE The Label and Class attributes values have a specific use in a job ticketing context such as defined by the DPT specification and its use of JDF.

7.1 DTD and Schema

The DTD for PPML 2.2 is located at http://www.podi.org/ppml/ppml220.dtd and the corresponding XML Schema is located at http://www.podi.org/ppml/ppml220.xsd.

For example, to validate a PPML Version 2.2 document by means of a DOCTYPE declaration, one would include the following XML mark up immediately following the XML declaration and any intervening white space:

```
<! DOCTYPE PPML PUBLIC
    "-//PODi//DTD PPML 2.20//EN" "http://www.podi.org/ppml/ppml220.dtd">
```

For example, to validate a PPML Version 2.2 document using XML schema the PPML element could look like:

```
<PPML xmlns="urn://www.podi.org/ppml/ppml2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn://www.podi.org/ppml/ppml2
   http://www.podi.org/ppml/ppml220.xsd">
```

7.2 Attribute types

The following attribute types are used throughout this specification. In some cases, the type may be specified as an “attribute type” x “count” indicating that the actual value shall be a valid value for the attribute repeated “count” times separated by white space.

7.2.1 Integer

An *integer* is specified as an optional sign character (‘+’ or ‘–’, with ‘+’ being the default) followed by one or more digits “0” to “9”. The range for a PPML integer encompasses (at a minimum) –2147483648 to +2147483647.

7.2.2 Number

A *number* is either an “integer” or an optional sign character (‘+’ or ‘–’, with ‘+’ being the default) followed by zero or more digits “0” to “9” followed by a dot (.) followed by zero or more digits “0” to “9” with at least one digit required either before or after the dot. An optional exponent may follow the digits after the dot. The exponent is the letter ‘E’ or ‘e’ followed by an *integer*. A *number* has the capacity for at least a single-precision floating point number (see ICC32) and has a range (at a minimum) of –3.4e+38F to +3.4e+38F.

7.2.3 Scope

An attribute of type Scope shall have one of the values “Global”, “PPML”, “DocSet”, “Job”, “Document”, or “Page”.

7.2.4 Boolean

An attribute of type Boolean shall have one of the values “Yes” or “No”.

7.2.5 URI

An attribute of type URI shall have a value whose contents shall adhere to RFC 2396.

7.2.6 Checksum

An attribute of type Checksum shall have a value consisting of characters 0 to 9, A through F, or a through f. The value shall have a length that is a multiple of two.

7.2.7 String

An attribute of type String shall have a value consisting of an arbitrary length sequence of XML characters.

7.2.8 DateTime

An attribute of type DateTime shall have a value whose contents shall adhere to the subset of ISO 8601 described by W3C's Date and Time Formats specification document.

Example: “1997-07-16T19:20:30+01:00”

7.2.9 MimeType

An attribute of type MimeType shall have a value that is a format name registered with IANA as a MIME media type.

Table 1 — IANA Identifiers

Format	IANA identifier
PostScript	`application/postscript`
Encapsulated PostScript (EPS)	`application/postscript`
PDF	`application/pdf`
PCL	`application/vnd.hp-PCL`
PCL XL	`application/vnd.hp-PCLXL`
AFP	`application/vnd.ibm.modcap`
TIFF	`image/tiff`
JPEG	`image/jpeg`
GIF	`image/gif`
SVG (scaleable vector graphics)	`image/svg-xml`

7.2.10 Index

An attribute of type Index shall have an integer value greater than zero.

7.2.11 IndexRange

An attribute of type IndexRange shall have a value containing a comma-separated list of one or more ranges. Each range shall be either a single index or a range of indices given as l-h, where l is smaller than h. The index values must increase monotonically.

7.2.12 Usage

An attribute of type Usage shall have one of the values “Single”, “Multiple” or “Unknown”.

7.2.13 Weight

An attribute of type Weight shall have a number value between 1 and 100, inclusive, where 1 indicates minimum importance and 100 indicates maximum importance.

7.2.14 ResourceType

An attribute of type ResourceType shall have one of the values “Font” or “ProcSet”.

7.2.15 Encoding

An attribute of type Encoding shall have a value of “None” or any encoding name registered with IANA.

7.2.16 CharacterSet

An attribute of type CharacterSet shall have a value that shall be the name of a character set registered with IANA.

7.2.17 ConformanceSubSet

An attribute of type ConformanceSubSet shall have a value that shall be the name of a conformance subset registered with PODi. See *Annex A Conformance Statements*.

7.2.18 Rectangle

An attribute of type Rectangle shall have a value consisting of four numbers separated by white space defining the lower left corner and upper right corner of a rectangle in the PPML coordinate system.

7.2.19 Identifier

An attribute of type Identifier shall have a value consisting of a sequence of one or more XML characters.

7.2.20 OverwriteMode

An attribute of type OverwriteMode shall have a value consisting of “No”, “Yes”, or “Delete””.

7.3 The <PPML> Element

7.3.1 Model

```
PPML     (CONFORMANCE*,
         METADATA*,
         TICKET?,
         SUPPLIED_RESOURCES?,
         REQUIRED_RESOURCES?,
         IMPOSITION*,
         (PRINT_LAYOUT | PAGE_DESIGN)?,
         PRIVATE_INFO*,
         (TICKET_SET | TICKET_REF | REUSABLE_OBJECT | SEGMENT_ARRAY |
          (DOCUMENT_SET | JOB) )*)
```

NOTE Shall only occur as the root element of a PPML dataset.

7.3.2 Attributes

Attribute	**Required /Optional**	**Type**	**Description**
Label	Optional	Identifier	An identifying label for this PPML element
Version	Required	Identifier	States which version of the PPML Functional specification governs the semantics of this PPML dataset. For this specification, the version should be 2.2.
Class	Optional	NMTOKEN	Identifies the PPML dataset.
Creator	Optional	Identifier	Identifies an application or person that created the file, for instance to potentially aid in post-processing.
CreationDate	Optional	DateTIme	Time stamp of creation
ResourcesIncluded	Optional	Boolean	Indicates whether all referenced content data, fonts, and other resources are supplied with the dataset.
SheetLayoutIncluded	Optional	Boolean	Indicates if this dataset includes PPML impositioning instructions.

7.3.3 Description

The **PPML** element is the root element of a PPML dataset, encompassing all other elements of the dataset.

7.3.4 Semantics

The **PPML** element shall define a PPML static scope and a *Global* static scope. The PPML static scope shall be defined in the *Global* static scope. The *Global* static scope shall contain global data definitions defined by previously processed PPML datasets. Those data definitions shall have their point of definition reset to the position of the start tag of the PPML element. The point of definition of the *PPML* static scope shall be the position of the start tag of the PPML element.

A **PPML** element defines a logical data set. That logical data set is defined as the sequence of logical document sets defined by the **DOCUMENT_SET** and **JOB** sub-elements of the **PPML** element that defines that logical data set. The order of the logical document sets is defined by the order in which the **DOCUMENT_SET** and **JOB** elements occur.

If the *ResourcesIncluded* attribute of a **PPML** element is present and has the value “Yes”, the PPML dataset shall obey the following restrictions:

- the PPML dataset as a whole shall have been exchanged in a single transmission to the PPML Consumer;
- no references shall be made to data definitions in the Global static scope except those that have been defined in the PPML dataset itself;
- none of the content data in the PPML dataset shall depend on resources other than those supplied via the PPML dataset.

If the *SheetLayoutIncluded* attribute of a **PPML** element is present and has the value “Yes”, the PPML dataset shall include a **PRINT_LAYOUT** element that shall contain a **SHEET_LAYOUT** element.

The valid *Version* attribute is 2.2 for this specification. Valid values correspond to the versions of the PPML Schema (1.0, 1.01, 1.02, 1.5, 2.1, or 2.2.)

7.3.4 Implementation notes

A PPML dataset is not required to contain any Document Sets. A valid dataset may contain only a set of Reusable Object definitions with `Scope="Global"` which are being sent to the Consumer for pre-processing and storage in the Consumer system.

7.4 *The <DOCUMENT_SET> or <JOB> Element*

7.4.1 Model

```
DOCUMENT_SET   (METADATA*,
            SUPPLIED_RESOURCES?,
            REQUIRED_RESOURCES?,
            IMPOSITION*,
            (PRINT_LAYOUT | PAGE_DESIGN)?,
            PRIVATE_INFO*,
            (TICKET_SET | TICKET_REF | REUSABLE_OBJECT | SEGMENT_ARRAY
            | DOCUMENT)+)
```

NOTE The use of this element is defined in the model of the following element: **PPML**.

7.4.2 Attributes

Attribute	Required /Optional	Type	Description
Label	Optional	Identifier	An identifying label for this Document Set.
Class	Optional	NMTOKEN	Identifies Document Sets or Jobs as members of named groups of Document Sets or Jobs.
DocumentCount	Optional	Integer	Number of ordered sets of related pages in this Document Set. If this attribute is used, it shall be accurate.

7.4.3 Description

A Document Set is a group of documents to be treated as a unit, perhaps because the documents are intended for a single recipient, such as a cover letter, a brochure, cover pages, body pages, insert pages, and a postcard. PPML does not require that Document Set be used in this way; it is merely a convenient grouping mechanism.

NOTE In the case of job ticketed usage the semantics of the page set (the document) are defined by the job ticket data.

7.4.4 Semantics

A **DOCUMENT_SET** or **JOB** element shall define a *DocSet* static scope in the *PPML* static scope. The point of definition of the *DocSet* static scope shall be the position of the start tag of the **DOCUMENT_SET** or **JOB** element.

Each **DOCUMENT_SET** or **JOB** element defines a logical document set. That logical document set is defined as the sequence of logical documents defined by the **DOCUMENT** sub-elements of the

DOCUMENT_SET or **JOB** element that defines that logical document set. The order of the logical documents is defined by the order in which the **DOCUMENT** elements occur.

NOTE **JOB** is a synonym for **DOCUMENT_SET**: they can be used interchangeably. In attribute values, ***Job*** is a synonym for ***DocSet***.

7.5 The <DOCUMENT> Element

7.5.1 Model

```
DOCUMENT    (METADATA*,
            SUPPLIED_RESOURCES?,
            REQUIRED_RESOURCES?,
            PAGE_DESIGN?, PRIVATE_INFO*,
            (TICKET_SET | TICKET_REF | REUSABLE_OBJECT | SEGMENT_ARRAY
            | PAGE)+)
```

NOTE The use of this element is defined in the models of the following elements: **DOCUMENT_SET** or **JOB**.

