SDL XPP

XML Professional Publisher:

XML Export

for use with XPP 9.3
November 2018

Contents

About This Manual

Where Do I Start?

This manual addresses distinct tasks that will allow you to understand and use the XPP XML Export Utility.

To accomplish these tasks, use this guide as follows.

If you need to . . .	*Then read . . .*
Learn about XML Export	Chapter 1
Run the XML Export utility	Chapter 2
Understand the XML Export hierarchy	Chapter 3
Understand groups, classes and styles	Chapter 4
Understand how XML Export handles computer generated text	Chapter 4
Understand how XML Export handles traces	Chapter 5
Understand how XML Export handles images, rules, cross-references and shapes	Chapter 6
Understand XML Export tabular output	Chapter 7
Understand how XML Export handles document tags	Chapter 8
Find more detailed information about XML Export tags and attributes	Appendix A
Find sample XML Export documents and XSLT fragments	Appendix B
Find a sample XML-to-ePub conversion	Appendix C

Conventions Used in This Manual

This manual uses a set of symbolic, typographic, and terminology conventions to categorize specific information. Take a few moments to become familiar with these conventions before you use this manual.

Convention	*Description*
Bold	Bold type, used in procedures, indicates the object of the action. It may be a menu option, a push button, or a field, and so forth. For example, "select **Open**" means select the menu option called Open. Position **cursor** means to move the cursor to a specific location. Enter appropriate **information** means that you enter information that is appropriate for your site or for the specific job. Bold may be used elsewhere in the manual to denote emphasis.
Key	Capital first letter and the word "key" indicates a key on the keyboard. Capital first letter and the words "Softkey menu" indicate the menu pad to the right of the XyView. Unless otherwise indicated, this manual assumes that you press the **Enter** key at the end of a command line.
Key+Key	This sequence indicates a Shortcut Key combination. Hold down the first key while also pressing the second key, that is **ALT+F4** means to hold down the *Alt* key while also pressing the *F4* key.
`Message`	A monospaced typeface indicates an application's response to your action. For example, "the message `Enter Value` appears" means that the application displays the words "Enter Value."
Italics	In running text, an italic typeface indicates a new term; for example, "The replacement string of characters is the *output string*." In a system message, a field entry, or an argument to a command, an italic typeface indicates a variable. For example, *filename* is a variable in the message "Can't open *filename*." Italics are also used for the names of programs, such as *Perl*.
" "	Quotation marks indicate that you should enter exactly what the instructions tell you to enter. For example, type "**yes**" means to type the letter **y**, the letter **e**, and the letter **s**.
[ ]	Reverse-video square brackets represent tags in standard XPP. Tags are general-purpose commands defined in the Item Format Spec and embedded in a document. They generally format logical components of text, such as chapter openings, headers, or lists. For example, the tag for beginning a chapter might be **[chap]**.

Convention	*Description*
《 》	Reverse-video angle brackets represent XPP-supplied macros (XyMacros) and user-defined macros. XyMacros are commands embedded in text to set or change formatting or typographic style. For example, the XyMacro to end a page is **《ep》**. Reverse-video angle brackets also represent tags when you use XPP in either XML or SGML mode. Note that when in XML or SGML mode, XPP does not use the conventional reverse-video square brackets.
《?xxx》	Reverse-video angle brackets with a beginning question mark represent macros when using XPP in SGML mode.
《?xxx?》	Reverse-video angle brackets with a beginning and ending question mark represent macros when using XPP in XML mode.

When entering values for some arguments in macros and for some fields in specs, you must *qualify* the value by specifying a *unit* of measure. The available unit qualifiers are:

- q — Points
- p — Picas
- c — Ciceros
- d — Didots
- i — Inches
- m — Millimeters
- k — Kyus
- n — Microns (XPP units)
- z —centimeters

For example, 6q means 6 points, 4p means 4 picas.

Note: *You can also use pc, pt, in, mm, and cm in fields where the system allows the standard ISO unit abbreviations.*

Chapter 1

Overview of the XML Export Utility

Understanding XML Export

In simplest terms, you can use the XPP XML Export Utility to generate an XML file containing format descriptions of composed XPP pages.

A file output using XML Export represents the final paginated form of an XPP Division in a format that is suitable for post-processing into other XML-based formats, such as ePub, XHTML, WordML, as well as non-XML formats, such as RTF and EDGAR HTML.

The XML Export utility takes data structures that already exist in an XPP division, containing information about pages, streams, blocks, and lines, and converts them into a consistent set of XML tags in a well-formed markup. You can then use standard transformation tools (such as XSLT) to develop a core set of reference conversions. In this way, any XML-based publishing format can be derived directly from any XPP page.

With XML Export, you can:

- Access all elements of a composed XPP page including Stream, Block, and Line tags, with final XPP composition values represented as XML attributes.
- Group lines together if they have the same tag structure. For example, paragraphs of text within a column are grouped.
- Identify pickups and stories as named streams with an id attribute.
- Identify blocks by their type: frill, main, footnote, title, caption, annotation, or graphic. Footnote and graphic blocks also contain id attributes identifying them by name.
- Refer to font family and variant numbers using actual font names.
- Convert color rule numbers and pattern percentages into their associated RGB color values. If the numbers are specified in the CMYK color space, they will be converted to the RGB equivalent.

  The color can be unspecified for transparent colors (or no color) for block and line background colors and shape main and alternate colors. The color will also be unspecified for all image types except xyraster images, because only xyraster images can be colorized.
- Express measurement values in any type of units. The default is points.
- Generate cross references for pickup requests, footnote requests, and XPP references.
- Mark images with the complete filename and the full path to the graphic.

Chapter 2

Running the XML Export Utility

Running the XML Export Utility Using divxml

The XML Export Utility is run by a program called **divxml**. The executable file location is specified in the XYV_EXECS environment variable. This utility can be run only from the command line or by using a Job tool. The XML Export Utility can read pages from a division, a specified set of divisions, or all of the divisions in a specified Job. Depending on the output options specified, this utility can output XML tags and attributes for every XPP element on every page specified.

