PERSONALIZED PRINT

MARKUP LANGUAGE

PODi

January 2011

Personalized Print Markup Language

Application Notes

PODi: the Digital Printing Initiative

1240 Jefferson Road, Rochester, New York 14623, USA

Tel: (585) 239-6014

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

the Digital Printing initiative

January 2011
PPML Application Notes

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 document. Please send comments via email to ppmlinfo@podi.org.

Contents

PODi i
PODi the Digital Printing Initiative iii
Foreword vi
Personalized Print Markup Language Application Notes 1
Introduction 1
Imaging model 2
Notes on Transforming, Clipping and Positioning 3
Supplied resources 20
Reusable content 24
Deleting the Image Library 26
Dynamically positioning reusable objects 28
Using reusable objects for background imagery 32
Global Supplied Resources 36
Job ticketing 38
PostScript to PPML application notes 39
PDF to PPML application notes 46
Page picking 50
Impositioning 52
Annex A (informative) Revision history 56
Index 57

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. There are several documents that are a part of the PPML library. The purpose of this document is to aid developers of Consumer and Producer solutions.

These application notes were developed by the Technical Working Group during the development of the PPML specification.

NOTE Some of the elements of the PPML 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 PPML or any of the conformance subsets for any specific purpose.

PODi Senior Technologist: Dr. Paul Jones

PODi Director of Technology: James Mekis

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

Personalized Print Markup Language Application Notes

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 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*.

The purpose of PPML is to:

- optimize ripping and 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.

The Application Notes for the Personalized Print Markup Language (PPML) are intended to provide working examples to demonstrate proper PPML coding to achieve desired formatting. They are a companion to, but not a replacement for, the PPML library of functional and conformance specifications.

This document discusses the PPML imaging model and how that model impacts the rendering of objects. Examples of PPML code are included with explanations of how these examples are rendered as the result of transforms, clipping, and positioning. These guidelines should be used to guide producer and consumer implementations.

The sections on Supplied Resources and Reusable Content show example code for identifying and reusing resources and content. Examples show both proper and improper code to help developers focus on best practices.

Also included are sections on PostScript to PPML and PDF to PPML conversion to identify packaging differences for resources. Code examples are provided.

Imaging model

PPML defines how graphical elements are composed into pages, documents and jobs. The imaging model defines the final appearance when multiple graphical elements are placed onto a medium. It specifically defines the appearance of overlapping graphical elements.

In PPML, each graphical element is represented by a PPML object. A page in PPML is defined as a sequence of PPML objects, where each object will be rendered in the order they are defined. The imaging model defines how an object is combined into the content constructed from the objects preceding it in the PPML page definition.

Every PPML object has a binary mask associated with it that defines which areas are transparent and which are opaque. Conceptually, that mask defines the area erased from the page before the object is rendered onto that page. The binary mask also restricts the rendering area for that object.

The background of the content data that defines the object is initially considered transparent. Drawing commands in the content data update the binary mask to erase the background.

In image data, the alpha channel or image mask defines the binary mask for the object. The drawing areas of partially transparent colors set those areas in the mask to opaque. No mixing of colors will occur for partially transparent objects that overlap.

An **alpha channel** is a type of channel used in graphics software for saving selections. Most bitmap editing software allows you to save multiple alpha channels with an image when it is saved in the program's native file format. Any of the alpha channels can be reloaded as a selection or mask at any time, even after closing and reopening the image.

A few of the standard image formats (TIFF and PNG, for example) provide support for an embedded alpha channel which represents up to 256 levels of transparency. Images with an embedded alpha channel can be ported to other applications while retaining transparency as long as the other application also supports alpha channels. Like a mask, the darkest areas of an alpha channel is most transparent, white areas are opaque, and shades of gray represent varying levels of transparency.

http://graphicssoft.about.com

Note that PPML/GA does not allow the use of image data with alpha channels.

The binary mask is defined to be transparent outside the area specified by the dimensions of that object. The dimensions of an object are specified by the *Dimensions* attribute of the **SOURCE** element in the definition of that object.

The binary mask is further modified by any clipping specified for that object. Clipping is introduced by the *ClippingBox* attribute on the **SOURCE** element.

In PPML, an object can be placed onto a page using several steps of transformation and clipping. Each **VIEW** element defines one step of transformation and clipping. The **TRANSFORM** sub-element in the **VIEW** element defines a transformation of the object coordinate system.

Transformations are specified by a matrix that has the same syntax and semantics as transform matrices defined in the PostScript and PDF specifications. After transformation, any clipping defined by a **CLIP_RECT** sub-element is applied. Clipping and transformation apply to both the object itself and its binary mask.

Notes on Transforming, Clipping and Positioning

The following two examples show how to process a simple case of a **MARK** on a PPML page: a single EPS file is transformed and clipped in various ways and then placed on a page. All of the instructions in the first example will be contained in the **MARK** element; the second example shows how the same result could be accomplished using a **REUSABLE_OBJECT** element.

Both examples use the same original EPS file – a few words of text, which fits into a box that is 100 units high and 150 units wide. The result we want to achieve is a part of this EPS file, reduced, cropped, and rotated, as shown at the right.

Source

Desired Result

Self-Contained MARK Example

A self-contained **MARK** has this structure:

The simplest possible **MARK** contains one **VIEW** element and one **OBJECT** element.

- An **OBJECT** is a **VIEW** of a single **SOURCE**.
- Each of the **VIEW**s can contain a **TRANSFORM** and a **CLIP_RECT**.

To process a **MARK**, the Consumer must first process each **OBJECT** inside it. To do that, it first processes the **SOURCE** in the **OBJECT**. Here is the sequence the Consumer must follow:

- Process the **SOURCE**, applying its *ClippingBox*, if any.
- Take the result and transform it using the **TRANSFORM** from the **OBJECT**'s **VIEW.**
- Take the result and clip it using the **CLIP_RECT** from the **OBJECT**'s **VIEW.**

This produces one **OBJECT** that will be contained in the **MARK**.

Now, position the **OBJECT** in the **MARK**'s coordinate space.

Repeat the above steps for each **OBJECT** in the **MARK**.

Now, apply the **MARK**'s **VIEW**:

- Take the set of **OBJECT**s (one or more) and transform it using the **TRANSFORM** from the **MARK**'s **VIEW.**

- Take the result and clip it using the **CLIP_RECT** from the **MARK**'s **VIEW**.

This produces the final piece of page content that will appear on the page. The last step will be to position it on the page, using the **MARK**'s *Position* attribute.

The following PPML fragment achieves our desired result using a self-contained **MARK**:

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Format=”application/postscript”
                Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

A PPML Consumer processes this fragment using the following steps:

1. Read the SOURCE element in the OBJECT

First, the Consumer finds the **SOURCE** element inside the **MARK**:

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

The *ClippingBox* attribute crops the edges of the EPS file, as shown by the dashed line:

☑ *Current coordinate space: the* **SOURCE***.*

This is the content defined by this SOURCE element:

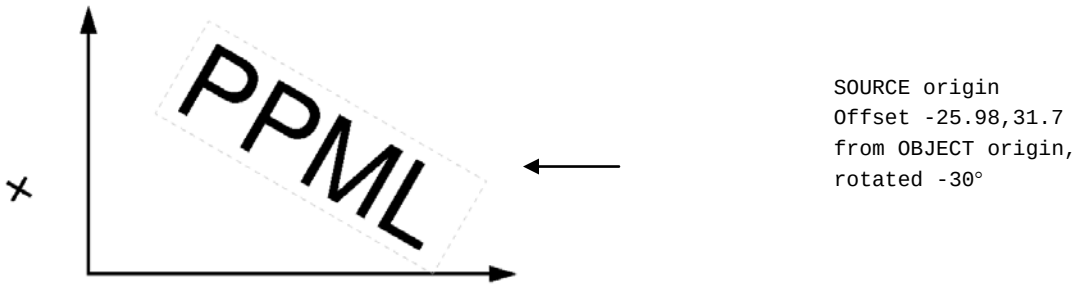
+ SOURCE's origin

2. Completing the OBJECT: Apply VIEW

Next, the Consumer applies the **OBJECT**'s **VIEW**, starting with the **TRANSFORM** element:

```
<MARK Position="30 40">
  <VIEW>
    <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
    <CLIP_RECT Rectangle="0 0 75 75" />
  </VIEW>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
</MARK>
```

The transformation component of this **VIEW** specifies a translation of (-25.98,31.7) and a rotation of –30°.

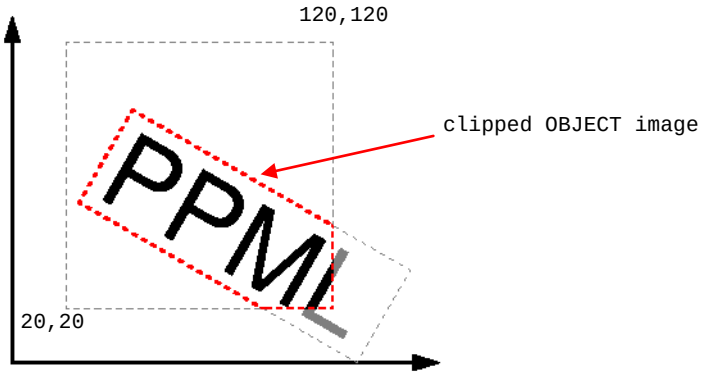


☑ *Current coordinate space: the* **OBJECT***.*

Now process the **OBJECT**'s **CLIP_RECT**.