7.5.2 Attributes

Attribute	Required /Optional	Type	Description
Label	Optional	Identifier	An identifying label for this Document. NOTE In the case of a job ticked workflow, the value of *Label* may be used to associate the pages with product definition semantics or a set of processing parameters (*e.g.* when used with JDF).
Class	Optional	NMTOKEN	Identifies individual Documents as members of named groups of Documents.
Dimensions	Optional	Number × 2	Width and height of pages in this Document, in PPML units. Use of this attribute is no longer recommended. Use the **PAGE_DESIGN** element instead.
PageCount	Optional	Integer	Number of pages in the document. If this attribute is used, it shall be accurate.
DocumentCopies	Optional	Integer	Specifies the number of identical copies to print of this ordered set of related pages.

7.5.3 Description

The **DOCUMENT** element marks a set of related pages. For example, the set of pages resulting from the binding of layout information (*e.g.* a template). In the case of job ticketed usage, the semantics of the page set are defined by the job ticket data. When printing personalized information for people on a mailing list, the Document tag may be used to delimit each individual set of pages that will be sent to one recipient on the list.

7.5.4 Semantics

A **DOCUMENT** element shall define a *Document* static scope in the *DocSet* static scope defined by the parent element of the **DOCUMENT** element. The point of definition of the *Document* static scope shall be the position of the start tag of the **DOCUMENT** element.

Each **DOCUMENT** element defines a logical document. That logical document is defined as the sequence of logical pages defined by the **PAGE** sub-elements of the **DOCUMENT** element that defines that logical document. The order of the logical pages is defined by the order in which the **PAGE** elements occur that define those logical pages.

If the attribute *DocumentCopies* is present on a **DOCUMENT** element, the PPML Consumer shall process that **DOCUMENT** element as if exact copies of that **DOCUMENT** element had been encountered as many times as the *DocumentCopies* attribute specifies.

If the attribute *Dimensions* with a value of “w h” is present on a **DOCUMENT** element, the PPML Consumer shall process that **DOCUMENT** element as if it had a **PAGE_DESIGN** child element with a *TrimBox* attribute with a value of “0 0 w h”. If that **DOCUMENT** element already has a **PAGE_DESIGN** element child the *Dimensions* attribute has no effect.

7.6 *The <PAGE> Element*

7.6.1 Model

```
PAGE        (METADATA*,
            SUPPLIED_RESOURCES?,
            REQUIRED_RESOURCES?,
            PAGE_DESIGN?,
            PRIVATE_INFO*,
            (TICKET_SET | TICKET_REF)*,
            (REUSABLE_OBJECT | SEGMENT_ARRAY | MARK)*)
```

NOTE The use of this element is defined in the model of the following element): **DOCUMENT**.

7.6.2 Attributes

Attribute	Required /Optional	Type	Description
Label	Optional	Identifier	An identifying label for this Page. NOTE In the case of a job ticked workflow, the value of Label may be used to associate the pages with product definition semantics or a set of processing parameters (e.g. when used with JDF).
Class	Optional	NMTOKEN	Identifies individual Pages as members of named groups of Pages.
Dimensions	Optional	Number $\times$ 2	Width and height of this Page, in PPML units. Use of this attribute is no longer recommended. Use the **PAGE_DESIGN** element instead.

7.6.3 Description

The **PAGE** element specifies the graphical content of each individual page in each ordered set of related pages.

7.6.4 Semantics

A **PAGE** element shall define a *Page* static scope within the *Document static scope* defined by the parent element of the **PAGE** element. The point of definition of the *Page static scope* shall be the position of the start tag of the **PAGE** element.

The content defined by a **PAGE** element shall be the composition of the content defined by the **MARK** sub-elements in the order in which they occur. A **PAGE** element that has no **MARK** sub-elements shall be interpreted as an empty page containing no marked points.

The trim box and bleed box that apply to a **PAGE** element shall be defined by the TrimBox and BleedBox in effect for the Page static scope defined by that **PAGE** element. (See the definition of the **PAGE_DESIGN** element.)

The TrimBox and BleedBox that apply to a **PAGE** element indicate that the designer's intent is to specify a logical page whose finished dimensions are equal to the extent of the TrimBox that applies to that **PAGE** element. The content of that logical page shall equal the content defined by the **PAGE** element that applies to that **PAGE** element. The BleedBox that applies to a **PAGE** element identifies bleed edges included by the designer to compensate for imperfect trimming.

If the attribute *Dimensions* with a value of "w h" is present on a **PAGE** element, the PPML Consumer shall process that **PAGE** element as if it had a **PAGE_DESIGN** child element with a *TrimBox* attribute with a value of "0 0 w h". If that **PAGE** element already has a child **PAGE_DESIGN** element, then the *Dimensions* attribute has no effect.

7.7 The <PAGE_DESIGN> Element

7.7.1 Model

```
PAGE_DESIGN EMPTY
```

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, **JOB**, **DOCUMENT,** or **PAGE**.

7.7.2 Attributes

Attribute	Required /Optional	Type	Description
TrimBox	Required	Rectangle	Bounding box of the page content area.
BleedBox	Optional	Rectangle	Bounding box of the page's bleed area.

7.7.3 Description

The **PAGE_DESIGN** element specifies the finished rectangular area of a Page as well as optional bleed box information.

7.7.4 Semantics

The *TrimBox* and *BleedBox* attributes of a **PAGE_DESIGN** element define the TrimBox and BleedBox that shall be in effect within the static scope defined by the parent element of that **PAGE_DESIGN** element. This TrimBox and BleedBox shall be in effect for any lower static scope unless explicitly redefined in that static scope.

If no value is specified for the BleedBox attribute of a **PAGE_DESIGN** element then there shall be no explicit BleedBox in effect. Therefore, in this situation, no explicit hint is provided to the PPML Consumer about the existence of bleed edges of the intended finished page.

At least one **PAGE_LAYOUT** or **PAGE_DESIGN** element must be in effect for each Page.

NOTE The **PAGE_LAYOUT** element and its semantics are defined in the PPML Impositioning specification.

NOTE It is recommended that a **PAGE_DESIGN** element be included to ensure that PPML Consumers that do not implement PPML impositioning instructions, and will therefore ignore the **PAGE_LAYOUT** element, can still know the designer's intent regarding the finished page size.

7.7.4.1 The Trim Box

The required *TrimBox* attribute indicates the rectangular region of interest of the page design and defines the intended finished page size.

This information is useful to a PPML processor such as a PPML viewer, page proofer, or imposition layout tool only interested in page design definitions. An imposition tool, for example, may use the TrimBox information as the description of the intended finished page design, and use its dimensions to locate cut marks on imposed sheets as needed.

7.7.4.2 The Bleed Box

The optional *BleedBox* attribute indicates that page content extends outside of the design rectangle specified by the TrimBox attribute and recommends to a PPML Consumer, such as an imposition processor, a preferred bleed extent.

The *BleedBox* attribute if specified shall completely contain the rectangular region defined by the TrimBox or be equal to it.

If no *BleedBox* is specified then no hint is provided to the PPML Consumer of the existence of bleed edges of the intended finished page.

7.7.5 Implementation

All dimensions in the attributes shall be interpreted as being in “upright” orientation. For instance, a portrait letter-size page will have `PAGE_DESIGN TrimBox="0 0 612 792"` and a landscape letter-size page will have `PAGE_DESIGN TrimBox="0 0 792 612"`. No separate *Orientation* attribute is needed.

NOTE Any page may be rotated later when it is used in imposition. The page itself, and its content, are independent of imposition and printing.

7.8 The <CONFORMANCE> Element

7.8.1 Model

```
CONFORMANCE EMPTY
```

NOTE The use of this element is defined in the model of the following element: **PPML**.

7.8.2 Attributes

Attribute	**Required/ Optional**	**Type**	**Description**
Subset	Required	ConformanceSubSet	Declares to which PPML subset the dataset conforms.
Level	Optional	Identifier	Optional qualifier, further identifying the features within the subset.

7.8.3 Description

The optional **CONFORMANCE** element declares that the enclosing dataset conforms to a specific PPML Conformance subset. This model allows multiple **CONFORMANCE** elements, since a dataset could conform to more than one Conformance subset.

This element occurs at the start of the model for the **PPML** element so that a PPML Consumer can know, from the very beginning that nothing in the dataset exceeds the restrictions of a defined Conformance subset.

The **CONFORMANCE** element informs the PPML Consumer that the dataset meets the subset’s requirements.

7.8.4 Semantics

The *Subset* and *Level* attributes of a **CONFORMANCE** element shall identify a conformance subset specification that may constrain the set of valid PPML datasets. The PPML dataset shall adhere to each of the identified conformance subset specifications.

7.9 The <TICKET>Element {deprecated}

7.9.1 Model

```
  TICKET      (EXTERNAL_DATA | INTERNAL_DATA)
```

NOTE The use of this element is defined in the model of the following element: **PPML**.

7.9.2 Attributes

Attribute	Required /Optional	Type	Description
Format	Required	MimeType	Indicates the job ticket format.

7.9.3 Description

A **TICKET** element may be used to explicitly identify the job ticket data for the dataset. The ticket data may either be stored inline within the **TICKET** element, or stored in a separate file and accessed by an external reference.

If a **TICKET** element is not present in a PPML dataset, all **TICKET_REF** elements defined in that PPML dataset are considered references to ticket data implicitly identified at the application level.

NOTE This element has been deprecated in PPML 2.2 for the purpose of forcing job ticket data to be specified outside of the PPML context. This has been done to better guarantee alignment of PPML as a content resource of a job ticket and to better guarantee device independence of PPML data.

7.9.4 Semantics

A **TICKET** element shall define the job ticket for the PPML dataset that contains that **TICKET** element.

The value of the *Format* attribute of a **TICKET** element shall identify the job ticket format of the job ticket defined for that **TICKET** element.

The data specified via the **EXTERNAL_DATA** or **INTERNAL_DATA** sub-element in a **TICKET** element shall specify the data of the job ticket for that **TICKET** element. That data shall adhere to the job ticket format specified for that ticket element.