Note: *Divxml passes pdfmarks to the XML file as processing instructions only. Any actual processing of pdfmarks is done by the output driver.*

All text is output as UTF-8. All XCS characters that do not have a Unicode equivalent are output as &xfxxxx, or the UTF-8 equivalent. If your XPP divisions have characters that are not mapped to ″real″ Unicode (i.e., they are in the Unicode Private Use range), you must include the -ncrh argument to XML Export (divxml). This will render those characters in their hex values which you can translate as desired via XSL.

Regardless of whether the original XPP division exists in ML or “Classic” mode, the Utility produces an XML output file.

Note: *Empty trace inserts are not reflected in the XML Export output.*

All output units are measured in points. The qualified unit designator (″q″) is not output with the number. However, if you specify the ″-units″ option, the output value is represented with its qualifier name. For example, if you specify ″-units i″, an output value of three inches is represented as ″3i″.

Note: *It is the user′s responsibility to process numbers and units properly when doing arithmetic operations.*

Command line summary

Format **divxml [–options]** *filename***.xml**

Option	***Description***
–cd *pathname*	Directory path name of a division. The default is the current directory.

Option	*Description*
filename.**xml**	Output file name. The default is **tout.xml**. The output file format is UTF-8, unless you specify the –ncrh or –ncrd options. Use the keyword **stdout** for writing to standard output. **Note:** If you specify **stdout**, the command runs silently by setting the –xsh and –s options.
–job	Perform job-level processing of multiple divisions in the Document Assembly Ticket. If you specify this option with the –cd option, you must specify the *pathname* for the –cd option to the job level.
–da *fieldname*	Process a Document Assembly Ticket field, if that field is set to *yes* in the Document Assembly Ticket. This switch must be used with the **–job** switch. Legal field names are: **ci[ti]**—CITI field **co[mpose]**—COMPOSE field **pr[int]**—PRINT field **s[pell]**—SPELL field **l[oep]**—LOEP field **pd[f]**—PDF field **e[dgar]**—EDGAR field **n[one]** Process all divisions listed in the DA ticket. **tm[ain]** Process divisions of type main. **tc[iti]** Process divisions of type citi. **tt[ocpdf]** Process divisions of type pdftoc. **ti[dxpdf]** Process divisions of type pdfidx. **tl[oep]** Process divisions of type loep. *fieldname* can be entered in upper or lower case. You can provide just enough letters to uniquely identify the entry. If you do not specify this option, all divisions specified in the Document Assembly Ticket with div type set to *main* are processed, regardless of how any other field is set.
–div *div1,div2, . . .*	Process divisions in the job, overriding those specified in the Document Assembly Ticket.
–all	Process all of the divisions in the job, ignoring the Document Assembly Ticket specifications.

Option	*Description*
–p# –P#	Specifies a starting and/or ending page type/number. Without this option (or any other page selection option(s)), all Main pages are processed. If both a starting and ending page are specified, they both must be of the same page type. If –p is specified and not –P, the end page is the last of the –p page type. If –P is specified and not –p, the start page is the first of the –P page type.
–ip *pglist*	Specifies a list of pages to process. Single page type/numbers in the list are separated by commas. Ranges of pages can be specified separated by hyphens. You can mix and match single page numbers with page ranges. If the *pglist* argument contains a slash (/) anywhere in its string, it is assumed to be the name of a path/file that contains the page list. Page list files can contain one or more lines; new line characters are treated as commas.
–ud	Process only those pages that have the current update number specified in the Division Ticket or the Job Ticket. The current update number is obtained from the Division Ticket. If that value is set to **none**, the update number is obtained from the Job Ticket. The current number is compared to the update number for each page; if they match, the page is processed. This option is meaningful only if the *Loose-leaf* field in the Division Ticket is not set to **off**.
–un#	Process only those pages that have the update number specified as #. This # is compared to the update number for each page; if they match, the page is processed.
–ch	Process only those pages that have changed since the last time they were typeset. Only those pages that have been edited or composed since they were last typeset are processed. This option works in conjunction with the **–ud/un** and **–bu** options.

Option	*Description*
–bu	Process backup pages. If the page is a right-hand page, force the following page, if any, to be processed. If the page is a left-hand page, force the previous page to be processed. This option is meaningful only if the Division Ticket's *Page Sheet Style* field specifies **double sided**. This option is illegal by itself (that is, it is meaningless unless it is used in conjunction with one of the above page selection options).
–sess#	Process only those pages that have the specified session number #. Use –1 (minus one) to specify the division's current session number. The default behavior is to process pages with any session number.
+m +s +f +p +fr +l	Process only main (+m), story (+s), footnote (+f), pickup (+p), frills (+fr), and layout (+l). The default behavior is to process all types of blocks.
–noframes	If frills are output with divxml, the frame blocks can be suppressed with the `-noframes` option.
–wait	Wait indefinitely if the XSF files are in use in a division.
–nol	Open division in read-only mode without locking.
–s	Be completely silent except for error messages.
–xsh	Suppress the output of starting title and exit information.
–X	Output extra-verbose progress messages for debugging purposes. Exit Status: 0 = Success, no errors were found. 1 = One or more warning errors occurred. –1 (255) = Fatal error, the program aborted.
–dos *or* –unix	Force Windows/Unix line endings (<return><linefeed>). Line endings default to the current platform.
–unit *value*	Output units: picas (p), points (q), millimeters (m), inches (i), ciceros(c), didots (d), kyus (k). The default unit is points (q).
–bidi	Output the text in bidi order. By default text is output in logical order. For more information, see "Bidirectional Composition" in the XPP documentation.

Option	***Description***
–ligin	Output the input side of ligatures. By default, the output side of ligatures outputs. For example, if there is an “ffl” ligature, the default output is the single Unicode character (&#xfb04). However, if the “–ligin” option is used, the individual characters output as they were originally entered (“ffl”).
–ncrh, –ncrd	Output multibyte Unicode characters as either hexidecimal numeric character references or as decimal numeric character references. Single byte Unicode characters within the normal ASCII range.
–wpi	Output processing instructions. By default, processing instructions are ignored.
–tags	Output the content start tag names with attributes as standalone XML tags. Output does not include the content tags and their matching end tags by default. For more information, please see Chapter 8.
–noa	Do not output attributes and values (used in conjunction with –tags). By default, output does not include the content tags and their matching end tags. For more information, see Chapter 8.
–tage	Output content end tag names. By default, output does not include the content tags and their matching end tags. For more information, see Chapter 8.
–fattr	Force the display of true/false attribute flags (for example, lastpage=”false”). By default, only “true” attributes are displayed.