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECTPosition="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

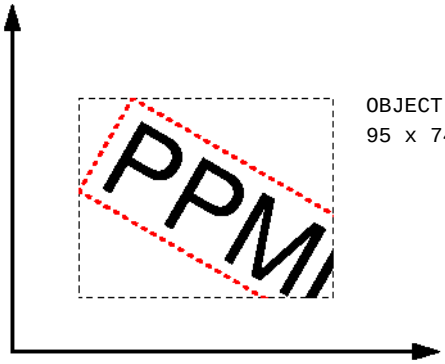
The **CLIP_RECT** (20,20 to 120,120) clips the rotated image like this:



☑ *Current coordinate space: the* **OBJECT***.*

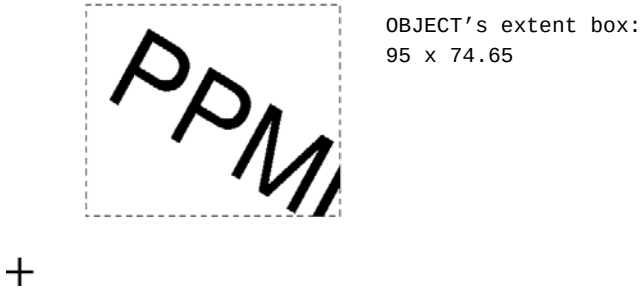
> **Note**
>
> The drawings use color to highlight the clipping area.

Next, determine the extent box of this **OBJECT** element:



☑ *Current coordinate space: the* **OBJECT**.

This is the content that this **OBJECT** element defines:

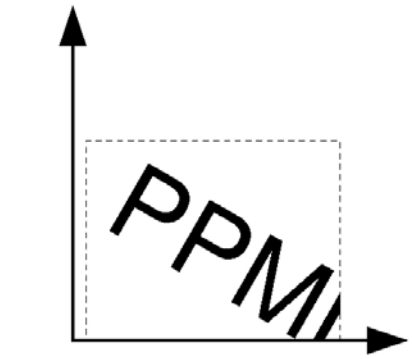


3. Place the OBJECT in the MARK, and apply the MARK's VIEW

A **MARK** can contain several **OBJECT**s, each with its own position. When each **OBJECT** is complete, its origin can be placed anywhere within the coordinates of its enclosing **MARK** element. This is done using the **OBJECT** element's *Position* attribute.

In this example the **MARK** contains only one **OBJECT**, positioned at (-20,-20).

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```



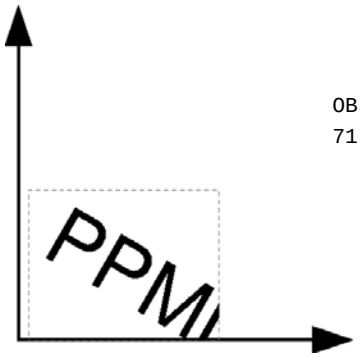
-20,-20 (offset of OBJECT's origin in the MARK)

☑ *Current coordinate space: the **MARK**.*

Next, apply the **MARK**'s **TRANSFORM**: scale the **OBJECT** to 75% of its original size:

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

Result:

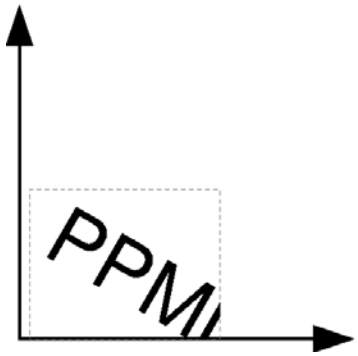


☑ *Current coordinate space: the **MARK**.*

Next, apply the **MARK***'s* **CLIP_RECT***:* in this case, the extra clipping does not influence the result.

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

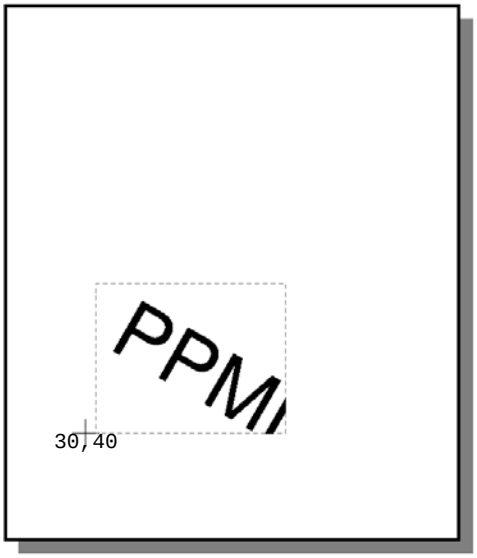

The **MARK**'s content is now complete. The content can now be positioned on the page, as shown below.



4. Position the MARK on the page.

The only remaining step is to process the **MARK** element's *Position* attribute.

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

☑ Current coordinate space: the PAGE.

The entire **MARK** is now complete: the content has been marked onto the page.

```
<MARK Position="30 40">
   <VIEW>
      <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
      <CLIP_RECT Rectangle="0 0 75 75" />
   </VIEW>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
</MARK>
```

The following PostScript code could be placed before the EPS source to produce this result:

```
30 40 translate                                 % MARK position
0 0 75 75 rectclip                              % MARK clipping
[0.75 0 0 0.75 0 0] concat                      % MARK transform
-20 -20 translate                               % OBJECT position
20.0 20.0 100.0 100.0 rectclip                  % OBJECT clipping
[0.866 -0.5 0.5 0.866 -25.98 31.7] concat       % OBJECT transform
30.0 50.0 120.0 40.0 rectclip                   % SOURCE clipping
% insert content of file "ppml.eps" here
```


REUSABLE_OBJECT Example

This example renders the same content as the previous example, but uses a **REUSABLE_OBJECT**.

A **REUSABLE_OBJECT** has this structure:

- The simplest possible **REUSABLE_OBJECT** contains a **VIEW**, one **OBJECT**, and an **OCCURRENCE_LIST** with one **OCCURRENCE**.
- Each **OCCURRENCE** specifies a **VIEW** of all the ***OBJECT***s in this **REUSABLE_OBJECT**.
- A **MARK** can include a particular **OCCURRENCE** of a **REUSABLE_OBJECT** by including an **OCCURRENCE_REF**.
- It only makes sense to use **REUSABLE_OBJECT** if its **OCCURRENCE**s are used in more than one **MARK**; it is probable (but not required) that the PPML Consumer will optimize the **OBJECT** for reuse.

To process a **REUSABLE_OBJECT**, the Consumer must first process each **OBJECT** inside it. To do that, it first processes the **SOURCE** in the **OBJECT**. It is the same sequence as is used for **OBJECT**s within a **MARK**:

- Process the **SOURCE**, applying its *ClippingBox* if any.
- Take the result and transform it using the **TRANSFORM** from the **OBJECT**'s **VIEW.**
- Take the result and clip it using the **CLIP_RECT** from the **OBJECT**'s **VIEW.**

This produces one **OBJECT** that will be contained in the **REUSABLE_OBJECT**.

Now, position the **OBJECT** in the **REUSABLE_OBJECT**'s coordinate space.

Repeat the above for each **OBJECT** in the **REUSABLE_OBJECT**.

Now, apply the **REUSABLE_OBJECT**'s **VIEW**:

- Take the set of (one or more) **OBJECT**s and transform it using the **TRANSFORM** from the **REUSABLE_OBJECT**'s **VIEW.**
- Take the result and clip it using the **CLIP_RECT** from the **REUSABLE_OBJECT**'s **VIEW.**

Now, apply each **OCCURRENCE**'s **VIEW**:

- Take the result and transform it using the **TRANSFORM** from the **OCCURRENCE**'s **VIEW .**
- Take the result and clip it using the **CLIP_RECT** from the **OCCURRENCE**'s **VIEW .**
- Repeat the above for each **OCCURRENCE** in the **OCCURRENCE _LIST.**

This process produces the final piece of page content for each **OCCURRENCE**. They are now ready to be included on a page with an **OCCURRENCE_REF**. The last step will be to position the content on the page, using the **MARK**'s *Position* attribute.

The following PPML fragment achieves our desired result using a **REUSABLE_OBJECT**:

```
<REUSABLE_OBJECT>
   <OBJECT Position="-20 -20">
      <SOURCE Format="application/postscript"
                Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
  <VIEW />
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="example">
         <VIEW>
            <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
            <CLIP_RECT Rectangle="0 0 75 75" />
         </VIEW>
      </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>

<MARK Position="30 40">
   <OCCURRENCE_REF Ref="example" />
</MARK>
```

A PPML Consumer processes this fragment using the following steps.

1. Create the OBJECT specified in the REUSABLE_OBJECT.

Use steps 1 and 2 from the previous example to obtain the **OBJECT** by reading its **SOURCE** and applying its **VIEW**.

```
<REUSABLE_OBJECT>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
  <VIEW />
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="example">
         <VIEW>
            <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
            <CLIP_RECT Rectangle="0 0 75 75" />
         </VIEW>
      </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```


This is the content that this OBJECT element defines**:**

☑ *Current coordinate space: the* **OBJECT**

2. Place the OBJECT, and apply the REUSABLE_OBJECT's and OCCURRENCE's VIEWs.

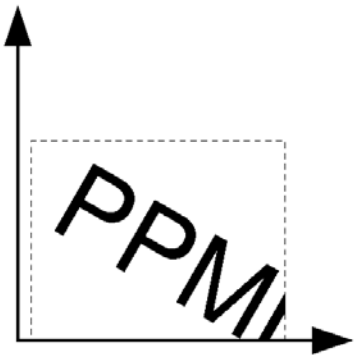
A **REUSABLE_OBJECT** can contain several **OBJECT**s, each with its own position. Thus, when each **OBJECT** is complete, its origin can be placed anywhere within the coordinates of its enclosing **REUSABLE_OBJECT** element. This is done using the **OBJECT** element's *Position* attribute.

In this example, the **OBJECT** is positioned at (-20,-20).