7.10 The <TICKET_REF>Element {deprecated}

7.10.1 Model

```
TICKET_REF  EMPTY
```

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, **JOB**, **DOCUMENT, PAGE, TICKET_SET,** and **TICKET_STATE**.

7.10.2 Attributes

Attribute	Required /Optional	Type	Description
ExtIDRef	Optional	Identifier	The name of a piece ticket information in the job ticket.
Ref	Optional	Identifier	The name of a previously defined Ticket Set element.

7.10.3 Description

A **TICKET_REF** can occur in various places within the PPML hierarchy and is used to reference uniquely identified pieces of ticket information defined in the job ticket associated with the PPML dataset. The job itself is either explicitly identified by a **TICKET,** or in the absence of a **TICKET** element, identified implicitly from the context of the dataset at the application level.

These pieces of ticket information referenced from within the PPML hierarchy may be used to specify characteristics of the print product definition, such as media type and single- and two-sided printing for the various pages, as well as process control parameter resources, or other production information unrelated to the definition of a PPML page's content. The semantics of the information referenced by **TICKET_REF** are not defined by this specification and are instead defined in the specification of the particular format identified by the value of the *Format* attribute of the **TICKET** element.

NOTE This element has been deprecated in PPML 2.2 for the purpose of forcing job ticket data to be specified outside of the PPML context. This has been done to better guarantee alignment of PPML as a content resource of a job ticket and to better guarantee device independence of PPML data.

7.10.4 Semantics

A **TICKET_REF** element shall define a ticket state context.

Exactly one *ExtIDRef* or *Ref* attribute shall be present for a **TICKET_REF** element.

The value of the *ExtIDRef* attribute of a **TICKET_REF** element shall be the single identifier contained in the ticket state context that shall be created.

The value of the *Ref* attribute of a **TICKET_REF** element shall identify the data definition of a ticket state context.

If the parent element of a **TICKET_REF** element defines a static scope, then the ticket state context defined for that **TICKET_REF** element shall be defined in that static scope. That ticket state context shall have a point of definition equal to the position of the start tag of the **TICKET_REF** element.

7.11 The <TICKET_SET> Element {deprecated}

7.11.1 Model

```
TICKET_SET  (TICKET_REF*)
```

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, **JOB**, **DOCUMENT**, **PAGE**, **MARK**, **REUSABLE_OBJECT** and **OCCURRENCE_LIST**.

7.11.2 Attributes

Attribute	Required /Optional	Type	Description
ID	Required	Identifier	The name of this Ticket Set.

7.11.3 Description

The **TICKET_SET** element is an aggregation of several **TICKET_REF** elements.

NOTE This element has been deprecated in PPML 2.2 for the purpose of forcing job ticket data to be specified outside of the PPML context. This has been done to better guarantee alignment of PPML as a content resource of a job ticket and to better guarantee device independence of PPML data.

7.11.4 Semantics

A **TICKET_SET** element shall define a data definition of type ticket state context. The value of the *ID* attribute of a **TICKET_SET** element shall be the name of that data definition.

The Ticket State context defined by a **TICKET_SET** element shall be the concatenation of ticket state contexts specified by each **TICKET_REF** sub-element in the order that they are specified.

If no **TICKET_REF** sub-elements are present, the ticket state context defined by the **TICKET_SET** parent shall be an empty sequence of identifiers.

7.12 The <TICKET_STATE> Element {deprecated}

7.12.1 Model

```
   TICKET_STATE    (TICKET_REF*)
```

NOTE The use of this element is defined in the model of the following element: **OCCURRENCE**.

7.12.2 Attributes

None

7.12.3 Description

The **TICKET_STATE** element provides hints during the definition of Occurrences regarding Ticket State contexts that may be in effect when the Occurrence is referenced.

The Ticket State is the state of the PPML Consumer relative to all parameters that can be controlled by a Job Ticket. This list of parameters will vary from PPML Consumer to PPML Consumer, because different products have different features that a Job Ticket can control.

NOTE This element has been deprecated in PPML 2.2 for the purpose of forcing job ticket data to be specified outside of the PPML context. This has been done to better guarantee alignment of PPML as a content resource of a job ticket and to better guarantee device independence of PPML data.

7.12.4 Semantics

A **TICKET_STATE** element shall define a ticket state context.

The ticket state context for a **TICKET_STATE** element shall be defined by the concatenation of ticket state contexts specified by each **TICKET_REF** sub-element in the order in which they are specified.

If a **TICKET_STATE** element has no **TICKET_REF** sub-elements, the ticket state shall be defined by an empty sequence of identifiers.

The semantics of the ticket state context defined by a **TICKET_STATE** element shall be defined by the job ticket for the PPML dataset containing that **TICKET_STATE** element.

7.13 The <MARK> Element

7.13.1 Model

```
MARK        ((VIEW?, OBJECT+) | OCCURRENCE_REF | SEGMENT_REF)
```

NOTE The use of this element is defined in the model of the following element: **PAGE**.

7.13.2 Attributes

Attribute	**Required /Optional**	**Type**	**Description**
Position	Required	Number × 2	Specifies a translation to be applied before placing the content on to a page.

7.13.3 Description

The **MARK** element specifies the placement of content data on a page.

Conceptually, each **MARK** defines a rectangular raster image that consists of “marked” and completely “transparent” pixels. Each **MARK** is rasterized independently from any other **MARK** on the page. When a **MARK** overlaps **MARK**s previously placed on the page, its marked pixels completely obscure the previous **MARK**s’ pixels, and the transparent pixels leave the previous **MARK**s’ pixels unaffected. Which pixels in a **MARK**’s raster image are marked and which are transparent depends on the **MARK**’s content data and the content data format.

7.13.4 Semantics

A **MARK** element shall define content.

The content defined by a **MARK** element shall be the content defined by the children of that **MARK** element transformed by the transformation matrix specified by the value of the *Position* attribute of that **MARK** element. The transformation matrix for a *Position* attribute value of “*x y*” shall be [1 0 0 1 *x y*].

If a **MARK** element has a single **OCCURRENCE_REF** sub-element, then the content defined by the children of that **MARK** element is the content identified by that **OCCURRENCE_REF** sub-element.

If a **MARK** element has a single **SEGMENT_REF** sub-element, then the content defined by the children of that **MARK** element is the content identified by that **SEGMENT_REF** sub-element.

Otherwise, the content defined by the children of that **MARK** element shall be the composition of the content defined by each **OBJECT** sub-element of that **MARK** element in the order in which the

OBJECT sub-elements are specified. If a **VIEW** sub-element is present in that **MARK** element, this content shall have the view applied to it as specified by that **VIEW** sub-element.

7.14 The <VIEW> Element

7.14.1 Model

```
   VIEW        (TRANSFORM?, CLIP_RECT?)
```

NOTE The use of this element is defined in the models of the following elements: **MARK**, **OBJECT**, **REUSABLE_OBJECT**, **OCCURRENCE**, and **SEGMENT_ARRAY**.

7.14.2 Attributes

None

7.14.3 Description

The **VIEW** element defines an optional transformation and clipping that may be applied to content.

7.14.4 Semantics

A **VIEW** element shall define a *View*. A View is a transformation followed by optional clipping.

If present, the **TRANSFORM** sub-element shall specify the transformation for the View.

If present, the **CLIP_RECT** sub-element shall specify the clipping for the View.

NOTE Applying transformation and clipping, as specified by a **VIEW** element, to the composition of content is equivalent to applying the transformation and clipping to the individual content before composition.

7.15 The <TRANSFORM> Element

7.15.1 Model

```
TRANSFORM    EMPTY
```

NOTE The use of this element is defined in the models of the following elements: **VIEW**.

7.15.2 Attributes

Attribute	Required /Optional	Type	Description
Matrix	Required	Number × 6	Defines the transformation matrix to be applied.

7.15.3 Description

The **TRANSFORM** element specifies a transformation to be applied to content.

7.15.4 Semantics

The value of a *Matrix* attribute on a **TRANSFORM** element shall define the transformation matrix that shall be used to transform content.

7.16. The <CLIP_RECT> Element

7.16.1 Model

```
CLIP_RECT    EMPTY
```

NOTE The use of this element is defined in the model of the following element: **VIEW**.

7.16.2 Attributes

Attribute	Required /Optional	Type	Description
Rectangle	Required	Rectangle	Defines the rectangle to be used for clipping.

7.16.3 Description

The **CLIP_RECT** element specifies the clipping to be applied to content.

7.16.4 Semantics

The value of a *Rectangle* attribute on a **CLIP_RECT** element shall define the bounding box that shall be used to clip content. A value of "*llx lly urx ury*" for the *Rectangle* attribute shall define a bounding box of (*llx,lly*), (*urx,ury*) that shall be used to clip content.

7.17 The <OBJECT> Element

7.17.1 Model

```
OBJECT      (METADATA*,SOURCE, VIEW?)
```

NOTE The use of this element is defined in the models of the following elements: **MARK** and **REUSABLE_OBJECT**.

7.17.2 Attributes

Attribute	Required /Optional	Type	Description
Class	Optional	NMTOKEN	Identifies individual objects as members of named groups of objects.
Position	Required	Number × 2	A value of “x y” shall specify a translation over (x,y) to be applied to the content.

7.17.3 Description

The **OBJECT** element defines content that shall be used as part of the definition of a Reusable Object or as a part of a Mark.

7.17.4 Semantics

An **OBJECT** element shall define content.

The content defined by an **OBJECT** element shall be the content defined by the children of that **OBJECT** element transformed by the transformation matrix specified by the value of the *Position* attribute of that **OBJECT** element. The transformation matrix for a *Position* attribute value of “*x y*” shall be [1 0 0 1 *x y*].

The content defined by the children of the **OBJECT** element shall be defined as the content defined by the **SOURCE** element. If a **VIEW** sub-element is present in that **OBJECT** element, this content shall have the View applied to it as defined by that **VIEW** sub-element.

7.18 The <SOURCE> Element

7.18.1 Model

```
SOURCE      ((INTERNAL_DATA | EXTERNAL_DATA)+ | EXTERNAL_DATA_ARRAY)
```

NOTE The use of this element is defined in the model of the following element: **OBJECT**.

7.18.2 Attributes

Attribute	Required /Optional	Type	Description
Dimensions	Required	Number x 2	The width and height of the virtual medium used during rendering of the data. The first number specifies the width and the second number specifies the height.
Format	Required	MimeType	Indicates the content file format of the data.
ClippingBox	Optional	Rectangle	Specifies how the content outside the box is to be clipped.