Option	*Description*

–fm *mapfile*	Use the font mapping file to change the Postscript names.
	
	You can use the font mapping file to map Postscript font names into other forms, typically browser based HTML fonts. The format of the map file is similar to a “property file”: it declares the Postscript name followed by an equal sign and then the name of the substitution font name. You can add comments by starting a line with a “#”. Blank lines are ignored and spaces are also significant. Use the wildcards “*” for all characters, “?” for single character, and “[a-x]” for a single character match of any character within a range.
	
	For example, a font mapping file might look like this:
	
	# This is a comment
	Helvetica=Arial
	Helvetica-Bold=Arial-Bold
	Times-New*=Times
	
	Note: Wildcards cannot carry a match to the right side of the expression.

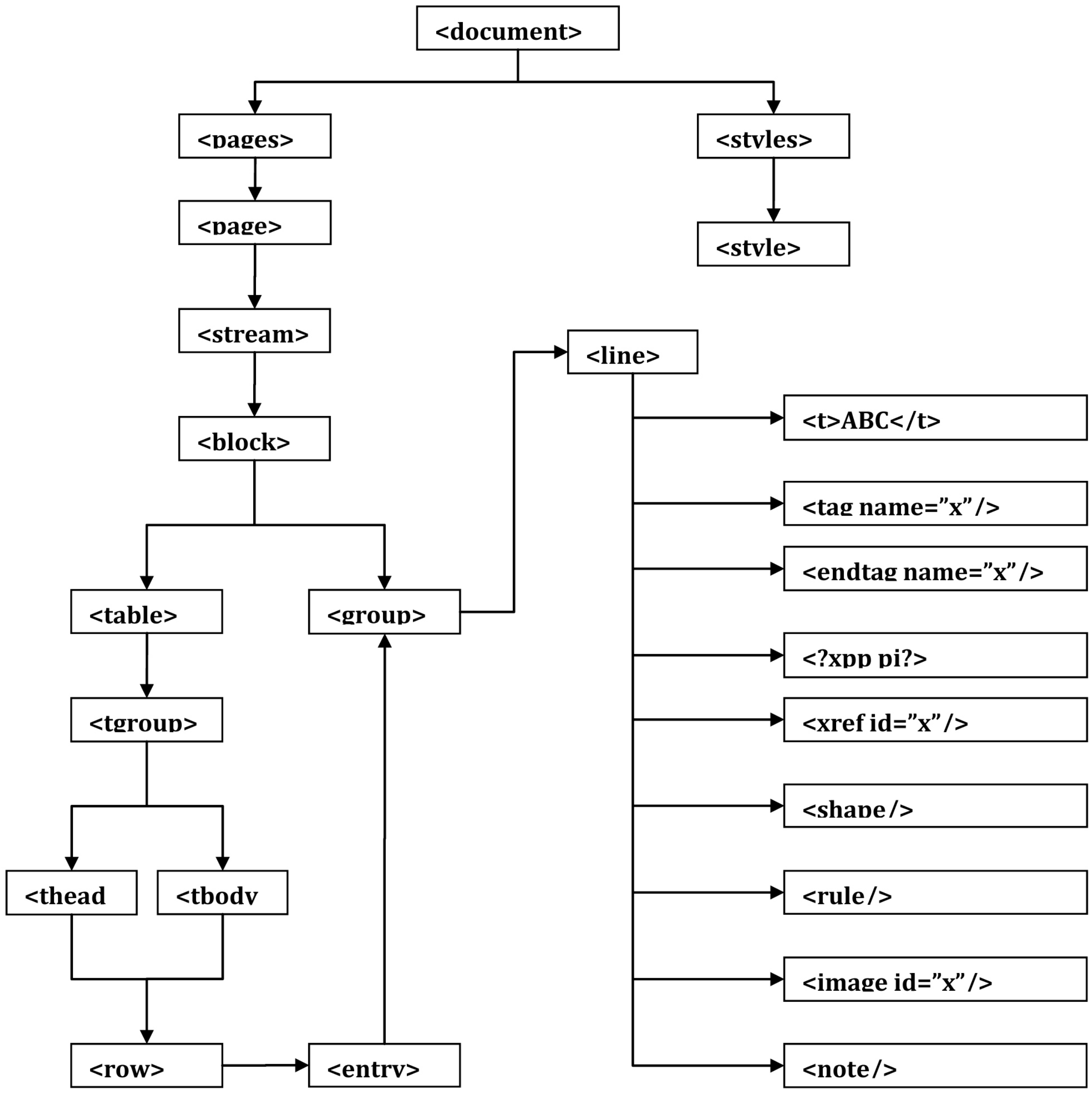
–ts	Print the output by tabbing with spaces instead of tab characters. By default, the output tabs using the tab character.
–dump	Output all database information.

Chapter 3

Hierarchical Structure

Hierarchical Structure of XML Instances

This chart shows the hierarchical structure of an XML instance generated by the XML Export Utility. You can find a complete list of tags and their attributes in Appendix A.



The Basic Elements: 《document》, 《page》, 《stream》, 《block》, and 《line》

Each page of a composed XPP division is made up of elements represented by distinct data structures. XML Export captures these data structures and outputs them in a hierarchical XML form: **Document/Page/Stream/Block/ Line**. The descriptions of the basic XML tags that follow are not complete but will give you an overview of the structures and associated values. (For a complete description, see Appendix A.)