```
<REUSABLE_OBJECT>
   <OBJECT Position="-20 -20">
      <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
         <EXTERNAL_DATA Src="ppml.eps" />
      </SOURCE>
      <VIEW>
         <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
         <CLIP_RECT Rectangle="20 20 120 120" />
      </VIEW>
   </OBJECT>
  <VIEW />
   <OCCURRENCE_LIST>
      <OCCURRENCE Name="example">
         <VIEW>
            <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
            <CLIP_RECT Rectangle="0 0 75 75" />
         </VIEW>
      </OCCURRENCE>
   </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```



```
-20,-20 (offset of OBJECT's origin in the REUSABLE_OBJECT)
```



☑ *Current coordinate space: the **REUSABLE_OBJECT**.*

Next, apply the **REUSABLE_OBJECT**'s **VIEW**: transform and clip the **OBJECT** as specified. In this example, the **REUSABLE_OBJECT**'s **VIEW** is empty and no processing is required.

```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
  <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
        <CLIP_RECT Rectangle="0 0 75 75" />
      </VIEW>
    </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

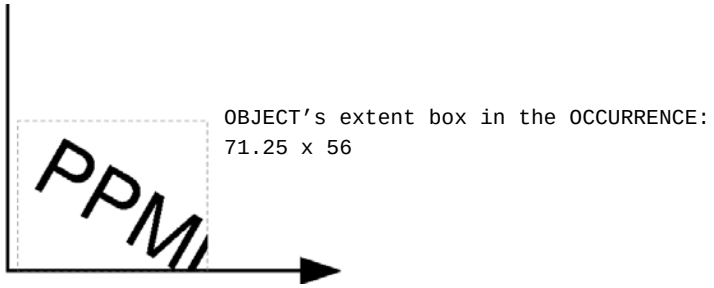
Next, apply the **OCCURRENCE**'s **TRANSFORM**: scale the **OBJECT** to 75% of its current size:

```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
  <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
```



```
          <CLIP_RECT Rectangle="0 0 75 75" />
        </VIEW>
      </OCCURRENCE>
    </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

Result:



☑ *Current coordinate space: the* **OCCURRENCE***.*

Next, apply the **OCCURRENCE**'s **CLIP_RECT**: in this case, the extra clipping has no effect.

```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
  <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
        <CLIP_RECT Rectangle="0 0 75 75" />
      </VIEW>
    </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```


The **OCCURRENCE**'s content is now complete**.**

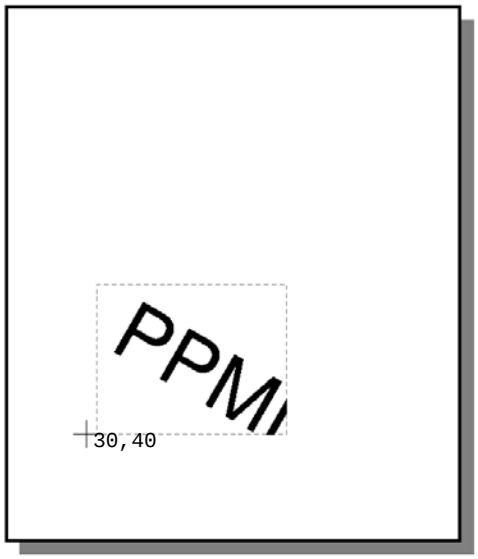
```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
  <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
        <CLIP_RECT Rectangle="0 0 75 75" />
      </VIEW>
    </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

3. Position the OCCURRENCE on the PAGE.

The only remaining step is to apply the **MARK** element's *Position* attribute to the **OCCURRENCE** created in step 2:

```
<REUSABLE_OBJECT>
  <OBJECT Position="-20 -20">
    <SOURCE Dimensions="150 100" ClippingBox="30 50 160 90">
      <EXTERNAL_DATA Src="ppml.eps" />
    </SOURCE>
    <VIEW>
      <TRANSFORM Matrix="0.866 -0.5 0.5 0.866 -25.98 31.7" />
      <CLIP_RECT Rectangle="20 20 120 120" />
    </VIEW>
  </OBJECT>
  <VIEW />
  <OCCURRENCE_LIST>
    <OCCURRENCE Name="example">
      <VIEW>
        <TRANSFORM Matrix="0.75 0 0 0.75 0 0" />
        <CLIP_RECT Rectangle="0 0 75 75" />
      </VIEW>
    </OCCURRENCE>
  </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

```
<MARK Position="30 40">
  <OCCURRENCE_REF Ref="example" />
<MARK/>
```



☑ Current coordinate space: the PAGE.

The entire **MARK** is now complete: the content has been marked onto the page.

Supplied resources

When using PostScript content data in a PPML dataset, each PostScript file typically contains a similar prologue and uses the same set of fonts. To reduce the size and processing time of those PostScript files, PPML includes the ability to allow fonts and PostScript ProcSets to be specified once such that they may be used throughout the PPML dataset.

Typically the prologue of a PostScript file can be converted into a PostScript ProcSet creating PostScript “snippets” that can be used throughout a PPML dataset without having to include that PostScript prologue in each of the PostScript “snippets”. Similarly, fonts supplied in PPML need not be included in each PostScript “snippet” that uses those fonts.

PPML uses the **SUPPLIED_RESOURCE** element to define a font or PostScript ProcSet for reuse. The **SUPPLIED_RESOURCE_REF** element is used to define which content data may depend on a supplied resource. The **SUPPLIED_RESOURCE_REF** element ensures that the supplied resource is in memory for those pieces of content data requiring that supplied resource. If a supplied resource is not a required resource for a given piece of content data that supplied resource may not be referenced in that content data.

PPML uses the **SUPPLIED_RESOURCE** element to define a font or PostScript ProcSet for reuse.

A PPML Consumer may optimize the use of supplied resources by making the supplied resource persistent at the point of definition. A Consumer should be prepared to handle multiple resources with the same name (defined in different PPML scopes). A PPML Consumer must ensure that a supplied resource is available when the supplied resource is a required resource for a piece of content data. Note that a PPML Consumer is not required to disable access to a supplied resource for content data that do not require that supplied resource. A PPML Consumer must however ensure that the correct version of a supplied resource is available to content data. Supplied resources take precedence over any resource installed in a PPML Consumer.

In a level 1 PPML/GA dataset no content data may depend on resources installed in a PPML Consumer, therefore all resources needed must be supplied in that PPML dataset. In a level 2 PPML/GA dataset references to resources installed in a PPML Consumer are possible. In such cases, that dataset may not print correctly if processed by a different PPML Consumer.

PPML supports the definition of supplied resources which depend on other supplied resources. The required resources at the point of definition of a supplied resource must be in memory before that supplied resource is loaded into memory. When that supplied resource is referenced, those required resources may no longer be required. It is up to the PPML Producer to ensure that at the point of reference of a supplied resource the resources that still need to be in memory are required resources for the content data referencing that supplied resource.

Example:

```
ProcSet X:
   1 dict begin
      /myfunc { /Y /ProcSet findresource /dosomething get exec } def
   currentdict end
   /X exch /ProcSet defineresource
ProcSet Y:
   1 dict begin
      /dosomething { (hello) show } def
   currentdict end
   /Y exch /ProcSet defineresource
ProcSet Z:
   1 dict begin
      /myfunc /Y /ProcSet findresource /dosomething get def
   currentdict end
   /Z exch /ProcSet defineresource
```

In this example, X is dependent on Y when X is used. Y need not be in memory when X is loaded into memory. Z however is only dependent on Y when Z is loaded into memory. Y need not be in memory when Z is used. Therefore, Y must be required resource at the point of definition of Z and Y must be a required resource whenever X is a required resource.

For Y to be a required resource for Z, Y must be defined in a different scope from Z and made a required resource for the scope in which Z is defined. Y and Z cannot be defined in the same scope, as required resources defined in a scope are *not* required resources for supplied resource defined in the same scope. Therefore, the following example will *not* work for Y and Z but will work for Y and X:

```
   <PAGE>
   <SUPPLIED_RESOURCES>
      <SUPPLIED_RESOURCE Type=″ProcSet″ Name=″Y″ ResourceName=″Y″
Format=″application/postscript″>
         <EXTERNAL_DATA Src=″Y.ps″/>
      </SUPPLIED_RESOURCE>
      <SUPPLIED_RESOURCE Type=″ProcSet″ Name=″Z″ ResourceName=″Z″
Format=″application/postscript″>
         <EXTERNAL_DATA Src=″Z.ps″/>
      </SUPPLIED_RESOURCE>
   </SUPPLIED_RESOURCES>
   <REQUIRED_RESOURCES>
      <SUPPLIED_RESOURCE_REF Ref=″Y″/>
      <SUPPLIED_RESOURCE_REF Ref=″Z″/>
   </REQUIRED_RESOURCES>
   <MARK Position=″0 0″>
```



```
        :
        reference Z
        :
    <MARK>
```

Some developers try to streamline the process with the following solution, ***which is not supported***:

```
    <PAGE>
    <SUPPLIED_RESOURCES>
        <SUPPLIED_RESOURCE Type=″ProcSet″ Name=″Y″ ResourceName=″Y″
Format=″application/postscript″>
            <EXTERNAL_DATA Src=″Y.ps″/>
        </SUPPLIED_RESOURCE>
    </SUPPLIED_RESOURCES>
    <REQUIRED_RESOURCES>
        <SUPPLIED_RESOURCE_REF Ref=″Y″/>
    </REQUIRED_RESOURCES>
    <SUPPLIED_RESOURCES>
        <SUPPLIED_RESOURCE Type=″ProcSet″ Name=″Z″ ResourceName=″Z″
Format=″application/postscript″>
            <EXTERNAL_DATA Src=″Z.ps″/>
        </SUPPLIED_RESOURCE>
    </SUPPLIED_RESOURCES>
    <REQUIRED_RESOURCES>
        <SUPPLIED_RESOURCE_REF Ref=″Z″/>
    </REQUIRED_RESOURCES>
    <MARK Position=″0 0″>
        :
        reference Z
        :
    <MARK>
    </PAGE>
```

The above example shows invalid PPML as the PPML syntax rules do not allow multiple **SUPPLIED_RESOURCES** and **REQUIRED_RESOURCES** elements within a single scope. The correct method for creating a dependency of Z on Y therefore is:

```
    <DOCUMENT>
        <SUPPLIED_RESOURCES>
        <SUPPLIED_RESOURCE Type=″ProcSet″ Name=″Y″ ResourceName=″Y″
Format=″application/postscript″>
```



```
            <EXTERNAL_DATA Src=”Y.ps”/>
         </SUPPLIED_RESOURCE>
      </SUPPLIED_RESOURCES>
      <REQUIRED_RESOURCES>
         <SUPPLIED_RESOURCE_REF Ref=”Y”/>
      </REQUIRED_RESOURCES>
      <PAGE>
      <SUPPLIED_RESOURCES>
         <SUPPLIED_RESOURCE Type=”ProcSet” Name=”Z” ResourceName=”Z”
            Format=”application/postscript”>
            <EXTERNAL_DATA Src=”Z.ps”/>
         </SUPPLIED_RESOURCE>
      </SUPPLIED_RESOURCES>
      <REQUIRED_RESOURCES>
         <SUPPLIED_RESOURCE_REF Ref=”Z”/>
      </REQUIRED_RESOURCES>
      <MARK Position=”0 0”>
         :
         reference Z
         :
      <MARK>
      </PAGE>
   </DOCUMENT>
```


Reusable content

In many variable data print (VDP) jobs certain objects are used multiple times (such as logos, signatures and page backgrounds). These objects can be considered *reusable*. Establishing objects as reusable may enable the Consumer to optimize the rendering of those objects on a page. Examples of such optimizations include pre-rasterization, pre-ripping and caching.

Reusable object definitions are *scoped*. Scoping a reusable object restricts references to that reusable object to a well-defined part of the PPML hierarchy. Scoping informs the Consumer when a reusable object can no longer be referenced allowing the definition of that reusable object to be discarded.

To support PPML Producers that generate PPML in a stream, PPML allows *scope promotion*. Scope promotion allows the definition of a reusable object to extend beyond the scope in which the definition occurs. Scope promotion is useful if a Producer does not know in advance if a reusable object will be needed or when that reusable object will be used. If the Producer detects the need for a reusable object, its definition can only be made in the scope currently being defined in the PPML stream. If the Producer expects that the reusable object will be needed after the end of the scope being defined, scope promotion may be used to avoid additional definitions of that reusable object.

The concept of an *occurrence* was introduced to inform a Consumer of how a reusable object will be placed on to a page. Knowing the rotation and scaling of the reusable object up front simplifies the process of rendering a reusable object on a page in an optimized manner. By only allowing an occurrence to be positioned (without additional scaling or rotation) onto a page, pre-rasterization of the reusable object becomes a viable optimization method.

To avoid repeated definitions of frequently used reusable objects and resources in different PPML datasets the concept of global scope was introduced. Using scope promotion, an occurrence or resource can be defined in global scope. This allows an occurrence or resource definition to be referenced without having to provide that definition in PPML again. The submitter of a PPML dataset containing such references will have to ensure that the appropriate definitions are available to the Consumer.

The use of global definitions across multiple PPML datasets may introduce dataset interdependencies with respect to the order in which the Consumer must process those datasets. A Consumer is not required to keep global definitions indefinitely and may not have the ability to do so. The Producer must coordinate with the Consumer to ensure that such workflows are reliable. Global occurrences can be useful in facilitating a set of PPML datasets using consistent versions of those occurrences (such as a large image library), where the generator of those PPML datasets need not be able to recreate the definition of those occurrences.

A Consumer is not required to keep global definitions indefinitely and may not have the ability to do so.

A Producer must take care when redefining a global occurrence with very different content. It is recommended that a Producer issue new identifiers instead of re-using existing identifiers. Existing global definitions may be deleted by providing a new definition with the *Overwrite* attribute set to "*Delete*".

The concept of *environment* was introduced to avoid problems with similarly named global reusable object definitions by different customers. The environment is used to separate the global reusable

object definitions based on a named context. Often a unique name, such as a registered domain name or the company name, is used to ensure that global reusable object definitions are not accidentally replaced. Producers may wish to use a GUID (*Global Unique Identifier*) or UUID (Universal Unique Identifier) to provide a unique name for each occurrence.

The following PPML dataset defines a reusable global image library of 100 images:

```
<PPML>
   <CONFORMANCE SubSet="GA" Level="2"/>
   <REUSABLE_OBJECT>
      <OBJECT Position="0 0">
         <SOURCE Format="image/tiff" Dimensions="72 72">
            <EXTERNAL_DATA Src="image001.tiff"/>
         </SOURCE>
      </OBJECT>
      <OCCURRENCES>
         <OCCURRENCE Name="image001" Environment="www.podi.org"
Scope="Global"/>
      </OCCURRENCES>
   </REUSABLE_OBJECT>
         :
   <REUSABLE_OBJECT>
      <OBJECT Position="0 0">
         <SOURCE Format="image/tiff" Dimensions="72 72">
            <EXTERNAL_DATA Src="image100.tiff"/>
         </SOURCE>
      </OBJECT>
      <OCCURRENCES>
         <OCCURRENCE Name="image100" Environment="www.podi.org"
Scope="Global"/>
      </OCCURRENCES>
   </REUSABLE_OBJECT>
</PPML>
```

Those images may be used in a separate PPML dataset as follows:

```
<PPML>
   <CONFORMANCE SubSet="GA" Level="2"/>
   <DOCUMENT_SET>
      <DOCUMENT>
         <PAGE>
            <MARK Position="500 700">
               <OCCURRENCE_REF Ref="image053"
                  Environment="www.podi.org"/>
            </MARK>
```



```
            <MARK Position="50 800">
               variable content defined here
            </MARK>
         </PAGE>
         <PAGE>
            <MARK Position="500 700">
               <OCCURRENCE_REF Ref="image075"
                    Environment="www.podi.org"/>
            </MARK>
            <MARK Position="50 800">
               variable content defined here
            </MARK>
         </PAGE>
      </DOCUMENT>
   </DOCUMENT_SET>
</PPML>
```

Deleting the Image Library

When the image library is no longer needed, the occurrences may be deleted by sending the following PPML dataset:

```
<PPML>
   <CONFORMANCE SubSet="GA" Level="2"/>
   <REUSABLE_OBJECT>
      <OBJECT Position="0 0">
         <SOURCE Format="image/tiff" Dimensions="72 72">
            <INTERNAL_DATA/>
         </SOURCE>
      </OBJECT>
      <OCCURRENCES>
         <OCCURRENCE Name="image001" Environment="www.podi.org" Scope="Global"
Overwrite="Delete"/>
      </OCCURRENCES>
   </REUSABLE_OBJECT>
         :
   <REUSABLE_OBJECT>
      <OBJECT Position="0 0">
         <SOURCE Format="image/tiff" Dimensions="72 72">
            <INTERNAL_DATA/>
         </SOURCE>
      </OBJECT>
      <OCCURRENCES>
         <OCCURRENCE Name="image100" Environment="www.podi.org" Scope="Global"
Overwrite="Delete"/>
      </OCCURRENCES>
   </REUSABLE_OBJECT>
```



```
</PPML>
```

Here, empty **INTERNAL_DATA** elements are used to define a reusable object with empty content. By setting the value of the *Overwrite* attribute on the **OCCURRENCE** element to "*Delete*", the source data in the reusable object definition is ignored and the occurrence can no longer be referenced. Alternatively, it is also possible to supply the original reusable object definition and just set the *Overwrite* attribute to "*Delete*" for each **OCCURRENCE** element that is no longer needed.

It is also possible to include the above **REUSABLE_OBJECT** definitions into the last PPML dataset that uses those reusable objects. These **REUSABLE_OBJECT** definitions must be included at the end of the PPML dataset to ensure that the reusable objects are still available in the rest of the PPML dataset. The following example illustrates this:

```
<PPML>
    <CONFORMANCE SubSet="GA" Level="2"/>
    <DOCUMENT_SET>
        <DOCUMENT>
            <PAGE>
                <MARK Position="500 700">
                    <OCCURRENCE_REF Ref="image053" Environment="www.podi.org"/>
                </MARK>
                <MARK Position="50 800">
                    variable content defined here
                </MARK>
            </PAGE>
            <PAGE>
                <MARK Position="500 700">
                    <OCCURRENCE_REF Ref="image075" Environment="www.podi.org"/>
                </MARK>
                <MARK Position="50 800">
                    variable content defined here
                </MARK>
            </PAGE>
        </DOCUMENT>
    </DOCUMENT_SET>
    <REUSABLE_OBJECT>
        <OBJECT Position="0 0">
            <SOURCE Format="image/tiff" Dimensions="72 72">
                <INTERNAL_DATA/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCES>
            <OCCURRENCE Name="image001" Environment="www.podi.org"
```

```
Scope="Global" Overwrite="Delete"/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
           :
    <REUSABLE_OBJECT>
        <OBJECT Position="0 0">
           <SOURCE Format="image/tiff" Dimensions="72 72">
              <INTERNAL_DATA/>
           </SOURCE>
        </OBJECT>
        <OCCURRENCES>
           <OCCURRENCE Name="image100" Environment="www.podi.org"
Scope="Global" Overwrite="Delete"/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
</PPML>
```

Dynamically positioning reusable objects

In many documents, certain parts of the document may be static but must be positioned dynamically to accommodate other objects that may have varying heights. For instance: a letter may have a paragraph that refers to the name and address of the recipient causing that paragraph to be 3 or 4 lines depending on the length of the name and address. Any paragraphs that follow must be moved up or down accordingly. If these paragraphs are completely static and the text of those paragraphs need not flow around other objects (such as images) those paragraphs can be optimized as a reusable object in PPML.

Example:

```
<PPML>
    <CONFORMANCE SubSet="GA" Level="1"/>
    <REUSABLE_OBJECT>
        <OBJECT Position="0 0">
            <SOURCE Format="application/postscript"
                Dimensions="75 400">
                <EXTERNAL_DATA Src="paragraph1.ps"/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCES>
            <OCCURRENCE Name="paragraph1"/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
```

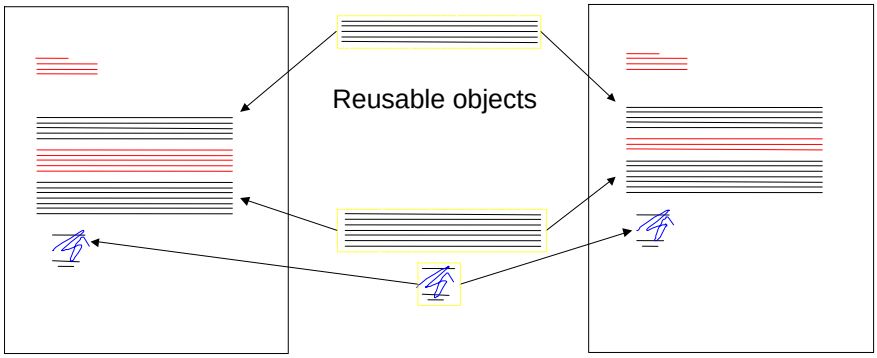
```
<REUSABLE_OBJECT>
   <OBJECT Position=”0 0”>
      <SOURCE Format=”application/postscript”
         Dimensions=”100 400”>
         <EXTERNAL_DATA Src=”paragraph3.ps”/>
      </SOURCE>
   </OBJECT>
   <OCCURRENCES>
      <OCCURRENCE Name=”paragraph3”/>
   </OCCURRENCES>
</REUSABLE_OBJECT>
<REUSABLE_OBJECT>
   <OBJECT Position=”0 0”>
      <SOURCE Format=”image/tiff” Dimensions=”72 72”>
         <INTERNAL_DATA/>
      </SOURCE>
   </OBJECT>
   <OCCURRENCES>
      <OCCURRENCE Name=”signature”/>
   </OCCURRENCES>
</REUSABLE_OBJECT>
<DOCUMENT_SET>
   <DOCUMENT>
   <PAGE>
      <MARK Position=”30 700”>
         <OBJECT Position=”0 0”>
            <SOURCE Format=”application/postscript”
                Dimensions=”200 70”>
               <INTERNAL_DATA>
                  ... address block recipient 1...
               </INTERNAL_DATA>
            </SOURCE>
         </OBJECT>
      </MARK>
      <MARK Position=”30 500”>
         <OBJECT Position=”0 0”>
            <SOURCE Format=”application/postscript”
                Dimensions=”200 70”>
               <INTERNAL_DATA>
```



```
                ... greeting recipient 1 ...
            </INTERNAL_DATA>
        </SOURCE>
    </OBJECT>
</MARK>
<MARK Position="30 488">
    <OCCURRENCE_REF Ref="paragraph1"/>
</MARK>
<MARK Position="30 413">
    <OBJECT Position="0 0">
        <SOURCE Format="application/postscript"
            Dimensions="400 35">
            <INTERNAL_DATA>
                ... personalized paragraph ...
                ... 3 lines ...
            </INTERNAL_DATA>
        </SOURCE>
    </OBJECT>
</MARK>
<MARK Position="30 388">
    <OCCURRENCE_REF Ref="paragraph3"/>
</MARK>
<MARK Position="30 288">
    <OCCURRENCE_REF Ref="signature"/>
</MARK>
</PAGE>
</DOCUMENT>
    :
<DOCUMENT>
<PAGE>
    <MARK Position="30 700">
        <OBJECT Position="0 0">
            <SOURCE Format="application/postscript"
                Dimensions="200 70">
                <INTERNAL_DATA>
                    ... address block recipient n...
                </INTERNAL_DATA>
            </SOURCE>
        </OBJECT>
```



```
            </MARK>
            <MARK Position="30 500">
               <OBJECT Position="0 0">
                  <SOURCE Format="application/postscript"
                       Dimensions="200 70">
                     <INTERNAL_DATA>
                        ... greeting recipient n...
                     </INTERNAL_DATA>
                  </SOURCE>
               </OBJECT>
            </MARK>
            <MARK Position="30 488">
               <OCCURRENCE_REF Ref="paragraph1"/>
            </MARK>
            <MARK Position="30 413">
               <OBJECT Position="0 0">
                  <SOURCE Format="application/postscript"
                       Dimensions="400 47">
                     <INTERNAL_DATA>
                        ... personalized paragraph ...
                        ... 4 lines ...
                     </INTERNAL_DATA>
                  </SOURCE>
               </OBJECT>
            </MARK>
            <MARK Position="30 376">
               <OCCURRENCE_REF Ref="paragraph3"/>
            </MARK>
            <MARK Position="30 276">
               <OCCURRENCE_REF Ref="signature"/>
            </MARK>
         </PAGE>
         </DOCUMENT>
      </DOCUMENT_SET>
   </PPML>
```

Using reusable objects for background imagery

In direct marketing applications, the variable data is often overlaid onto a full-color background image. Using reusable objects for background images can greatly improve RIPping and printing performance.

The following PPML shows how a reusable object may be used as a background in our previous example:

```
<PPML>
    <CONFORMANCE SubSet="GA" Level="1"/>
    <REUSABLE_OBJECT>
        <OBJECT Position="0 0">
            <SOURCE Format="application/pdf"
                Dimensions="595 842">
                <EXTERNAL_DATA Src="background.pdf"/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCES>
            <OCCURRENCE Name="background"/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
    <REUSABLE_OBJECT>
        <OBJECT Position="0 0">
            <SOURCE Format="application/postscript" Dimensions="100 400">
                <EXTERNAL_DATA Src="paragraph3.ps"/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCES>
            <OCCURRENCE Name="paragraph3"/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
    <REUSABLE_OBJECT>
```



```
        <OBJECT Position="0 0">
            <SOURCE Format="image/tiff" Dimensions="72 72">
                <INTERNAL_DATA/>
            </SOURCE>
        </OBJECT>
        <OCCURRENCES>
            <OCCURRENCE Name="signature"/>
        </OCCURRENCES>
    </REUSABLE_OBJECT>
    <DOCUMENT_SET>
        <DOCUMENT>
        <PAGE>
            <MARK Position="0 0">
                <OCCURRENCE_REF Ref="background"/>
            </MARK>
            <MARK Position="30 700">
                <OBJECT Position="0 0">
                    <SOURCE Format="application/postscript"
                         Dimensions="200 70">
                        <INTERNAL_DATA>
                            ... address block recipient 1...
                        </INTERNAL_DATA>
                    </SOURCE>
                </OBJECT>
            </MARK>
            <MARK Position="30 500">
                <OBJECT Position="0 0">
                    <SOURCE Format="application/postscript"
                         Dimensions="200 70">
                        <INTERNAL_DATA>
                            ... greeting recipient 1 ...
                        </INTERNAL_DATA>
                    </SOURCE>
                </OBJECT>
            </MARK>
            <MARK Position="30 488">
                <OCCURRENCE_REF Ref="paragraph1"/>
            </MARK>
            <MARK Position="30 413">
```



```
            <OBJECT Position="0 0">
               <SOURCE Format="application/postscript"
                    Dimensions="400 35">
                  <INTERNAL_DATA>
                     ... personalized paragraph ...
                     ... 3 lines ...
                  </INTERNAL_DATA>
               </SOURCE>
            </OBJECT>
         </MARK>
         <MARK Position="30 388">
            <OCCURRENCE_REF Ref="paragraph3"/>
         </MARK>
         <MARK Position="30 288">
            <OCCURRENCE_REF Ref="signature"/>
         </MARK>
      </PAGE>
      </DOCUMENT>
         :
      <DOCUMENT>
      <PAGE>
         <MARK Position="0 0">
            <OCCURRENCE_REF Ref="background"/>
         </MARK>
         <MARK Position="30 700">
            <OBJECT Position="0 0">
               <SOURCE Format="application/postscript"
                    Dimensions="200 70">
                  <INTERNAL_DATA>
                     ... address block recipient n...
                  </INTERNAL_DATA>
               </SOURCE>
            </OBJECT>
         </MARK>
         <MARK Position="30 500">
            <OBJECT Position="0 0">
               <SOURCE Format="application/postscript"
                    Dimensions="200 70">
                  <INTERNAL_DATA>
```



```
                    ... greeting recipient n...
                </INTERNAL_DATA>
            </SOURCE>
        </OBJECT>
    </MARK>
    <MARK Position="30 488">
        <OCCURRENCE_REF Ref="paragraph1"/>
    </MARK>
    <MARK Position="30 413">
        <OBJECT Position="0 0">
            <SOURCE Format="application/postscript"
                Dimensions="400 47">
                <INTERNAL_DATA>
                    ... personalized paragraph ...
                    ... 4 lines ...
                </INTERNAL_DATA>
            </SOURCE>
        </OBJECT>
    </MARK>
    <MARK Position="30 376">
        <OCCURRENCE_REF Ref="paragraph3"/>
    </MARK>
    <MARK Position="30 276">
        <OCCURRENCE_REF Ref="signature"/>
    </MARK>
  </PAGE>
  </DOCUMENT>
 </DOCUMENT_SET>
</PPML>
```

Note that here we add a reusable object before the other content placed on the page. Similarly, overlays such as “DRAFT” or “TEST” can be added using a reusable object placed after all the other content is placed on the page.

Global Supplied Resources

Like global reusable objects, Fonts and PostScript ProcSets can be downloaded into a PPML Consumer so that their definition may be omitted in other PPML datasets.

The following PPML dataset defines a global font:

```
<PPML>
    <CONFORMANCE SubSet="GA" Level="2"/>
    <SUPPLIED_RESOURCES>
        <SUPPLIED_RESOURCE Name="Lucinda-Grande"
            ResourceName="LucindaGrande" Type="Font" Scope="Global"
            Environment="www.podi.org" Format="application/postscript">
            <EXTERNAL_DATA Src="lucindagrande.ps"/>
        </SUPPLIED_RESOURCE>
    </SUPPLIED_RESOURCES>
</PPML>
```

This global font may be referenced in other PPML datasets as follows:

```
<PPML>
    <CONFORMANCE SubSet="GA" Level="2"/>
    <REQUIRED_RESOURCES>
        <SUPPLIED_RESOURCE_REF Name="Lucinda-Grande"
             Environment="www.podi.org"/>
    </REQUIRED_RESOURCES>
    <DOCUMENT_SET>
        <PAGE>
            <MARK Position="0 0">
                <OBJECT Position="0 0 ">
                    <SOURCE Format="application/postscript"
                         Dimensions="100 40">
                        <INTERNAL_DATA>
                            (LucindaGrande) findfont 20 scalefont
                            0 5 moveto
                            (Hello world) show
                        </INTERNAL_DATA>
                    </SOURCE>
                </OBJECT>
            </MARK>
        </PAGE>
    </DOCUMENT_SET>
```



```
</PPML>
```

When the global font is no longer needed one can delete the global supplied resource by redefining the supplied resource with the *Overwrite* attribute set to “Delete”:

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”2”/>
    <SUPPLIED_RESOURCES>
        <SUPPLIED_RESOURCE Name=”Lucinda-Grande”
            ResourceName=”LucindaGrande” Type=”Font” Scope=”Global”
            Environment=”www.podi.org” Overwrite=”Delete”
            Format=”application/postscript”>
            <EXTERNAL_DATA Src=”lucindagrande.ps”/>
        </SUPPLIED_RESOURCE>
    </SUPPLIED_RESOURCES>
</PPML>
```

NOTE It is not necessary to provide the original definition when deleting a global supplied resource; an empty **INTERNAL_DATA** element may be used instead (see also **Deleting the Image Library** on Page 26).

Job ticketing

The use of **TICKET** and **TICKET_REF** for associating job ticket parameters with PPML pages is discouraged. Instead, a JDF job ticket should be used that refers to the PPML dataset as explained in the DPT 2.2 Application Notes.

The use of **TICKET_REF** with JDF is discouraged as **TICKET_REF** relies on the use of the resource update concept, which has been deprecated by CIP4 in JDF 1.3.

PostScript to PPML application notes

A typical PostScript file uses a prolog to define procedures used in the descriptions of the pages. In addition to the prolog some PostScript files embed font definitions. Some PostScript files also use PostScript forms for reusable content.

To convert such PostScript files to a PPML dataset, the procsets in the prolog, the embedded fonts, the forms and the page descriptions need to be packaged into separate entities.

A typical prolog will either create entries in `userdict` or will create separate dictionaries with those entries in them.

Example 1:

```
%%BeginProlog
/myfunc1 { ... } bind def
   :
/myfunc-n { ... } bind def
%%EndProlog
```

Example 2:

```
%%BeginProlog
/mydict 10 dict def
mydict begin
/myfunc1 { ... } bind def
   :
/myfunc-n { ... } bind def
end
%%EndProlog
```

In either case, a PostScript ProcSet should be defined that contains all the entries defined in the prolog. Such a ProcSet first creates a dictionary to contain all the entries; then that dictionary is defined as a named PostScript ProcSet using `defineresource`.

Example 1 can be transformed into a PostScript ProcSet named “myprocset” as follows:

```
<<
/myfunc1 { ... }
   :
/myfunc-n { ... }
>>
   (myprocset) exch /ProcSet defineresource pop
```

NOTE If multiple ProcSets are being used, their interdependencies must be made explicit in PPML. The order of definition of ProcSets in PPML may not be the same order in which ProcSets are loaded into VM (virtual memory). See also the notes on

Global Supplied Resources on Page 36.

Each page description must be packaged as a separate PostScript fragment that does not depend on the execution of other pages. The `showpage` operator need not be included.

Each of the page descriptions that used the functions defined in the original prolog will need some additional PostScript code to access the named ProcSet instead. The `findresource` operator may be used for this purpose as follows:

```
(myprocset) /ProcSet findresource begin
   : (insert original page description here)
end
```

Each embedded font definition will have to be packaged into a standalone font definition. This may include removing any references to functions defined in the prolog. Alternatively, the original embedded font definition can be made dependent on the named ProcSet created for the prolog as described for page descriptions. Note that the font will need to use the `findresource` operator to access the ProcSet.

For form definitions, the contents of the `PaintProc` procedure of the defined form must be packaged into a standalone PostScript fragment. This may include removing any references to functions defined in the prolog. Alternatively, the original embedded `PaintProc` definition can be made dependent on the named ProcSet created for the prolog as described for page descriptions.

Given the procset, standalone font, form and page descriptions, a first version of a PPML dataset can be created as follows (assuming A4 paper):

```
<PPML>
   <CONFORMANCE SubSet=”GA” Level=”1”/>
   <SUPPLIED_RESOURCES>
      <SUPPLIED_RESOURCE Type=”ProcSet” Format=”application/postscript”
         Name=”prolog” ResourceName=”myprocset”>
         <EXTERNAL_DATA Src=”myprocset.ps”/>
      </SUPPLIED_RESOURCE>
   </SUPPLIED_RESOURCES>
   <REQUIRED_RESOURCES>
      <SUPPLIED_RESOURCE_REF Ref=”prolog”/>
   </REQUIRED_RESOURCES>
   <PAGE_DESIGN TrimBox=”0 0 595 842”/>
   <JOB>
      <SUPPLIED_RESOURCES>
      <SUPPLIED_RESOURCE Type=”Font”
         Format=”application/postscript”
         Name=”font-1” ResourceName=”fontname-1”>
         <EXTERNAL_DATA Src=”font-1.ps”/>
      </SUPPLIED_RESOURCE>
```



```
        :
        <SUPPLIED_RESOURCE Type=”Font”
        Format=”application/postscript”
        Name=”font-m” ResourceName=”fontname-m”>
        <EXTERNAL_DATA Src=”font-m.ps”/>
    </SUPPLIED_RESOURCE>
    </SUPPLIED_RESOURCES>
<REQUIRED_RESOURCES>
    <SUPPLIED_RESOURCE_REF Ref=”font-1”/>
        :
    <SUPPLIED_RESOURCE_REF Ref=”font-m”/>
</REQUIRED_RESOURCES>
<REUSABLE_OBJECT>
    <OBJECT Position=”0 0”>
        <SOURCE Format=”application/postscript”
            Dimensions=”[form-1-urx] [form-1-ury]”
            ClippingBox=”[form-1-bounding-box]”>
            <EXTERNAL_DATA Src=”form-1.ps”/>
        </SOURCE>
    </OBJECT>
    <OCCURRENCE_LIST>
        <OCCURRENCE Name=”form-1-view-1”>
            <VIEW>
                <TRANSFORM Matrix=”[form-1-transform-1]”/>
            </VIEW>
        </OCCURRENCE>
            :
        <OCCURRENCE Name=”form-1-view-2”>
            <VIEW>
                <TRANSFORM Matrix=”[form-1-transform-2]”/>
            </VIEW>
        </OCCURRENCE>
    </OCCURRENCE_LIST>
</REUSABLE_OBJECT>
    :
<REUSABLE_OBJECT>
    <OBJECT Position=”0 0”>
        <SOURCE Format=”application/postscript”
            Dimensions=”[form-k-urx] [form-k-ury]”
```



```
            ClippingBox="[form-k-bounding-box]">
            <EXTERNAL_DATA Src="form-k.ps"/>
         </SOURCE>
      </OBJECT>
      <OCCURRENCE_LIST>
         <OCCURRENCE Name="form-k-view-1">
            <VIEW>
               <TRANSFORM Matrix="[form-k-transform-1]"/>
            </VIEW>
         </OCCURRENCE>
            :
         <OCCURRENCE Name="form-k-view-2">
            <VIEW>
               <TRANSFORM Matrix="[form-k-transform-2]"/>
            </VIEW>
         </OCCURRENCE>
      </OCCURRENCE_LIST>
   </REUSABLE_OBJECT>
   <DOCUMENT>
         <PAGE>
            <MARK Position="0 0">
               <OBJECT Position="0 0">
                  <SOURCE Format="application/postscript"
                     Dimensions="595 842">
                     ClippingBox="[bbox-page-1-1]">
                     <EXTERNAL_DATA Src="page-1-1.ps"/>
                  </SOURCE>
               </OBJECT>
            </MARK>
            <MARK Position="[x] [y]">
               <OCCURRENCE_REF Ref="form-1-view-1"/>
            </MARK>
            <MARK Position="0 0">
               <OBJECT Position="0 0">
                  <SOURCE Format="application/postscript"
                     Dimensions="595 842"
                     ClippingBox="[bbox-page-1-2]">
                     <EXTERNAL_DATA Src="page-1-2.ps"/>
                  </SOURCE>
```



```
                </OBJECT>
            </MARK>
        </PAGE>
            :
        <PAGE>
            <MARK Position="0 0">
                <OBJECT Position="0 0">
                    <SOURCE Format="application/postscript"
                         Dimensions="595 842">
                        <EXTERNAL_DATA Src="page-n.ps"/>
                    </SOURCE>
                </OBJECT>
            </MARK>
        </PAGE>
    </DOCUMENT>
</JOB>
</PPML>
```

The above PPML dataset adheres to the PPML/GA specification, which is indicated by the inclusion of the **CONFORMANCE** element. To ensure that the font and form definitions have access to the procedures defined in the original prolog a **SUPPLIED_RESOURCE** for myprocset is created at the PPML level. The **SUPPLIED_RESOURCE_REF** in the **REQUIRED_RESOURCES** element at the PPML level is needed to inform the PPML Consumer that the myprocset ProcSet shall be made available in VM for all PostScript fragments used in the PPML scope.