7.18.3 Description

The **SOURCE** element defines content data to be rendered into content by a specified rendering processor.

NOTE The content data from all enclosed elements are concatenated by the processor in the order the elements occur, and are processed as a single unit by the format processor, the same as if all the data had been submitted to the PPML Consumer as a single object.

7.18.4 Semantics

A **SOURCE** element shall define content.

The rendering processor used shall be identified by the *Format* attribute of a **SOURCE** element. The rendering process shall have access to the supplied resources in effect for the lowest static scope defined by an ancestor element of that **SOURCE** element.

The value of the *Dimensions* attribute of a **SOURCE** element shall define the dimensions used by the rendering processor. A value of "*w h"* for the Dimensions attribute shall define a bounding box of (0,0), (*w,h*) to be used for clipping content. Those dimensions shall match the extent of the virtual medium defined by the source data that is to be rendered. The Dimensions attribute also has the following requirements:

- If there is spatial information in the content data, the *Dimensions* attribute shall clip the content data.

- If there is no spatial information in the content data then the *Dimensions* shall be used and the content data shall be scaled to those *Dimensions*.

- **PDF:** Dimensions shall match the MediaBox of the page.

- **EPS:** Dimensions may be anything, but content shall never be scaled.

- **TIFF:** Dimensions shall match the width and height calculated from the XResolution, YResolution, ResolutionUnit, ImageWidth and ImageLength fields embedded in the TIFF unless the ResolutionUnit field is equal to 1 (no units), in which case the image shall be scaled to the dimensions specified in the Dimensions attribute.

- **JPEG:** in the absence of a JFIF header the image shall be scaled to the dimensions specified in the Dimensions attribute. If a JFIF header is present the Dimensions attribute should match the width and height calculated from the XDensity, YDensity and Units fields in the JFIF header and the number of rows/columns encoded in the JPEG data unless the Units field in the JFIF header equals 0 (no units) in which case the image shall be scaled to the dimensions specified in the Dimensions attribute.

NOTE If the Dimensions do not match the extent information in the content data, the result is undefined and may vary from Consumer to Consumer.

A value of "*llx lly urx ury*" for the *ClippingBox* attribute of a **SOURCE** element shall define a bounding box of (*llx,lly*), (*urx,ury*) to be used for clipping content defined by that **SOURCE** element. If the *ClippingBox* attribute is not present for a **SOURCE** element then the bounding box as specified by the *Dimensions* attribute of that **SOURCE** element shall be used for clipping the content defined for that **SOURCE** element.

The content shall be determined by rendering (with the indicated rendering processor) the source data that is defined by the children of a **SOURCE** element. If the source data defined by the children of a **SOURCE** element is empty then that **SOURCE** element shall define content with no marked points. This content shall be clipped as specified by the *Dimensions* attribute and clipped as specified by the *ClippingBox* attribute of that **SOURCE** element.

If a **SOURCE** element has a single **EXTERNAL_DATA_ARRAY** element then the source data specified by the children of that **SOURCE** element shall be defined solely by that **EXTERNAL_DATA_ARRAY** sub-element.

Otherwise, the source data specified by the children of a **SOURCE** element shall be interpreted as the concatenation of the source data defined by each sub-element of that **SOURCE** element, in the order in which the sub-elements occur. That source data shall be interpreted as a single segment.

7.19 The <EXTERNAL_DATA> Element

7.19.1 Model

```
EXTERNAL_DATA    EMPTY
```

NOTE The use of this element is defined in the models of the following elements: **SOURCE, SEGMENT_ARRAY, SUPPLIED_RESOURCE,** and **TICKET**.

7.19.2 Attributes

Attribute	**Required /Optional**	**Type**	**Description**
Src	Required	URI	The location of the external data.
Checksum	Optional	Checksum	The checksum of the external data.
ChecksumType	Optional	Identifier	Identifies the type of checksum. Default="MD5".
SourceUsage	Optional	Usage	A hint to the Consumer regarding future usage of data from this source.

7.19.3 Description

An **EXTERNAL_DATA** element identifies, by location and access method, the source data to be used for describing content, resources, or job tickets.

7.19.4 Semantics

An **EXTERNAL_DATA** element shall identify data stored external to the PPML data. The data format of that data shall be identified by the value of the *Format* attribute on the parent element of that **EXTERNAL_DATA** element.

The value of the *Src* attribute on an **EXTERNAL_DATA** element shall specify a URI identifying the location of the data.

The value of the *ChecksumType* attribute on an **EXTERNAL_DATA** element shall specify the algorithm to use in calculating the checksum. If the *ChecksumType* attribute is not present, the algorithm shall be the MD5 algorithm. The algorithm used shall define the number of hexadecimal digits to use and whether leading zeros may be omitted. For the MD5 algorithm, the 128-bit value shall be encoded as 32 hexadecimal digits where leading zeros shall be included. A hexadecimal digit shall be defined as an ASCII character from the set "0123456789ABCDEFabcdef".

If present, the *CheckSum* attribute on an **EXTERNAL_DATA** element shall equal the checksum value calculated over the data as indicated by the algorithm specified for that **EXTERNAL_DATA** element.

7.20 The <EXTERNAL_DATA_ARRAY> Element

7.20.1 Model

```
EXTERNAL_DATA_ARRAY   EMPTY
```

NOTE The use of this element is defined in the models of the following elements: **SOURCE.**

7.20.2 Attributes

Attribute	Required /Optional	Type	Description
Src	Required	URI	Location of the external data.
Checksum	Optional	Checksum	The checksum of the external data.
ChecksumType	Optional	Identifier	Identifies the type of checksum. Default="MD5".
Index	Optional	Index	The index of the segment to be used. Default="1".
IndexUsage	Optional	Usage	A hint to the Consumer, stating whether additional segments of the referenced file may be used later.

7.20.3 Description

An **EXTERNAL_DATA_ARRAY** element identifies, by location and access method, a multi-segment source datum out of which one segment is selected to be used as content.

7.20.4 Semantics

An **EXTERNAL_DATA_ARRAY** element shall define source data.

The value of the *Src* attribute on an **EXTERNAL_DATA_ARRAY** element shall specify a URI identifying the location of the data from which the source data will be extracted. The data format of that data shall be identified by the value of the *Format* attribute on the parent element of that **EXTERNAL_DATA_ARRAY** element.

The value of the *ChecksumType* attribute on an **EXTERNAL_DATA_ARRAY** element shall specify the algorithm to use in calculating the checksum. If the *ChecksumType* attribute is not present, the algorithm shall be the MD5 algorithm. The algorithm used shall define the number of hexadecimal digits to use and whether leading zeros may be omitted. For the MD5 algorithm, the 128-bit value shall be encoded as 32 hexadecimal digits where leading zeros shall be included. A hexadecimal digit shall be defined as an ASCII character from the set "`0123456789ABCDEFabcdef`".

If present, the *CheckSum* attribute on an **EXTERNAL_DATA_ARRAY** element shall equal the checksum value calculated over the data as indicated by the algorithm specified for that **EXTERNAL_DATA_ARRAY** element.

The value of the *Index* attribute on an **EXTERNAL_DATA_ARRAY** element defines which segment from multi-segment data shall be used to define the source data. A value of “1” specifies that the first segment shall be used. The index value shall not exceed the quantity of segments in the multi-segment data.

7.21 The <INTERNAL_DATA> Element

7.21.1 Model

```
INTERNAL_DATA   ANY
```

NOTE The use of this element is defined in the models of the following elements: SOURCE, SEGMENT_ARRAY, SUPPLIED_RESOURCE, and TICKET,

7.21.2 Attributes

Attribute	**Required /Optional**	**Type**	**Description**
Label	Optional	Identifier	An identifier for this element. NOTE In the case of a job ticked workflow, the value of Label may be used to associate the pages with product definition semantics or a set of processing parameters (e.g. when used with JDF).
Creator	Optional	Identifier	Identifies the application that created this content.
Encoding	Optional	Encoding	Encoding scheme of the data: Default=”None”.
CharacterSet	Optional	CharacterSet	Specifies the character set of the decoded data.

7.21.3 Description

An **INTERNAL_DATA** element contains source data to be used for describing content, resources, or job tickets.

NOTE The content data itself, contained in the **INTERNAL_DATA** element, must be valid XML content.

7.21.4 Semantics

An **INTERNAL_DATA** element shall define data. The data format of that data shall be identified by the value of the *Format* attribute on the parent element of that **INTERNAL_DATA** element.

The value of the *Encoding* attribute on an **INTERNAL_DATA** element shall specify the algorithm to decode the contents of that **INTERNAL_DATA** element. If no *Encoding* attribute is present for an **INTERNAL_DATA** element then no decoding shall occur and the data defined for that **INTERNAL_DATA** element shall be character data.

The value of the *CharacterSet* attribute on an **INTERNAL_DATA** element shall specify the character set of the defined data. If no *CharacterSet* attribute is present the character set shall be assumed to be the same character set as used for the PPML dataset (as defined by the XML header of that PPML dataset) containing that **INTERNAL_DATA** element. That character set shall only be used if the data format requires character data. In all other cases, the data defined for that **INTERNAL_DATA** element shall be binary.

If no *Encoding* attribute on an **INTERNAL_DATA** element is present and the data format of that **INTERNAL_DATA** element requires binary data, the contents of that **INTERNAL_DATA** element shall be converted into binary data by encoding these contents using the encoding specified for the PPML dataset containing that **INTERNAL_DATA** element (as defined by the XML header of that PPML dataset).

NOTE The above definition states that when binary data is needed, the bytes that make up the content of the INTERNAL_DATA element are used without interpretation.

An **INTERNAL_DATA** element shall contain a single well-formed XML element or PCDATA.

NOTE An **INTERNAL_DATA** element may be empty.

If the content of an **INTERNAL_DATA** element consists of a single well-formed XML element then:

- the data defined by that **INTERNAL_DATA** element shall be the sequence of bytes between the end of the **INTERNAL_DATA** start tag and the beginning of the **INTERNAL_DATA** end tag including all the white space;
- that **INTERNAL_DATA** element shall be the only child of its parent;
- the content of the **INTERNAL_DATA** element shall specify a namespace that differs from any of the PPML namespaces and shall not be the empty namespace;
- the *CharacterSet* attribute of that **INTERNAL_DATA** element shall not be present;
- the *Encoding* attribute of that **INTERNAL_DATA** element shall not be present.