Document

- Root element

Page

- X,Y offset within a frame
- Page width and depth
- List of streams within the page

Stream

- Stream type (main, story, frill, pickup, footnote)
- X,Y offset within a page
- Stream width and depth
- List of blocks within a stream

Block

- Block type (text, graphics, title, annotation, caption, footnote separator)
- X,Y offset within a stream
- Block width and depth
- List of lines within the block (text and tables)

Line

- X, Y offset within a block
- Line indents and measure
- Composition parameters
- Text within a line.

  Note: This is in logical order unless the **–bidi** option is used. If **–bidi** is enabled, then output is right-to-left characters, where appropriate, on a line by line basis. For more information about the **–bidi** option, please refer to the *XML Professional Publisher: Bidirectional Composition* manual.

Chapter 4

Classes, Groups, and Styles

Classes, Groups, and Styles

The XML Export Utility assembles adjacent lines of text that share the same style formatting into structures bounded by the <group> tag. Each group of lines carries the @class attribute with a value set either to the name of a generic label entered into the Class field (introduced in the XPP 8.4 Item Format specification), or to the name of the Item Format tag itself if the Class field is empty.

Classes

You will find a new *Class* field in the Item Format pgraf rule. Allowable values are any text strings that are valid within an XSL match statement.

Assigning a class to a particular XPP tag is a form of style mapping. For example, if HTML tag names are assigned to XPP tags (h1, h2, p, and so on), XML Export enables you to reference them when creating transformations for the XML output.

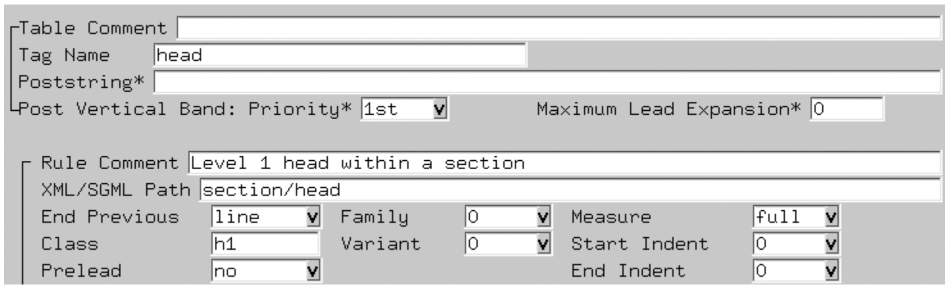
Groups

The **<group>** tag allows all lines that are part of the same classification to be grouped together. The **<group>** tag is the fundamental element for determining the document style. This tag enables you to find the real scope of a particular style format and to use class names as a basis to create different types of XML files from the XML Export output.

For example, the following XML source document contains four tags: top, section, head, and para.

```
<top>
    <section>
        <head>head</head>
        <para>A p</para>
    </section>
</top>
```

In the Item Format specification, the class entry for **<head>** is *h1*



```
Table Comment
Tag Name      head
Poststring*
Post Vertical Band: Priority* 1st        Maximum Lead Expansion* 0

  Rule Comment Level 1 head within a section
  XML/SGML Path section/head
  End Previous   line      Family   0      Measure       full
  Class          h1        Variant  0      Start Indent  0
  Prelead        no                        End Indent    0
```

and the entry for **<para>** is “p”:

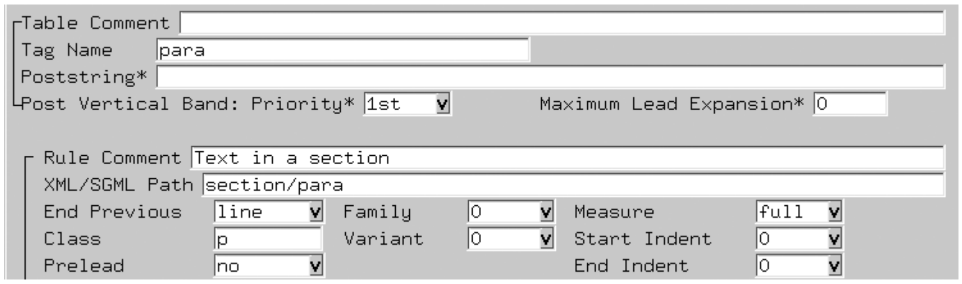


Table Comment					
Tag Name	para				
Poststring*					
Post Vertical Band: Priority*	1st			Maximum Lead Expansion*	0
Rule Comment	Text in a section				
XML/SGML Path	section/para				
End Previous	line	Family	0	Measure	full
Class	p	Variant	0	Start Indent	0
Prelead	no			End Indent	0

The XML Export output would be:

```
<group class=“h1”>
      <line>head</line>
</group>
<group class=“p”>
      <line>A p</line>
</group>
```

The **<group>** tag also recognizes that tags in XPP may be nested. While any group of lines has exactly one class name, the parent tags might also have a class name. To include class ancestry, the **<group>** tag contains a *@pclass* attribute whose value is an Xpath statement describing the ancestry of the current *@class* name (that is, the full Xpath name except for the final class name). If parent tags do not have a class name, the actual tag name is used.

In the following source XML fragment, **<top>** has a class name *doc*;

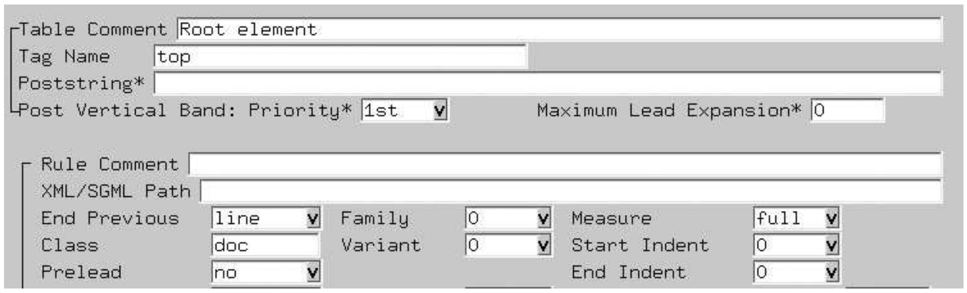


Table Comment	Root element				
Tag Name	top				
Poststring*					
Post Vertical Band: Priority*	1st			Maximum Lead Expansion*	0
Rule Comment					
XML/SGML Path					
End Previous	line	Family	0	Measure	full
Class	doc	Variant	0	Start Indent	0
Prelead	no			End Indent	0