The fonts, which may be dependent on the myprocset ProcSet, are defined at the **JOB** level such that they have a defined dependency on the myprocset ProcSet.

Each form is defined using a **REUSABLE_OBJECT** element. The bounding box of the original form definition is needed to inform the PPML Consumer of the area of the form coordinate system to cache. The *Dimensions* attribute introduces clipping to a bounding box of "0 0 w h" for *Dimensions="w h"*. If the form's lower left corner has negative components, additional PostScript code must be added to translate the lower left corner of the form to (0,0). In that case, the bounding box for the form needs to be adjusted accordingly.

In PostScript, forms may depend on the graphics state from which the form is executed. This is not the case in PPML. Therefore, for such forms multiple **REUSABLE_OBJECT** definitions may be necessary. Those additional **REUSABLE_OBJECT** definitions would need to have the appropriate PostScript commands added to setup the graphics state in which the form is used.

To use a form, the page description will have to be split into two parts: the part before a form is executed and the part after a form is executed. The reference to the form will become an **OCCURRENCE_REF** in the PPML dataset. Each of the other parts is converted into a separate **MARK** element. The *ClippingBox* attribute of the **SOURCE** element should be added to reflect the area of the page that the part draws on.

Reusable content in PPML can only be translated when being placed onto a page. Therefore, for page descriptions that scale, rotate or shear a form, an **OCCURRENCE** must be defined with a **VIEW** element containing a **TRANSFORM** element that appropriately transforms the form. The

form reference can then be implemented using a **MARK** element that contains an **OCCURRENCE_REF** element. The **Ref** attribute of that **OCCURRENCE_REF** element must contain the name of the **OCCURRENCE** definition that appropriately transforms the form.

Image data which is repeated in multiple page descriptions should be converted into a **REUSABLE_OBJECT**. The process for this is similar to what is needed for forms except that the image data should be stored in TIFF or JPEG instead of in PostScript.

The transformation matrix in each **OCCURRENCE** defines the changes to the Current Transform Matrix (CTM) made at the point of reference to the image. Therefore, if the original code was:

```
300 400 translate
50 50 scale
100 200 1 [100 0 0 -200 0 200] decodeimage image
<image data>
```

The TIFF image would therefore contain 100 by 200 pixels at an x and y resolution of 72 dots per inch. Assuming the TIFF image data was stored in image1.tiff the **REUSABLE_OBJECT** definition would be:

```
<REUSABLE_OBJECT>
<OBJECT Position="0 0">
   <SOURCE Format="image/tiff" Dimensions="100 200">
      <EXTERNAL_DATA Src="image1.tif"/>
   </SOURCE>
   <VIEW>
      <TRANSFORM Matrix="0.01 0 0 0.005 0 0"/>
   </VIEW>
</OBJECT>
<OCCURRENCE_LIST>
   <OCCURRENCE Name="image1">
      <VIEW>
         <TRANSFORM Matrix="50 0 0 -50 300 450"/>
      </VIEW>
   </OCCURRENCE>
</OCCURRENCE_LIST>
</REUSABLE_OBJECT>
```

The value for **OBJECT/VIEW/TRANSFORM/@Matrix** can be found by executing the following PostScript fragment:

```
[100 0 0 200 0 0] matrix invertmatrix ==
```

By applying this transformation, the image is scaled to always be 1 by 1 point in size.

The value for **OCCURRENCE/VIEW/TRANSFORM/@Matrix** should be equal to:

$$[100\ 0\ 0\ -200\ 0\ 200] \times [100\ 0\ 0\ 200\ 0\ 0]^{-1} \times [50\ 0\ 0\ 50\ 300\ 400]$$

in which the matrix [50 0 0 50 300 400] describes the changes to the CTM. This transformation combines the effect of the image-matrix for the `image` operator and the transformations to the CTM into a single transformation mapping the 1x1 pt image to its correct place on the page.

The same transformation matrix can be found by executing the following PostScript fragment:

```
[100 0 0 -200 0 200]
[100 0 0 200 0 0]
matrix invertmatrix matrix concatmatrix
matrix currentmatrix matrix invertmatrix
300 400 translate
50 50 scale
matrix currentmatrix exch matrix concatmatrix matrix concatmatrix ==
```

For images that only differ in location on the page, the resulting transformation matrices will only differ in the last two numbers of those matrices. In such cases, the same **OCCURRENCE** definition can be referenced.

It is recommended to define the **OCCURRENCE** with a transformation matrix in which the last two numbers are set to 0. The *Position* attribute of the **MARK** element that is used to position the image (using **OCCURRENCE_REF**) can then be set to the last two numbers of the transformation matrix calculated for that image.

PDF to PPML application notes

Today many PDF based workflows exist. PPML/GA allows PDF content to be repurposed and optimized for printing. In PDF files that are created using VDP software, elements repeated on multiple pages are often stored as a Form XObject in PDF.

To effectively use PDF as content for a PPML dataset, the PDF output needs to be modified as follows:

- each Form or Image XObject must be output as a PDF page,
- each PDF page that references a Form or Image XObject should be split into multiple pages. Each page containing a part of the original page before or after a call to draw a Form or Image XObject.

PDF allows translucent form and Image objects to be defined. PPML does not support translucency therefore such Form and Image objects cannot be optimized in PPML and must remain a Form or Image XObject in the PDF content data.

For each Form or Image XObject, a **REUSABLE_OBJECT** is created. Using an **EXTERNAL_DATA_ARRAY** element the newly generated PDF page containing the Form or Image content is referenced.

PPML only allows a **REUSABLE_OBJECT** to be positioned (not rotated or scaled) and an **OCCURRENCE** must be defined for each transformation applied to the Form or Image in the original PDF content. References to Form and Image XObjects are replaced by **OCCURRENCE_REF** elements that reference the **OCCURRENCE** that applies the appropriate transformation. Each part of the original PDF pages is placed onto a PPML page using a **MARK** with an **EXTERNAL_DATA_ARRAY** element to reference that PDF page.

A PPML/GA compliant dataset can then be constructed as follows (using A4 paper):

```
<PPML>
   <CONFORMANCE SubSet=”GA” Level=”1”/>
   <PAGE_DESIGN TrimBox=”0 0 595 842”/>
   <JOB>
   <REUSABLE_OBJECT>
      <OBJECT Position=”0 0”>
         <SOURCE Format=”application/pdf”
            Dimensions=”[form-1-urx] [form-1-ury]”
            ClippingBox=”[form-1-bounding-box]”>
            <EXTERNAL_DATA_ARRAY Src=”content.pdf” Index=”1”
IndexUsage=”Mulitple”/>
         </SOURCE>
      </OBJECT>
      <OCCURRENCE_LIST>
         <OCCURRENCE Name=”form-1-view-1”>
```



```
            <VIEW>
               <TRANSFORM Matrix="[form-1-transform-1]"/>
            </VIEW>
         </OCCURRENCE>
            :
         <OCCURRENCE Name="form-1-view-2">
            <VIEW>
               <TRANSFORM Matrix="[form-1-transform-2]"/>
            </VIEW>
         </OCCURRENCE>
      </OCCURRENCE_LIST>
   </REUSABLE_OBJECT>
      :
   <REUSABLE_OBJECT>
      <OBJECT Position="0 0">
         <SOURCE Format="application/pdf"
            Dimensions="[form-k-urx] [form-k-ury]"
            ClippingBox="[form-k-bounding-box]">
            <EXTERNAL_DATA_ARRAY Src="content.pdf" Index="k"
IndexUsage="Multiple"/>
         </SOURCE>
      </OBJECT>
      <OCCURRENCE_LIST>
         <OCCURRENCE Name="form-k-view-1">
            <VIEW>
               <TRANSFORM Matrix="[form-k-transform-1]"/>
            </VIEW>
         </OCCURRENCE>
            :
         <OCCURRENCE Name="form-k-view-2">
            <VIEW>
               <TRANSFORM Matrix="[form-k-transform-2]"/>
            </VIEW>
         </OCCURRENCE>
      </OCCURRENCE_LIST>
   </REUSABLE_OBJECT>
   <DOCUMENT>
         <PAGE>
            <MARK Position="0 0">
               <OBJECT Position="0 0">
```

```
                    <SOURCE Format=”application/pdf”
                        Dimensions=”595 842”>
                        ClippingBox=”[bbox-page-1-1]”>
                        <EXTERNAL_DATA_ARRAY Src=”content.pdf” Index=”k+1”
IndexUsage=”Multiple”/>
                    </SOURCE>
                </OBJECT>
            </MARK>
            <MARK Position=”[x] [y]”>
                <OCCURRENCE_REF Ref=”form-1-view-1”/>
            </MARK>
            <MARK Position=”0 0”>
                <OBJECT Position=”0 0”>
                    <SOURCE Format=”application/pdf”
                        Dimensions=”595 842”
                        ClippingBox=”[bbox-page-1-2]”>
                        <EXTERNAL_DATA_ARRAY Src=”content.pdf” Index=”k+2”
IndexUsage=”Multiple”/>
                    </SOURCE>
                </OBJECT>
            </MARK>
        </PAGE>
            :
        <PAGE>
            <MARK Position=”0 0”>
                <OBJECT Position=”0 0”>
                    <SOURCE Format=”application/pdf”
                         Dimensions=”595 842”>
                        <EXTERNAL_DATA_ARRAY Src=”content.pdf” Index=”k+n”
IndexUsage=”Multiple/>
                    </SOURCE>
                </OBJECT>
            </MARK>
        </PAGE>
    </DOCUMENT>
  </JOB>
</PPML>
```


PPML with PDF-based content can also be converted into PPML/VDX compliant datasets. More PDF-based examples can be found in the PPML/VDX application notes[1].