NOTE The above rules for XML content inside an **INTERNAL_DATA** element guarantee that a consumer will not need a separate XML parser to process the data.

7.22 The <REUSABLE_OBJECT> Element

7.22.1 Model

```
   REUSABLE_OBJECT   (OBJECT+, VIEW?, OCCURRENCE_LIST)
```

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, **JOB**, **DOCUMENT** or **PAGE,**

7.22.2 Attributes

None

7.22.3 Description

The **REUSABLE_OBJECT** element defines a piece of content that is intended for multiple use. Reusable Objects exist for efficiency: to store frequently used items so they may be accessed with reduced processing.

7.22.4 Semantics

A **REUSABLE_OBJECT** element shall define content.

The content defined by a **REUSABLE_OBJECT** element shall be the composition of the content defined by each **OBJECT** sub-element of that **REUSABLE_OBJECT** element in the order in which the **OBJECT** sub-elements are specified. If a **VIEW** sub-element is present in that **REUSABLE_OBJECT** element, that content shall have the view applied to it as specified by that **VIEW** sub-element.

7.23 The <OCCURRENCE_LIST> Element

7.23.1 Model

```
OCCURRENCE_LIST (OCCURRENCE)+
```

7.23.2 Attributes

None

NOTE The use of this element is defined in the models of the following elements: **REUSABLE_OBJECT,**

7.23.3 Description

Within a **REUSABLE_OBJECT** definition element, the **OCCURRENCE_LIST** element declares named views (occurrences) that may be applied to the reusable content.

7.23.4 Semantics

None

7.24 The <OCCURRENCE> Element

7.24.1 Model

```
   OCCURRENCE  (VIEW?, TICKET_STATE*)
```

NOTE The use of this element is defined in the model of the following element: **OCCURRENCE_LIST**,

7.24.2 Attributes

Attribute	**Required /Optional**	**Type**	**Description**
Name	Required	Identifier	Name to use to refer to this **OCCURRENCE .**
Environment	Required if `Scope ="Global"`; not needed otherwise	Identifier	Specifies the environment in which a global occurrence should be defined. NOTE There is no guarantee that previously defined global objects are retained by a Consumer indefinitely.
Scope	Optional	Scope	Specifies the scope into which this occurrence is to be defined.
Overwrite	Optional	OverwriteMode	Defines whether global occurrences shall be redefined. or deleted. Default = "no."
Weight	Optional	Weight	The relative importance of caching this occurrence.

7.24.3 Description

The **OCCURRENCE** element specifies the View and relative importance with which a particular rendition of a Reusable Object will occur. By specifying Occurrence information in the definition of a Reusable Object, the PPML Producer facilitates optimization of rendering and storage by the eventual PPML Consumer.

7.24.4 Semantics

An **OCCURRENCE** element shall define a content data definition whose name is defined by the value of the *Name* and *Environment* attributes of that **OCCURRENCE** element. That content data definition shall be defined in a static scope defined by an ancestor element of the **OCCURRENCE** element. The static scope in which the content data definition for an **OCCURRENCE** element shall be defined shall be identified by the value of the *Scope* attribute of that **OCCURRENCE** element. If the *Scope* attribute of an **OCCURRENCE** element is not present, then the lowest static scope defined by an ancestor element of that **OCCURRENCE** element (that defines a static scope) shall be used.

If the *Scope* attribute of an **OCCURRENCE** element indicates that the data definition shall be made in the Global static scope and a data definition with the same name already exists in that Global static scope and the *Overwrite* attribute of an **OCCURRENCE** has a value of “*No*”, no new data definition shall be defined.

If the *Scope* attribute of an **OCCURRENCE** element indicates that the data definition shall be made in the Global static scope and a data definition with the same name already exists in that Global static scope and the *Overwrite* attribute of an **OCCURRENCE** has a value of “Delete”, the dynamic scope of that data definition shall end at the point of definition of the **OCCURRENCE** element.

NOTE Overwrite="Delete" signifies to the Consumer that the **OCCURRENCE** is no longer needed. A Consumer may safely remove all data related to that **OCCURRENCE**.

The content data definition shall take its value from the content defined by the **REUSABLE_OBJECT** ancestor element of an **OCCURRENCE** element. If a **VIEW** element is present on an **OCCURRENCE** element the view defined by that **VIEW** element shall be applied to the content defined by that **OCCURRENCE** element.

Each **TICKET_STATE** sub-element of an **OCCURRENCE** element shall define a ticket state context in which the content data definition defined by that **OCCURRENCE** element may be referenced. If an **OCCURRENCE** element has no **TICKET_STATE** sub-elements then the content data definition shall allow a reference in any ticket state context.

7.25 The <OCCURRENCE_REF> Element

7.25.1 Model

```
OCCURRENCE_REF EMPTY
```

NOTE The use of this element is defined in the model of the following element: **MARK.**

7.25.2 Attributes

Attribute	Required /Optional	Type	Description
Ref	Required	Identifier	Name of a previously defined Occurrence for this object.
Environment	Optional	Identifier	The environment in which the name of a Global Occurrence should be interpreted. (This attribute is required if the scope of the Occurrence is Global; otherwise, this attribute has no meaning.)

7.25.3 Description

The **OCCURRENCE_REF** element references content previously marked as being reusable.

7.25.4 Semantics

An **OCCURRENCE_REF** element shall define content.

The values of the *Ref* attribute and *Environment* attribute of an **OCCURRENCE_REF** element shall identify a content data definition whose content shall be used for the content defined for that **OCCURRENCE_REF** element. The point of reference used in identifying that content data definition shall be the position of the start tag of an **OCCURENCE_REF** element.

The content data definition identified by an **OCCURRENCE_REF** element shall not prohibit its use in the ticket state associated with the position of the start tag of that **OCCURRENCE_REF** element in the PPML dataset containing that **OCCURRENCE_REF** element.

The content defined by an **OCCURRENCE_REF** element shall be rendered according to the ticket state defined by the PPML dataset containing that **OCCURRENCE_REF** element for the position of the start tag of that **OCCURRENCE_REF** element.

7.26 The <SEGMENT_ARRAY> Element

7.26.1 Model

```
SEGMENT_ARRAY     (VIEW?, (INTERNAL_DATA | EXTERNAL_DATA)?)
```

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, **JOB**, **DOCUMENT,** or **PAGE,**

7.26.2 Attributes

Attribute	Required/ Optional	Type	Description
Dimensions	Required	Number × 2	The width and height of the virtual medium used during rendering of the data. The first number specifies the width and the second number specifies the height.
Format	Required	MimeType	Indicates the content file format of the content data.
ClippingBox	Optional	Rectangle	Specifies additional clipping of the content.
Name	Required	Identifier	Name to be used when referencing segments.
Environment	Required if `Scope = "Global"`; not needed otherwise	Identifier	Specifies the environment in which a global object shall be defined. NOTE There is no guarantee that previously defined global objects are retained by a Consumer indefinitely.
Scope	Optional	Scope	Specifies the scope in which the reusable content should be defined.
Overwrite	Optional	OverwriteMode	Defines whether global occurrences shall be redefined. or deleted. Default = “No.”
Src	Optional	URI	Identifies the location of the content data. (Deprecated)
Checksum	Optional	Checksum	The checksum of the referenced content data.
ChecksumType	Optional	Identifier	Identifies the type of checksum. Default=”`MD5`”.
Weight	Optional	Weight	The relative importance of caching the reusable content.

Attribute	Required/ Optional	Type	Description
IndexRange	Required	IndexRange	Specifies which of the segments within the content data to process and cache within the Consumer.

7.26.3 Description

The **SEGMENT_ARRAY** element defines a collection of reusable content from multiple segments of a multiple segment source file.

7.26.4 Semantics

A **SEGMENT_ARRAY** element shall define zero or more content data definitions.

The value of the *Format* attribute on a **SEGMENT_ARRAY** element shall specify the rendering process used for that **SEGMENT_ARRAY** element. That rendering process shall allow source data access to all supplied resources in effect for the lowest static scope defined by an ancestor element of that **SEGMENT_ARRAY** element.

A content data definition shall be defined for each index in the set of indices specified by the value of the *IndexRange* attribute of the **SEGMENT_ARRAY** element. The name of each such content data definition shall be the combination of the values of the *Name* and *Environment* attributes of that **SEGMENT_ARRAY** element and the index for which the content data definition shall be made. The value of each such content data definition shall be the content defined by the children of that **SEGMENT_ARRAY** element for the segment with the index for which that content data definition shall be made.

All content data definitions defined by a **SEGMENT_ARRAY** element shall be defined in a static scope defined by an ancestor element of that **SEGMENT_ARRAY** element. The *Scope* attribute of that **SEGMENT_ARRAY** element shall specify in which static scope the content data definition for that **SEGMENT_ARRAY** element shall be made. If a Scope attribute is not present for a **SEGMENT_ARRAY** element, all content data definitions shall be made in the lowest static scope defined by an ancestor element of that **SEGMENT_ARRAY** element.

A redefinition of a **SEGMENT_ARRAY** on a lower scope completely hides the ones on a higher scope. Consequently, a reference to a segment that is not in the *IndexRange* of the **SEGMENT_ARRAY** on the lowest scope results in an empty mark and is not resolved by a possible segment on a higher scope.

All content data definitions defined by a **SEGMENT_ARRAY** shall allow a reference in any ticket state.

The value of the *Dimensions* attribute on a **SEGMENT_ARRAY** element shall define the dimensions used by the rendering process to render source data from the data defined by the **INTERNAL_DATA** or **EXTERNAL_DATA** sub-element of that **SEGMENT_ARRAY** element.

The rendering process defined for a **SEGMENT_ARRAY** element shall define the content defined by the children of that **SEGMENT_ARRAY** element for a given segment.