<section> has a class name *sect*:

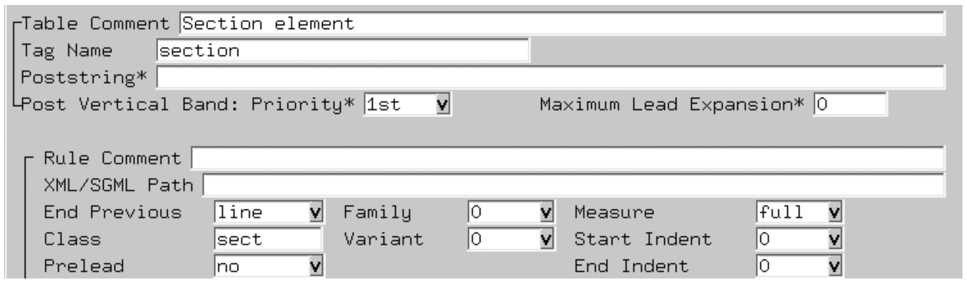


Table Comment	Section element				
Tag Name	section				
Poststring*					
Post Vertical Band: Priority*	1st			Maximum Lead Expansion*	0
Rule Comment					
XML/SGML Path					
End Previous	line	Family	0	Measure	full
Class	sect	Variant	0	Start Indent	0
Prelead	no			End Indent	0

<head> has a class name *h1*:

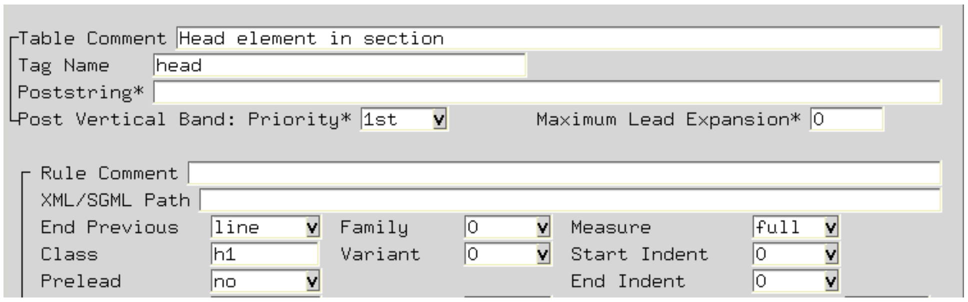


Table Comment: Head element in section
Tag Name: head
Poststring*:
Post Vertical Band: Priority* 1st   Maximum Lead Expansion* 0

Rule Comment:
XML/SGML Path:
End Previous: line   Family: 0   Measure: full
Class: h1   Variant: 0   Start Indent: 0
Prelead: no   End Indent: 0

and **<para>** has a class name *p*:

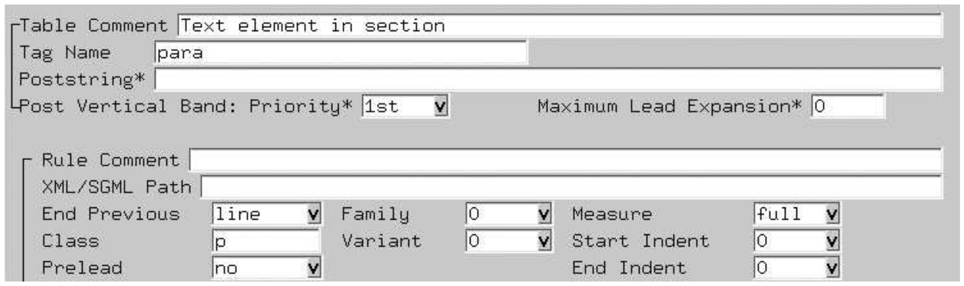


Table Comment: Text element in section
Tag Name: para
Poststring*:
Post Vertical Band: Priority* 1st   Maximum Lead Expansion* 0

Rule Comment:
XML/SGML Path:
End Previous: line   Family: 0   Measure: full
Class: p   Variant: 0   Start Indent: 0
Prelead: no   End Indent: 0

```
<top>
    <section>
        <head>head</head>
        <para>A p</para>
    </section>
</top>
```

Export outputs the following:

```
<group class="h1" pclass="doc/sect">
    <line>head</line>
</group>

<group class="p" pclass="doc/sect">
    <line>A p</line>
</group>
```

In addition to the class ancestry, the **<group>** tag also contains a *@dh* (document hierarchy) attribute whose value is an Xpath statement describing the XPP content tag ancestry (including any of the attributes of those original tags). When writing transforms for an XML Export output file, you cannot use an XPath statement to navigate the hierarchy of the original XPP content tags, but you can query the *@dh* attribute and check for the existence of parent tags using XSL string functions. This "special" processing is useful for applications in which you need the source document hierarchy. Refer again to the previous example source fragment:

```
<top>
    <section>
        <head>head</head>
        <para>A p</para>
    </section>
</top>
```

As before, **<top>** has a class name *doc*, **<section>** has a class name *sect*, **<head>** has a class name *h1*, and **<para>** has a class name *p*. The *@dh* attribute would appear in the XML output as follows:

```
<group class="h1" dh="/top/section/head">
    <line>head</line>
</group>

<group class="p" dh="/top/section/para">
    <line>A p</line>
</group>
```

The *@dh* attribute will also work for Divisions in "classic" mode. Even though there is no document hierarchy, groups of lines with the same style formatting still are linked together by a named class.