[1] Committee for Graphic Arts Technologies Standards*. Application Notes for CGATS.20 (PPML/VDX), August 204.*

Page picking

PPML may also be used to re-use existing PDF assets. Using an **EXTERNAL_DATA_ARRAY** individual PDF pages may be taken from multiple PDFs and combined into a new print file. Page numbers, headers and footers may be added using dynamically generated PostScript or PDF content. Simple applications include reversing the output to simulate face-down/face-up output.

The following PPML dataset shows how to concatenate pages from two PDF files into a single document.

```
<PPML>
    <CONFORMANCE SubSet="GA" Level="1"/>
    <PAGE_DESIGN TrimBox="0 0 595 842"/>
    <JOB>
    <DOCUMENT>
           <PAGE>
               <MARK Position="0 0">
                   <OBJECT Position="0 0">
                       <SOURCE Format="application/pdf"
                           Dimensions="[bbox-page-1-1]">
                           <EXTERNAL_DATA_ARRAY Src="file-1.pdf" Index="1"
IndexUsage="Multiple"/>
                       </SOURCE>
                   </OBJECT>
               </MARK>
           </PAGE>
           <PAGE>
               <MARK Position="0 0">
                   <OBJECT Position="0 0">
                       <SOURCE Format="application/pdf"
                           Dimensions="[bbox-page-1-2]">
                           <EXTERNAL_DATA_ARRAY Src="file-1.pdf" Index="2"
IndexUsage="Multiple"/>
                       </SOURCE>
                   </OBJECT>
               </MARK>
           </PAGE>
           <PAGE>
               <MARK Position="0 0">
                   <OBJECT Position="0 0">
                       <SOURCE Format="application/pdf"
```



```
                        Dimensions="[bbox-page-2-1]">
                        <EXTERNAL_DATA_ARRAY Src="file-2.pdf" Index="2"
IndexUsage="Multiple"/>
                    </SOURCE>
                </OBJECT>
            </MARK>
        </PAGE>
    </DOCUMENT>
</JOB>
</PPML>
```


Impositioning

Multiple PDF pages may be combined into a PPML sheet allowing complex imposition schemes to be implemented using PPML datasets without the need to modify the original content data.

The following PPML dataset shows how to implement 2-up A4 onto an A3 sheet:

```
<PPML>
    <CONFORMANCE SubSet=”GA” Level=”1”/>
    <PAGE_DESIGN TrimBox=”0 0 1190 842”/>
    <JOB>
    <DOCUMENT>
           <PAGE>
               <MARK Position=”0 0”>
                   <OBJECT Position=”0 0”>
                       <SOURCE Format=”application/pdf”
                           Dimensions=”[bbox-page-1]”>
                           <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”1”
IndexUsage=”Multiple”/>
                       </SOURCE>
                   </OBJECT>
               </MARK>
               <MARK Position=”595 0”>
                   <OBJECT Position=”0 0”>
                       <SOURCE Format=”application/pdf”
                           Dimensions=”[bbox-page-2]”>
                           <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”2”
IndexUsage=”Multiple”/>
                       </SOURCE>
                   </OBJECT>
               </MARK>
           </PAGE>
               :
           <PAGE>
               <MARK Position=”0 0”>
                   <OBJECT Position=”0 0”>
                       <SOURCE Format=”application/pdf”
                           Dimensions=”[bbox-page-n-1]”>
                           <EXTERNAL_DATA_ARRAY Src=”file.pdf” Index=”n-1”
IndexUsage=”Multiple”/>
```



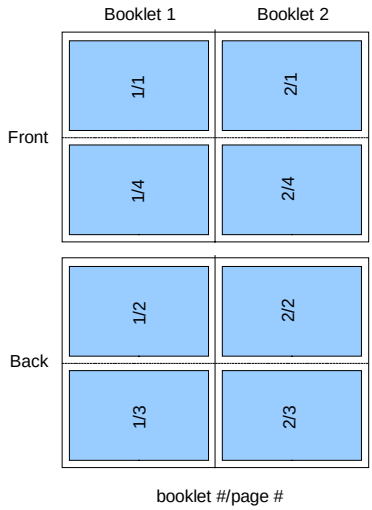
```
                    </SOURCE>
                </OBJECT>
            </MARK>
            <MARK Position="595 0">
                <OBJECT Position="0 0">
                    <SOURCE Format="application/pdf"
                        Dimensions="[bbox-page-n]">
                        <EXTERNAL_DATA_ARRAY Src="file.pdf" Index="n"
IndexUsage="Multiple"/>
                    </SOURCE>
                </OBJECT>
            </MARK>
        </PAGE>
    </DOCUMENT>
</JOB>
</PPML>
```

The following PPML dataset shows how to implement 4-up A5 onto an A3 sheet, where each of the A5 pages is rotated for folding into A5 booklets:

```
<PPML>
    <CONFORMANCE SubSet="GA" Level="1"/>
    <PAGE_DESIGN TrimBox="0 0 1190 842"/>
    <JOB>
    <DOCUMENT>
        <PAGE>
            <MARK Position="0 0">
                <OBJECT Position="0 0">
                    <SOURCE Format="application/pdf"
                        Dimensions="[bbox-page-1]">
                        <EXTERNAL_DATA_ARRAY Src="file.pdf" Index="1"
IndexUsage="Multiple"/>
                    </SOURCE>
                </OBJECT>
                <VIEW>
                    <TRANSFORM Matrix="0 1 1 0 0 0"/>
                </VIEW>
            </MARK>
            <MARK Position="0 421">
                <OBJECT Position="0 0">
                    <SOURCE Format="application/pdf"
                        Dimensions="[bbox-page-2]">
```



```
                        <EXTERNAL_DATA_ARRAY Src="file.pdf" Index="2"
IndexUsage="Multiple"/>
                    </SOURCE>
                <VIEW>
                    <TRANSFORM Matrix="0 1 -1 0 0 0"/>
                </VIEW>
                </OBJECT>
            </MARK>
            <MARK Position="595 0">
                <OBJECT Position="0 0">
                    <SOURCE Format="application/pdf"
                        Dimensions="[bbox-page-3]">
                        <EXTERNAL_DATA_ARRAY Src="file.pdf" Index="3"
IndexUsage="Multiple"/>
                    </SOURCE>
                </OBJECT>
                <VIEW>
                    <TRANSFORM Matrix="0 1 1 0 0 0"/>
                </VIEW>
            </MARK>
            <MARK Position="595 421">
                <OBJECT Position="0 0">
                    <SOURCE Format="application/pdf"
                        Dimensions="[bbox-page-2]">
                        <EXTERNAL_DATA_ARRAY Src="file.pdf" Index="4"
IndexUsage="Multiple"/>
                    </SOURCE>
                <VIEW>
                    <TRANSFORM Matrix="0 1 -1 0 0 0"/>
                </VIEW>
                </OBJECT>
            </MARK>
        </PAGE>
            :
    </DOCUMENT>
  </JOB>
</PPML>
```

Example 1 Imposed pages

Annex A
(informative)
Revision history

Version 1.0, January 2011

Initial release.

Index

A

alpha channel · 4

B

binary mask · 4, 5
booklets · 56

C

CLIP_RECT · 5, 6, 8, 11, 14, 15, 19
***ClippingBox* · 4, 5, 7, 14, 47**
CONFORMANCE · 46

D

***Digital Print Ticket (DPT) Specification* · 3**
***Dimensions* · 4, 46**

E

EXTERNAL_DATA_ARRAY · 49, 53

F

Form XObject · 49

G

graphical elements · 4
GUID (*Global Unique Identifier*) · 27

I

Image XObject · 49
imaging model · 4
Imaging model · 4
Impositioning · 55
INTERNAL_DATA · 29

J

JDF · 3
JOB · 46
job ticket · 40

M

MARK · 5, 6, 10, 11, 12, 13, 14, 20, 21, 47, 48, 49

O

OBJECT · 5, 6, 7, 8, 9, 10, 11, 14, 16, 17, 18, 46, 48, 49
OCCURRENCE · 47, 48
OCCURRENCE_REF · 47, 49

P

PostScript · 5, 13, 22, 38, 41, 43, 46, 47, 48, 53
PPML Consumer · 6, 14, 16, 22, 38, 46
PPML Producer · 22
ProcSet · 46

R

REQUIRED_RESOURCES · 46
Reusable content · 26, 47
REUSABLE_OBJECT · 5, 14, 16, 17, 18, 29, 46, 47, 49

S

scope promotion · 26
SOURCE · 4, 5, 6, 7, 14, 16, 47
SUPPLIED_RESOURCE · 22, 23, 24, 25, 38, 39, 43, 44, 46
SUPPLIED_RESOURCE_REF · 22, 46

T

TRANSFORM · 4, 5, 6, 7, 11, 14, 18, 47
transformation matrix · 47, 48

U

UUID (Universal Unique Identifier) · 27

V **VIEW · 4, 5, 6, 7, 8, 10, 14, 16, 18, 47, 48**

Version · 59