A value "*w h*" of the *Dimensions* attribute on a **SEGMENT_ARRAY** element shall define a bounding box of (*0,0), (w,h)* that shall be used for clipping content defined by the children of that **SEGMENT_ARRAY** element. Those dimensions shall match the extent of the virtual medium defined by the source data that is to be rendered. The *Dimensions* attribute also has the following requirements:

- If there is spatial information in the content data, the *Dimensions* attribute shall clip the content data.
- If there is no spatial information in the content data then the *Dimensions* shall be used and the content data shall be scaled to those *Dimensions*.
- **PDF:** Dimensions shall match the MediaBox of the page.
- **EPS:** Dimensions may be anything, but content shall never be scaled.
- **TIFF:** Dimensions shall match the width and height calculated from the XResolution, YResolution, ResolutionUnit, ImageWidth and ImageLength fields embedded in the TIFF unless the ResolutionUnit field is equal to 1 (no units), in which case the image shall be scaled to the dimensions specified in the Dimensions attribute.
- **JPEG:** in the absence of a JFIF header the image shall be scaled to the dimensions specified in the Dimensions attribute. If a JFIF header is present the Dimensions attribute should match the width and height calculated from the XDensity, YDensity and Units fields in the JFIF header and the number of rows/columns encoded in the JPEG data unless the Units field in the JFIF header equals 0 (no units) in which case the image shall be scaled to the dimensions specified in the Dimensions attribute.

NOTE If the Dimensions do not match the extent information in the content data, the result is undefined and may vary from Consumer to Consumer.

A value "*llx lly urx ury*" of the *ClippingBox* attribute on a **SEGMENT_ARRAY** element shall define a bounding box of (*llx,lly),* (*urx,ury)* that shall be used for clipping content defined by the children of that **SEGMENT_ARRAY** element. If the *ClippingBox* attribute is not present for a **SEGMENT_ARRAY** element then the bounding box as specified by the *Dimensions* attribute of that **SEGMENT_ARRAY** element shall be used for clipping content defined by the children of that **SEGMENT_ARRAY** element.

If a **VIEW** sub-element is present in a **SEGMENT_ARRAY** element then the view defined by that **VIEW** sub-element shall be applied to content defined by the children of that **SEGMENT_ARRAY** element before the clipping shall occur as defined by the *Dimensions* and *ClippingBox* attributes of that **SEGMENT_ARRAY** element.

The *CheckSum* and *ChecksumType* attributes of a **SEGMENT_ARRAY** element shall only be used when a *Src* attribute is present on that **SEGMENT_ARRAY** element.

If an *Src* attribute is present on a **SEGMENT_ARRAY** element, then that **SEGMENT_ARRAY** shall have the same semantics as if that **SEGMENT_ARRAY** element had an **EXTERNAL_DATA** sub-element with the same values specified for the *Src, CheckSum* and *ChecksumType* attributes as specified for that **SEGMENT_ARRAY** element.

If a *Src* attribute is present on a **SEGMENT_ARRAY** element, that **SEGMENT_ARRAY** element must be empty.

NOTE The use of a Src attribute is deprecated and may be removed altogether in a future version of this specification.

If the *Scope* attribute of a **SEGMENT_ARRAY** element indicates that the data definition shall be made in the Global static scope and a data definition with the same name already exists in that Global static scope and the *Overwrite* attribute of a **SEGMENT_ARRAY** has a value of "*No*", no new data definition shall be defined.

If the *Scope* attribute of a **SEGMENT_ARRAY** element indicates that the data definition shall be made in the Global static scope and a data definition with the same name already exists in that Global static scope and the *Overwrite* attribute of a **SEGMENT_ARRAY** has a value of "Delete", the dynamic scope of that data definition shall end at the point of definition of the **SEGMENT_ARRAY** element.

If the *Scope* attribute of a **SEGMENT_ARRAY** element indicates that the data definition shall be made in the Global static scope and the *Overwrite* attribute of that **SEGMENT_ARRAY** element has a value of "Yes" the dynamic scope of all content data definitions in the Global static scope with the same name shall end at the point of definition of the **SEGMENT_ARRAY** element.

NOTE In PPML 2.1 a partial redefinition was possible allowing segments from two or more **SEGMENT_ARRAY** definitions to be referenced. This feature has been removed as Producers were not using it and that feature complicated Consumer implementations.

7.27 The <SEGMENT_REF> Element

7.27.1 Model

```
SEGMENT_REF EMPTY
```

NOTE The use of this element is defined in the model of the following element: **MARK.**

7.27.2 Attributes

Attribute	Required/ Optional	Type	Description
Ref	Required	Identifier	Specifies the name of the multi-segment content that is to be selected.
Index	Optional	Index	Indicates which segment of the original multi-segment content is to be selected. Default = “1”.
Environment	Optional	Identifier	Specifies the environment in which the name of the global-scoped segment is defined.

7.27.3 Description

The **SEGMENT_REF** element references previously defined reusable content.

7.27.4 Semantics

A **SEGMENT_REF** element shall reference pre-defined content.

The values of the *Ref, Environment*, and *Index* attributes of a **SEGMENT_REF** element shall identify a content data definition whose content shall be used for the content defined for that **SEGMENT_REF** element. The point of reference used in identifying that content data definition shall be the position of the start tag of that **SEGMENT_REF** element.

The content defined by a **SEGMENT_REF** element shall be rendered according to the ticket state defined by the PPML dataset containing that **SEGMENT_REF** element for the position of the start tag of that **SEGMENT_REF** element.

8 Resources

8.1 The <SUPPLIED_RESOURCES> Element

8.1.1 Model

```
SUPPLIED_RESOURCES     (SUPPLIED_RESOURCE+)
```

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, **JOB**, **DOCUMENT,** or **PAGE**.

8.1.2 Attributes

None

8.1.3 Description

The **SUPPLIED_RESOURCES** element contains one or more supplied resource definitions.

8.1.4 Semantics

A **SUPPLIED_RESOURCES** element is a syntactical construct that is a container of supplied resource definitions.

NOTE The order in which resources are defined may not be the same order in which resources are loaded into the content processors memory.

8.2 *The <SUPPLIED_RESOURCE> Element*

8.2.1 Model

```
SUPPLIED_RESOURCE   (INTERNAL_DATA | EXTERNAL_DATA)?
```

NOTE The use of this element is defined in the model of the following element: **SUPPLIED_RESOURCES.**

8.2.2 Attributes

Attribute	Required /Optional	Type	Description
Name	Required	Identifier	An identifying name for this resource for use in **SUPPLIED_RESOURCE_REF**.
ResourceName	Required	Identifier	Name of the resource as referenced by the content of the **SOURCE** elements in which it is used.
Src deprecated in version 2.0	Optional	URI	Location of the resource file.
Format	Required	MimeType	Data format of the resource.
Type	Required	ResourceType	The type of resource to define.
SubType	Optional	Identifier	Optional resource subtype, e.g. Type1, TrueType etc.
Environment	Required if Scope ="Global"; not needed otherwise	Identifier	Specifies the environment in which a global resource should be defined. NOTE There is no guarantee that previously defined global objects are retained by a Consumer indefinitely.
Overwrite	Optional	OverwriteMode	Defines whether global resources shall be redefined. or deleted. Default = “No.”
Scope	Optional	Scope	Specifies the scope in which the resources should be defined.

8.2.3 Description

The Supplied Resource is a definition of a reusable resource such as a font or a PostScript ProcSet. A Supplied Resource cannot be used in content data until it is referenced by a

SUPPLIED_RESOURCE_REF in a **REQUIRED_RESOURCES** element that defines the dependency of that content data on that Supplied Resource.

8.2.4 Semantics

A **SUPPLIED_RESOURCE** element shall define a supplementary source data definition.

A supplementary source data definition defined by a **SUPPLIED_RESOURCE** element shall be defined in a static scope defined by an ancestor element of that **SUPPLIED_RESOURCE** element. The *Scope* attribute of that **SUPLLIED_RESOURCE** element shall specify in which static scope the data definition for that **SUPPLIED_RESOURCE** element shall be made. If a *Scope* attribute is not present for a **SUPPLIED_RESOURCE** element, all data definitions shall be made in the lowest static scope defined by an ancestor element of that **SUPPLIED_RESOURCE** element.

A value "Font" for a *Type* attribute of a **SUPPLIED_RESOURCE** element shall specify that a Font data definition shall be defined by that **SUPPLIED_RESOURCE** element.

The value of a *SubType* attribute of that **SUPPLIED_RESOURCE** element may qualify the type of Font data definition that shall be defined for that **SUPPLIED_RESOURCE** element.

A value "ProcSet" for a *Type* attribute of a **SUPPLIED_RESOURCE** element shall specify that a ProcSet data definition shall be defined by that **SUPPLIED_RESOURCE** element.

NOTE When using Adobe PostScript, the execution of an Adobe PostScript ProcSet definition shall execute the `defineresource` operator for the `/ProcSet` category and with a name equal to the value of the ResourceName attribute of the **SUPPLIED_RESOURCE** element. In Adobe PostScript source data for which this supplied resource is a required resource, execution of the `findresource` operator with a name equal to the value of the ResourceName attribute and category `/ProcSet` shall return the dictionary defined during the execution of the Adobe PostScript ProcSet definition.

NOTE The PPML Functional specification only defines requirements for when identified resources shall be available in content data. The PPML Functional specification does not place any further requirement on when resources are added or removed from VM. Testing for the presence or absence of a resource may therefore yield Consumer implementation dependent results.

The value of the *Name* attribute of a **SUPPLIED_RESOURCE** element shall define the name of the data definition that shall be defined for that **SUPPLIED_RESOURCE** element, unless the value of the *Scope* attribute equals "Global", in which case the name of the data definition is the combination of the *Name* and *Environment* attributes of that **SUPPLIED_RESOURCE** element.

The value of the *ResourceName* attribute of a **SUPPLIED_RESOURCE** element shall define the name under which source data may gain access to the supplementary source data contained in the data definition that shall be defined for that **SUPPLIED_RESOURCE** element.

If a *Src* attribute is present on a **SUPPLIED_RESOURCE** element, then that **SUPPLIED_RESOURCE** shall have the same semantics as if that **SUPPLIED_RESOURCE** element had an **EXTERNAL_DATA** sub-element with the same values specified for the Src attribute as specified for that **SUPPLIED_RESOURCE** element.

If a *Src* attribute is present on a **SUPPLIED_RESOURCE** element, that **SUPPLIED_RESOURCE** element must be empty.

NOTE The use of a Src attribute is deprecated and may be removed altogether in a future version of this specification.

The supplementary source data for the data definition that shall be defined for a **SUPPLIED_RESOURCE** element shall be defined by the sub element of that **SUPPLIED_RESOURCE**.