For example, given a simple classic XPP document in which **[head]** has a class name *h1* and **[para]** has a class name *p*:

```
[head]tag
[para]This is a para
```

the XML output would be:

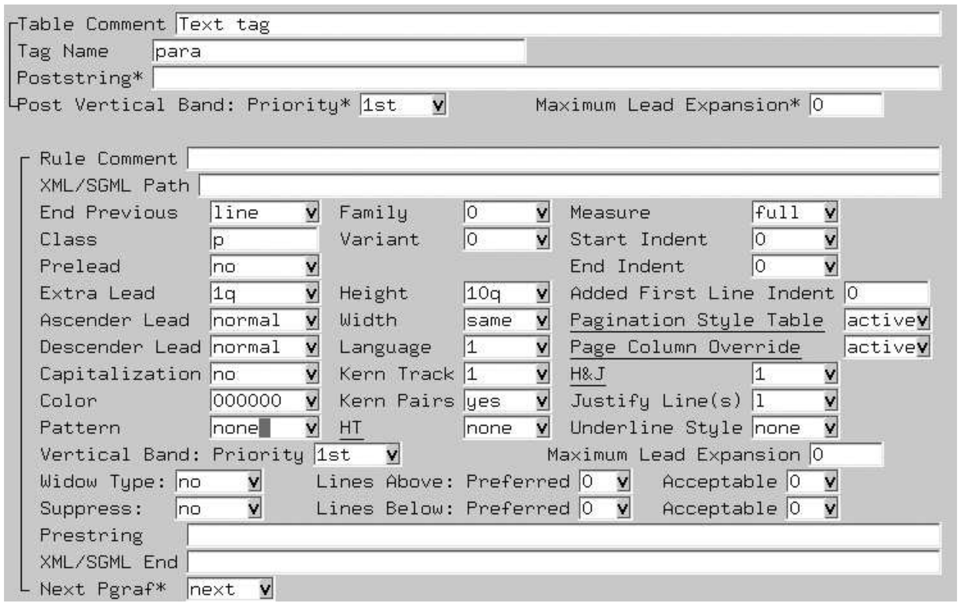
```
<group class="h1">
    <line>head</line>
</group>

<group class="p">
    <line>This is a para</line>
</group>
```

Styles

Each **<group>** tag has an associated *style*. The *@style* attribute of the **<group>** tag references a **<style>** tag from the styles section at the end of the XML Export file. (For details, see the sample output file in *Appendix B*.)

Each group style contains specifications derived from fields in the active Item Format plus any other processing instruction overrides in the Prestring field.

For example, the following Item Format:



```
Table Comment Text tag
Tag Name      para
Poststring*
Post Vertical Band: Priority* 1st          Maximum Lead Expansion* 0

Rule Comment
XML/SGML Path
End Previous      line     Family       0      Measure                  full
Class             p        Variant      0      Start Indent             0
Prelead           no                           End Indent               0
Extra Lead        1q       Height       10q    Added First Line Indent  0
Ascender Lead     normal   Width        same   Pagination Style Table   active
Descender Lead    normal   Language     1      Page Column Override     active
Capitalization    no       Kern Track   1      H&J                      1
Color             000000   Kern Pairs   yes    Justify Line(s)          1
Pattern           none     HT           none   Underline Style          none
Vertical Band: Priority 1st                    Maximum Lead Expansion 0
Widow Type:  no     Lines Above: Preferred 0     Acceptable 0
Suppress:    no     Lines Below: Preferred 0     Acceptable 0
Prestring
XML/SGML End
Next Pgraf*  next
```

would result in this **<style>** tag definition:

> **<style name=“para” font=“Times-Roman” size=“10” cm=“normal”**
> **color=“#000000” color-cmyk=“0 0 0 1” ff=“0” fv=“0”/>**

Text Tags and Styles

Within a line, all text with the same font characteristics are enclosed in a single **<t>** tag. As with groups, the style of the text is referenced with an *@style* attribute whose value is derived from the **Class** defined in the Item Format. These styles also are defined by a **<style>** tag, and as with groups each unique style generates a unique style class name based on the active tag.

The following example shows a source XML fragment with a **<para>** tag with class set to *p* and an **<emph>** tag with class set to *b*:

> **<para class=″p″>**This is **<emph class=″b″>**bold**</emph></p>**

Would generate:

> **<t style=“p”>**This is **</t><t style=“b”>**bold**</t>**
>
> **<styles>**
> **<style name=“p” font=“Helvetica” size=“10” color=“#000000”/>**
> **<style name=“b” font=“HelveticaBold” size=“10”**
> **color=“#000000”/>**
> **</styles>**

A group style represents the initial composition state of that group; the actual text within the group lines can contain processing instructions that override some or all of the specs listed in the group style. Therefore, the **<t>**

tag may also include an *@style* attribute that reflects these changes but is still based on the group style. (One might consider a text style as a "substyle" to the ancestor group style.)

Text styles that are based on group styles are also listed in the **<styles>** section at the end of the XML file and contain the same name as the associated group style appended with a unique sequence number (such as "style.###").

For example, a simple source XML document that contains a color change processing instruction and a **<para>** tag with class *p*:

```
<para>Is <?co;red?>red</para>
```

would result in the following XML output:

```
<group class="p" style="p">
    <line>
        <t style="p">Is </t>
        <t style="p.1">red</t>
    </line>
</group>

<styles>
    <style name="p.1" font="Helvetica" size="10"
      color="#FF0000"/>
    <style name="p" font="Helvetica" size="10" color="#000000"/>
</styles>
```

In another example, a small source text fragment that changes color and emboldens:

```
Is <?co;red?>red<?co;black?> Is <?bold?>bold
```

would be output as:

```
<group class="p" style="p">
    <line>
        <t style="p">Is </t>
        <t style="p.1">red</t>
        <t style="p"> Is </t>
        <t style="p.2">bold</t>
    </line>
</group>

<styles>
    <style name="p" font="Helvetica" size="10" color="#000000"/>
    <style name="p.1" font="Helvetica" size="10"
      color="#FF0000"/>
    <style name="p.2" font="HelveticaBold" size="10"
      color="#000000"/>
</styles>
```

Computer-Generated Text

Since text is grouped based on the font characteristics, the **<t>** text element contains an additional attribute (*cgt="true"*) when the text is computer generated.

For example, a computer-generated page reference number (25) created by:

Page number = **<ri;%relpgnum>**

would be output as:

<t . . >Page Number = **</t>**
<t . . . cgt="true">25**</t>**

Chapter 5

Edit Traces

Overview

Edit traced text within XPP is classified as inserted, strike-thru (marked for deletion), or deleted. Text that is actually deleted is not kept within the page. However, a deletion mark is maintained to indicate that text was deleted here.

@trace Attribute

For inserted and strike-thru text, additional attributes to the **<t>** tag flag a sequence of characters with the same edit trace classification. When text is traced, an *@trace* attribute can have the values *insert* or *strike-thru*. When the end of the trace is found, a new **<t>** tag will output to delineate the traced sequence of characters from the next section of text.

For example, given a source text line with **[SOI]** and **[EOI]** marks showing inserted text:

This is **[SOI]**inserted**[EOI]** text.

The XML coding would be:

<t>This is **</t>**

<t trace="insert">inserted**</t>**
<t> text.**</t>**

Similarly, if text is marked for strike-through with **[SOD]** and **[EOD]** marks:

This is **[SOD]**struck through**[EOD]** text.

The XML coding would be:

<t>This is **</t>**
<t trace="strike-thru">struck through**</t>**
<t> text**</t>**

Deleted text is treated the same way, but there is no text associated with the edit trace mark. In this case, the trace attribute will be placed in an empty **<t>** tag.

For example, a text line with a mark:

Something was **[DEL]** here

would output:

<t>Something was **</t>**
<t trace="delete"></t>
<t> here**</t>**

@trace-session and @trace-suppress

Composition also tracks the editing session number and the suppression of change bars and underlines.

The *@trace-session* attribute value is the session number that the trace was created. The *@trace-suppress* attribute indicates suppressed traces and can have values “underline”, “changebar”, or “all”.

Chapter 6

Images, Rules, Xrefs and Shapes

Overview

XML Export provides elements such as <image>, <rule>, <xref> and <shape> that you can use to output appropriate text and graphics depending on the final output format. For a complete list of the attributes associated with each of these tags, please see Appendix A, XML Export Markup Guide.

The Image Tag

The Image tag **<image>** contains information about a graphic and its reference.

- Image name
- Full Path to image
- Type of image
- X, Y offset to top left of image.
- Image width and depth
- Clipping values (x,y,w,d)
- Image color (RGB)
- Image rotation
- Scale factor
- Placement priority (any value between -127 and 127)

The Ruling Tag

The Ruling tag **<rule>** contains information about horizontal and vertical rules.

- X offset from current position, Y offset from base line
- Rule Width and Depth
- Rule Color

The Cross Reference Tag

The Cross reference tag **<xref>** contains information about references to other elements.

- Reference name (@id attribute)
- Reference type (**main, story, pickup, footnote, frills, layout**)

The Shape Tag

The Shape tag **<shape>** contains information about drawing objects within the document.

- Type of shape (**rectangle, oval, gradient, lineto**)
- X offset from current position, Y offset from base line
- Width and depth
- Color
- Outline type and size
- Dashes
- Priority
- Screen angle and frequency

Chapter 7

Tables

Overview

In XPP, you can code tables using either the CALS model or XPP macros. In either case, the table is organized as a related set of data structures that represent position, rules, layout, and other tabular formatting.

With XML Export, tables are output in the CALS format, regardless of how they are represented in XPP. Each cell (**<entry>** tag) is marked as belonging to a particular numbered column and each row contains a row number. Rows are then marked as belonging to header or body rows.

The CALS XML syntax is generated from the XPP table structure, enhanced with other XML attributes describing post-composition information such as final column widths, and ruling on/off.

Within a cell, text is output using the same structure as regular text. Each cell contains **<group>**, **<line>**, and **<t>** tags representing the actual content and style of the tabular text.

Following is a sample source fragment containing a two column table with one row and a two-column span:

```
<table frame=″all″>
    <tgroup tgroupstyle=″techdoc″ cols=″2″>
        <colspec colnum=″1″ colname=″col1″ colwidth=″83pt″/>
        <colspec colnum=″2″ colname=″col2″ colwidth=″155pt″/>
        <tbody>
            <row>
                <entry colsep=″1″ rowsep=″1″ namest=″col1″
                  nameend=″col2″>
                    <text>This is a table cell.</text>
                </entry>
            </row>
        </tbody>
    </tgroup>
</table>
```

XML Export would then output (not all attributes output are shown):

```
<table frame="all">
    <tgroup tgroupstyle="techdoc" cols="2">
        <colspec colnum="1" colname="col1" colwidth="83pt"/>
        <colspec colnum="2" colname="col2" colwidth="155pt"/>
        <tbody>
            <row row="1">
                <entry col="1" colsep="1" rowsep="1"
                  namest="col1" nameend="col2">
                    <group class="text" pclass="" dh="/text"
                      style="text.1">
                        <line>This is a table cell.</line>
                    </group>
                </entry>
            </row>
        </tbody>
    </tgroup>
</table>
```

Chapter 8

XPP Original Document Tags and XML Export

Overview

A source XML document that has been imported into XPP has an overall structure of its own, conforming to a DTD or schema. XPP's XML structured page output cannot preserve the original tagging structure, but you may want to refer to the original tag names and attributes in their locations within the XPP pages. This can be done by adding options to the divxml command (see Chapter 2, *Running the XML Export Utility*).

<tag>, <attr> and <endtag>

Original content tags are preserved with a stand-alone **<tag name="tagname">** element that encapsulates the original Item Format name. It also contains child tags that capture attribute names and values.

For example, a **<a type="b">** tag would be output as:

```
<tag name="a">
    <attr name="type">b</attr>
</tag>
```

Similarly, end tags (such as **</a>**) are output as:

```
<endtag name ="a"/>
```

Processing instructions are output at the point where they occur in the line. Thus, they may occur inside or outside a **<t>** tag. **<?ep;balance?>** in the source file would output as:

```
<?xpp ep;balance?>
```

In XPP classic mode, the tag and pgraf markers are also preserved. The tag name uses the same **<tag>** element (without attributes). However, the pgraf mark will be translated into the empty tag *@name="Xpgm"*. For example, a classic tag that has three rule definitions:

```
[text]This is text
▪rule two
▪rule three"
```

This would be output as:

```
<tag name ="text"/><t>This is text</t>
<tag name="Xpgm"/><t>rule two</t>
<tag name="Xpgm"/><t>rule three</t>
```

Appendix A

XML Export Markup Guide

XML Export Tag and Attribute List

Following you will find a list of all of the tags and attributes supported by the XML Export utility. Each section provides the tag name, any possible parent and child tags, a description of the element, and a table that describes the attributes.