NOTE Required resources that are in effect at the point of definition of a **SUPPLIED_RESOURCE** element can be referenced in the supplementary source data defined by that **SUPPLIED_RESOURCE** element.

If the *Scope* attribute of a **SUPPLIED_RESOURCE** element indicates that the data definition shall be made in the Global static scope and a data definition with the same name already exists in that Global static scope and the *Overwrite* attribute of the **SUPPLIED_RESOURCE** has a value of "Delete", the dynamic scope of that data definition shall end at the point of definition of that **SUPPLIED_RESOURCE** element.

NOTE Persistence of Global scope data is not guaranteed by a PPML Consumer.

If the *Scope* attribute of a **SUPPLIED_RESOURCE** element indicates that the data definition shall be made in the Global static scope and a data definition with the same name already exists in that Global static scope and the *Overwrite* attribute of that **SUPPLIED_RESOURCE** element has a value of "No", no new data definition shall be defined.

If the *Overwrite* attribute of a **SUPPLIED_RESOURCE** element has a value of "Delete" then the supplementary source data defined by the sub-element of that **SUPPLIED_RESOURCE** may be empty. In all other cases, the supplementary source data shall be non-empty.

8.3 The <SUPPLIED_RESOURCE_REF> Element

8.3.1 Model

```
SUPPLIED_RESOURCE_REF    EMPTY
```

NOTE The use of this element is defined in the model of the following element: **REQUIRED_RESOURCES**.

8.3.2 Attributes

Attribute	Required /Optional	Type	Description
Name	Required	Identifier	The name of a named **SUPPLIED_RESOURCE** element.
Environment	Required if Scope ="Global"; not needed otherwise	Identifier	Specifies the environment in which a global resource should be defined. NOTE There is no guarantee that previously defined global objects are retained by a Consumer indefinitely.

8.3.3 Description

This element defines a dependency for source data on a previously supplied resource.

8.3.4 Semantics

A **SUPPLIED_RESOURCE_REF** element shall define a supplied resource.

The value of the *Name* attribute of a **SUPPLIED_RESOURCE_REF** shall identify a data definition whose supplemental source data shall define the supplied resource for that **SUPPLIED_RESOURCE_REF** element. The point of reference used in identifying the data definition for that supplemental source data shall be the position of the start tag of that **SUPPLIED_RESOURCE_REF** element.

The supplied resource defined by a **SUPPLIED_RESOURCE_REF** shall have a name whose value is specified by the *ResourceName* attribute of the **SUPPLIED_RESOURCE** element that defined the supplemental source data contained by that supplied resource. When referencing data definitions in the Global scope the name of the data definition is the combination of the *Name* and *Environment* attributes of the **SUPPLIED_RESOURCE_REF** element.

The supplied resource defined by a **SUPPLIED_RESOURCE_REF** shall be in effect for the lowest static scope defined by an ancestor of that **SUPPLIED_RESOURCE_REF** element. That supplied resource shall be in effect for any lower static scope unless explicitly redefined in that static scope.

NOTE The ResourceName attribute of the **SUPPLIED_RESOURCE** element that is referenced by a **SUPPLIED_RESOURCE_REF** element must be unique in the scope in which the supplied resource is being defined.

8.4 The <REQUIRED_RESOURCES> Element

8.4.1 Model

```
REQUIRED_RESOURCES   (FONT*,
             PROCESSOR*,
             SUPPLIED_RESOURCE_REF*)
```

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, **JOB**, **DOCUMENT,** or **PAGE.**

8.4.2 Attributes

None

8.4.3 Description

The **REQUIRED_RESOURCES** element identifies resources required to process source data referenced from within the parent of that **REQUIRED_RESOURCES** element.

8.4.4 Semantics

Each of the child elements of a **REQUIRED_RESOURCES** element identifies a resource that shall be available for proper processing of source data. These resources may be needed for processing source data specified by **SOURCE** or **SUPPLIED_RESOURCE** elements descendant from the parent of the **REQUIRED_RESOURCES** element and that occur after that **REQUIRED_RESOURCES** element.

8.5 The <FONT> Element

8.5.1 Model

```
    FONT            EMPTY
```

NOTE The use of this element is defined in the model of the following element: **REQUIRED_RESOURCES.**

8.5.2 Attributes

Attribute	**Required /Optional**	**Type**	**Description**
FontName	Required	Identifier	Name of the font as referenced by the content of the **SOURCE** elements in which it is used.
Format	Required	MimeType	Data format of the font.

8.5.3 Description

The **FONT** element identifies a required font resource.

8.5.4 Semantics

A **FONT** element shall identify a font by name and format. The *Format* attribute of a **FONT** element shall be present and its value shall specify the format of the font as an IANA registered format name. The *FontName* attribute of a **FONT** element shall be present and its value shall specify the name of the font.

8.6 The <PROCESSOR> Element

8.6.1 Model

```
    PROCESSOR    EMPTY
```

NOTE The use of this element is defined in the model of the following element: **REQUIRED_RESOURCES**.

8.6.2 Attributes

Attribute	Required /Optional	Type	Description
Format	Required	MimeType	Name of the language or file format.
Revision	Optional	Identifier	The revision of the file format.

8.6.3 Description

The **PROCESSOR** element names a required content file format processor.

8.6.4 Semantics

A **PROCESSOR** element shall identify a content format processor. The *Format* attribute of a **PROCESSOR** element shall be present and its value shall specify the format of content data that the content format processor shall be able to consume. The value of the *Format* attribute shall be an IANA registered format name.

The *Revision* attribute of a **PROCESSOR** element may be present and if present, its value shall identify a particular revision of the format identified by the *Format* attribute.

9 Impositioning

9.1 *The <PRINT_LAYOUT> Element*

9.1.1 Model

```
PRINT_LAYOUT    EMPTY
```

9.1.2 Attributes

Refer to the *PPML Impositioning Specification* for more information.

9.1.3 Description

PRINT_LAYOUT is the master element for specifying PPML impositioning instructions.

NOTE The use of **PRINT_LAYOUT** is not recommended in combination with job ticketing as most job ticket formats have provisions for specifying the mapping of pages to sheet surfaces.

NOTE PPML impositioning instructions will not be enhanced and may be deprecated in the future. PODi will work with CIP4 to ensure that the JDF impositioning method provides functionality useful to variable data printing.

9.1.4 Semantics

The syntax and semantics of a **PRINT_LAYOUT** element are governed by the *PPML Impositioning* Specification.

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, or **JOB.**

9.2 The <IMPOSITION> Element

9.2.1 Model

```
IMPOSITION  EMPTY
```

9.2.2 Attributes

Refer to the *PPML Impositioning Specification* for more information.

9.2.3 Description

IMPOSITION is the master element for specifying PPML impositioning templates.

NOTE The use of **IMPOSITION** is not recommended in combination with job ticketing as most job ticket formats have provisions for specifying the mapping of pages to sheet surfaces.

9.2.4 Semantics

The syntax and semantics of an **IMPOSITION** element are governed by the *PPML Impositioning Specification*.

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, or **JOB.**

10 Application-specific Information

10.1 The <PRIVATE_INFO> Element

10.1.1 Model

```
PRIVATE_INFO       (#PCDATA)
```

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, **JOB**, **DOCUMENT,** or **PAGE**.

10.1.2 Attributes

Attribute	Required /Optional	Type	Description
Creator	Required	Identifier	The creator (person, application, system etc) of this element.
Identifier	Optional	Identifier	An arbitrary string identifying what information or feature is provided by the content of this element.
Encoding	Optional	Encoding	Identifies the encoding, if any, used in the content of this element.
CharacterSet	Optional	CharacterSet	Identifies the character set used in the content of this element.

10.1.3 Description

Private Info elements are private; their content is ignored by systems that do not know the meaning of the enclosed data.

10.1.4 Semantics

The contents of the **PRIVATE_INFO** element shall have no effect on the visual appearance of the content defined by the PPML pages. A **PRIVATE_INFO** element shall be processed as if it were an **INTERNAL_DATA** element. Processing of the contents defined by the **PRIVATE_INFO** shall have no effect on the visual appearance of the content defined by the PPML pages.

10.2 The <METADATA> Element

10.2.1 Model

```
    METADATA          (DATUM*)
```

NOTE The use of this element is defined in the models of the following elements: **PPML**, **DOCUMENT_SET**, **JOB**, **DOCUMENT, PAGE** and **OBJECT**.

10.2.2 Attributes

Attribute	Required /Optional	Type	Description
Creator	Required	Identifier	The creator (person, application, system, etc) of this element.
CreationDate	Optional	DateTIme	Time stamp of creation of the metadata.
Identifier	Optional	Identifier	An arbitrary string identifying the information provided by the content of this element.
Target	Optional	Identifier	The target (person,application, system) for whom this metadata is included. default: the target equals the creator.

10.2.3 Description

METADATA elements are used to specify information unrelated to the appearance of the content defined by the PPML dataset. Metadata is useful for annotating **PAGE**, **DOCUMENT**, **JOB**/**DOCUMENT_SET, PPML** and **OBJECT** elements with information about the PPML dataset in which this element appears.

10.2.4 Semantics

The contents of the **METADATA** element shall have no effect on the visual appearance of the content defined by the PPML pages.

10.3 The <DATUM> Element

10.3.1 Model

```
DATUM          (#PCDATA)
```

NOTE The use of this element is defined in the model of the following element: **METADATA**.

10.3.2 Attributes

Attribute	Required /Optional	Type	Description
Key	Required	String	The name for the information defined by this DATUM element.

10.3.3 Description

DATUM elements are used to define a single piece of information that is part of a **METADATA** element. The *Key* attribute identifies the name of that piece of information. The content of the **DATUM** element defines that information.

10.3.4 Semantics

The contents of the **DATUM** element shall have no effect on the visual appearance of the content defined by the PPML pages. The content of a **DATUM** element shall be either text or XML content from a different name space.

Annex A (informative) Conformance specifications

Registering Conformance Subsets

To register a Conformance Subset send a detailed description of the subset and access procedures to PODi at

PODi: the Digital Printing Initiative

1240 Jefferson Road, Rochester, New York 14623, USA

Tel: (585) 239-6014

http://www.podi.org

info@podi.org

Registered Conformance Statements

Graphic Arts (GA): PODi: the Digital Printing Initiative. *Graphic Arts Conformance Subset Version 2.2*. http://www.podi.org

ANSI CGATS.20:2002, "Graphic technology - Variable printing data exchange using PPML and PDF (PPML/VDX)", NPES 2002.

ISO 16612-1:2005(E), Graphic technology-Variable printing data exchange-Part 1: Using PPML 2.1 and PDF 1.4 (PPML/VDX-2005)

PODi does not guarantee suitability of a conformance subset for a specific purpose.

Annex B
(informative)
Revision history

Version 1.0, March 15, 2000

Initial release.

Version 1.01, May 18, 2000:

- **Inside front cover:** modify text and email address related to reader participation.
- 1.2 Organization of this Document: add "and Marks"
- **2.1.4 DTD:** Add reference to the official online version of the PPML DTD.

4.4.3 (attributes of **DOCUMENT** and **Page):** reposition *Labe*l attribute in the table. *(This does not affect functionality.)*

- 5.3.3 Attributes of **`MARK,`** new 5.5.3 Implementation note, new definition of the *Position* attribute.
- **5.7.1 Description of OBJECT element:** add a second paragraph, clarifying intent related to the change in 5.5.3 above.
- 5.7.3 Attributes of **OBJECT:** see 5.3.3 above.
- **5.8.2 Model of `SOURCE`:** add **EXTERNAL_DATA**_ARRAY, consistent with contexts listed in 5.10.3.
- Appendix 3: add SVG.
- **Reference card:** update per the above; document the list of allowed attribute values where appropriate, and show which choice is the default.

Version 1.02, December 14, 2000:

- New features and substantial additions
- Add support for multi-page source files:
- Created two new elements, **SEGMENT_ARRAY** and **SEGMENT_REF**;

- Added **SEGMENT_REF** to the model for **MARK**, and added **SEGMENT_ARRAY** to the model for **PPML**, **Job** [**Document Set** in version 2.x], **Document**, and **Page**.

- **Illustrations** of how PPML content objects are created and placed on a page:

- Added new section ***5.19 Definition of PPML Extent Boxes***

- Added section ***5.20 Notes on Transforming, Clipping and Positioning***

- **Imaging model re transparency & overprint:** Modify the following sections regarding the interaction of marks on a page:
  5.2 A **Page** contains **Marks**
  5.3.1 The **MARK** Element – Description
  9.1 Transparency / overprinting

- Additional changes and clarifications

- **2.1.4 DTD:** Add PUBLIC identifier; change statement regarding DTDs stored on the Web.

- **2.2 Non-XML Data:** remove sentence about a possible separate specification regarding transport issues.

- **5.8.1 SOURCE:** Add paragraph regarding non-content data, such as binary previews on Windows EPS files.

- **5.10.3 EXTERNAL_DATA_ARRAY:** Clarify minimum value of `Index` attribute.

- **6.6.3 IMPOSITION *Position* attribute:** Declare that the imposition structure does not include any trim or fold marks, so the marks do not affect position on the sheet,

- **6.8.1 SIGNATURE description:** Explain CELL positioning and rotation

- **6.9 The CELL Element:** Expand description (6.9.1), add rotation example (6.9.7), add "PageOrder <1" case at end of 6.9.5.

- **6.10 The HOR_TRIM_MARKS Element:** Add illustration of position of trim marks; clarify wording of mark suppression in the "touching pages" case.

- **Scope of OCCURRENCE_REF in sheet marks:** State in 6.10.1, 6.11.1, 6.14.1, 6.15.1 that the scope of a sheet mark's Occurrence Ref must be at least as high as the enclosing **IMPOSITION**.

- **6.14.1, HOR_FOLD_MARKS:** clarify suppression of trim marks near fold marks.

- **8.2.3, Attributes of FONT:** Add *Format* attribute. Also, change the *Name* attribute to *FontName* and add a descriptive note about its intent. ("`Name`" in other PPML elements is merely an arbitrary identifying string; in the **FONT** element, it denotes the actual name of the font, e.g. Helvetica-BoldOblique. Also, add *Format* attribute.

- 8.5 **SUPPLIED_RESOURCE**:

- **8.5.1 Description:** stipulate that the resource must be referenced to be used; stipulate that resources can be processed in any order.

- **8.5.3 Attributes:** add required *ResourceName* attribute; clarify that the *Name* attribute is for use in **SUPPLIED_RESOURCE_REF**; *Type* attribute has only two possible values (Font or ProcSet); add definition of ProcSet.

Version 1.5, May 31, 2001:

- New features and substantial additions

- Conformance subsets

- Add new Chapter 10, Conformance Subsets, particularly Section 10.2, Graphic Arts subset, with full definition of file formats and their constraints.

- Add new **CONFORMANCE** element (Section 4.7) and *ResourcesIncluded* attribute on **PPML**.

- **Page Dimension information:** For non-imposing Consumers (see below), add new **PAGE_DESIGN** element (section 4.6); add corresponding text in **PAGE_LAYOUT;** deprecate the use of the *Dimensions* attribute on *DOCUMENT* and *PAGE*.

- Additional changes and clarifications

- **Imposing and non-imposing Consumers:** clarify the term “imposition” as used in this specification (section 6.1.1) and update the boxed note in Section 6.1 regarding what features a Consumer may or may not support; add *SheetLayoutIncluded* attribute on **PPML**.

- **Enhanced REPEAT functionality** for imposing Consumers: in the *PageOrder* attribute of **CELL**, change the counter ***s*** to refer to sheets (not signatures) and add document counter ***d***.

Errata in Version 1.5

- Copyright date on table of contents page should be 2001

- DTD information in section 2.1.4 should refer to version 1.5

- Model for **PPML**:

- Section 4.2.2 should say CONFORMANCE?, not CONFORMANCE_LEVEL?

- Reference Card’s model should say CONFORMANCE? not CONFORMANCE

Version 2.0, April 4, 2002:

- New features and substantial additions

- PPML Architecture
- Add new section 1.2, **PPML Architecture**, detailing and extending the potential uses and applications of future versions of PPML
- Job ticketing
- Create separate spec document *PPML Job Ticketing*
- New section 3.4: Add section for job ticketing in Definitions section
- New sections 4.8 (`TICKET`) and 4.9 (`TICKET_REF`)
- Add *TICKET_REF* to models for **PPML, JOB/DOCUMENT_SET, DOCUMENT, PAGE, MARK, REUSABLE_OBJECTT** and **OCCURRENCE_LIST**
- Add TICKET to model for **PPML**

Change model for **INTERNAL_DATA** to ANY to accept job ticket data as content; change **EXTERNAL_DATA** Description and Context text to include use of **EXTERNAL_DATA** with **TICKET**.

- Refine wording of section 10.2 (Graphic Arts subset) regarding job ticketing
- Add Appendix E, Job Ticketing Formats to this document
- Packaging jobs for transport:
- Add **Appendix D, Packaging** (incorporates Tech Note TN1 into the specification)
- Schema support
- Rewrite section 2.1.4, “DTD and Schema”
- Created XML schema for PPML 2.0
- Additional changes and clarifications
- Correct all errata from Version 1.5 (see above)
- Improve accuracy of CDATA footnote in section 2.2, NonXML Data
- Deprecate **JOB** in favor of DOCUMENT_SET (section 4.3 and throughout document)
  This change was modified in version 2.1 – see below
- EXTERNAL_DATA, EXTERNAL_DATA_ARRAY, SEGMENT_ARRAY: Add Checksum and ChecksumType attributes
- SUPPLIED_RESOURCE: change model to INTERNAL_DATA | EXTERNAL_DATA; deprecate the Src attribute

- New Appendix F, Embedding Text
- Conformance subset strings (section 4.7): add instructions on how to submit strings to PODi
- Update Appendix A, Acknowledgements
- Add missing Level attribute to table in 4.7.3
- Remove duplicated section 5.20.2 (illustration of transformation and clipping)

Version 2.1, July 31, 2002:

Refinements to Job Ticketing of page content elements

- Remove **TICKET_REF** from between content elements inside PAGE. (See model for Page, section 0.) Thus, the smallest level to which TICKET_REF applies is an entire Page.
- Add **TICKET_SET** (an aggregation of TICKET_REFs, for convenience):
- Define **TICKET_SET** in new section.
- Add to model for **PPML**, **DOCUMENT_SET**, **DOCUMENT**, **PAGE**
- Definition of Reusable Objects:
- IAdd definition of Ticket State
- Add new element **TICKET_STATE** with description of Ticket State concept.
- In Reusable Object definitions, move all **TICKET_**REFs down to the lowest level (the individual **OCCURRENCE**) and thus remove all inheritance of ticket info from individual page content elements. See section 0 for discussion.
- Specifically, remove **TICKET_REF** from model for **REUSABLE_OBJECT** and **OCCURRENCE_**LIST and add **TICKET_STATE** to model for **OCCURRENCE**.

Additional changes

- **JOB** vs. D**OCUMENT_SET**: un-deprecate **JOB** (section 4.3 and throughout). (Keep **DOCUMENT_SET**, but allow **JOB** as a synonym. *(This change avoids invalidating existing datasets that use* **JOB***.)*
- Clerical changes, including: remove "::" notation in "**PAGE_DESIGN**::*TrimBox*" (section 4.6.5); correct the name of job ticketing spec in front matter; fix multiplication signs in attribute tables in chapter 6

Version 2.2, November, 2006:

Reformatted and reorganized to match ANSI CGATS template.

Semantic model definitions incorporated into all element descriptions.

Redefined the Attributes descriptions to define types and updated descriptions.

Deprecated **<TICKET>**, **<TICKET_REF>**, **<TICKET_SET**> and **<TICKET_STATE>**.

Removed Graphic Arts Conformance and created new Graphic Arts Conformance Subset document.

Removed Impositioning and created new Impositioning document.

Removed Packaging Annex and created Packaging document.

Removed Introduction to XML Annex.

Removed Job Ticketing formats Annex.

Added *Class* attributes to **PPML**, **DOCUMENT_SET**, **JOB**, **DOCUMENT** and **PAGE**.

Added the capability to delete global reusable objects and supplied resources.

Added the capability to define global supplied resources.

Added constraints to require the *Dimensions* attribute to match the extent information in content data (if present).

Added **METADATA** to **PPML, JOB, DOCUMENT, PAGE** and **OBJECT** elements.

Added **METADATA** and **DATUM** elements to support storage of meta information inside PPML.

Index