Note: Attribute value names are chosen to align with their composition system variable names.

<document>

Parent(s) none

Children **pages, styles**

Description The **document** tag is the root level tag that describes all of the pages in the document. The document tag contains pages and styles, and provides versioning as well as the namespace attribute.

Note: *XPP Divisions are not represented in the XML output. The pages are the result of either processing a single division and optionally selecting a page range, or using the job processing options to select multiple divisions. In either case, the XML output reflects just a list of pages.*

Attribute	*Description*
version	The XML schema version for the "divxml" output format. Useful for versioning of the spec.

Attribute	*Description*
xmlns	Namespace is www.xyenterprise.com for the <document> tag.

<styles>

Parent(s) **document**

Children **style**

Description The **styles** tag is a container for the collection of styles. It only contains style tags.

Note: *There are no attributes associated with this tag.*

<style>

Parent(s) **styles**

Children none

Description The **style** tag describes the font characteristics of known style elements. Each style tag has attributes describing the text font name, size, color, etc. The *@name* attribute is referenced from the **t** and **group** tags to describe font characteristics that apply to its children. The styles are generated after the XML page descriptions are declared. This should not be a problem if using XML tools.

Attribute	*Description*
name	Font style name.
basename	Name of style without the .x. (i.e. <style name=″para.1″ basename=″para″...>)
size	Point size of the text in units.
condensed-size	Condensed font height (if different from **size**) *[optional]*.
font	Postscript Font name from XPP fonts defined in the TSF font spec.
ff	Font family number (0 – 3000).
fv	Font variant number (0– 2047).
cm	Case Mode (**normal, smallcap, upper, lower**).

Attribute	*Description*
color	Color of text as a RGB value in hex form #RRGGBB, where RR = red, GG = Green, and BB = blue. This is derived from the XPP color number and the pattern percentage.
color-cmyk	Color of text in a percentage of CMYK as a space delimited array; i.e. color-cmyk="1.0 0.0 0.0 0.0".
color-spot	Color of text as a color name if it is a spot color *[optional]*.
background-color	Color of text background (#RRGGBB) *[optional]*.
background-color-cmyk	Color of text background in a percentage of CMYK as a space delimited array; i.e. background-color-cmyk= "1.0 0.0 0.0 0.0" *[optional]*.
background-color-spot	Color of text background as a color name if it is a spot color *[optional]*.
underline	Text is underlined if **true** *[optional]*.

<pages>

Parent(s) **document**

Children **page**

Description The pages tag is the container for a collection of pages that are derived from a XPP division or a group of Divisions via Job Processing.

Note: *There are no attributes associated with this tag.*

<page>

Parent(s) **pages**

Children **stream**

Description The page tag describes all of the visual elements on a page. Each page has a fixed position within a frame as well as a fixed width and depth.

Attribute	*Description*
relpgnum	Relative page number (1 – n) where n is the maximum number of pages allowed in XPP.
psrnum	Page rule number (1-255).
plname	Page layout name.

Attribute	*Description*
p1 thru p6	Page numbers %P1 – %P6.
px	Page offset X position in the frame.
py	Page offset Y position in the frame.
psx	Page width.
psy	Page depth.
fsx	Frame width.
fsy	Frame depth.
lastpage	Last page of the rule sequence if **true**.
pfexcept	Exception page format used if **true**.
pagetype	Page type (**left, right**).
rlfiller	Page is a filler due to page selection rules if **true**.
filler	Page is a filler if **true**.
notextinpg	There is no text in the page if **true**.
forcepend	Page is created due to pending pickups if **true**.

<stream>

Parent(s) **page**

Children **block**

Description The **stream** tag contains textual matter tied to a text flow. The main text stream manages the flow of regular text from column to column and from page to page. Pickups and stories are also streams. The stream is used to contain all of the blocks that exist for the named pickup or story on the page. There can be multiple streams within a page, but by convention there is only one each *main*, *frill*, and *layout* stream. The footnote stream is a special entity that it is used to position a group of footnotes at the bottom of a block or spanning multiple blocks at the bottom of the page. The **stream** tag contains only the block tag as child.

Attribute	*Description*
type	Stream type (**main, frill, story, footnote, pickup, layout**).
id	ID of the stream (story or pickup name if applicable).
sx	Stream x-position relative to top of page (0 for all streams except pickups and footnotes).
sy	Stream y-postion relative to top of the page (0 for all streams except pickups and footnotes).

Attribute	*Description*
ssx	Stream width (0 for all streams except pickups and footnotes).
ssy	Stream depth (0 for all streams except pickups and footnotes).
dldid	ID number of the pickup, footnote, story that is contained in the stream (*internal*).
groupnum	Unique number group used to process blocks that can be balanced.
svjmode	Saved vertical justification mode of last block in stream (**top, bottom, justify**).
epfilltype	End page fill type (**balance, align, justify**).
class	Security "portion" classification number (**0-255**).
rotate	Final rotation of the stream (**0, +- 90, +-180, +-270**).
strmrot	Requested rotation of the stream from pickup definition (**0, +- 90, +-180, +-270**).
vj_space	Vertical justification space about the stream (pickup only).
cerrors	Composition errors within stream if **true**.
cntedit	Content edits in stream if **true**.
recomp	Stream should be re-composed if **true**.
frzcmp	Compostion is frozen within stream if **true**.
pgscope	Page Pickup if **true** (pickup stream only).
hfull	Pickup is full width if **true** (pickup stream only).
vfull	Pickup is full depth if **true** (pickup stream only).
here	This is a "here, bound or suspend" pickup if **true** (pickup stream only).
before	This is a "before" pickup if **true** (pickup stream only).
justified	Stream is vertically justified if **true**.
ipcbal	Last blocks are IPC balanced if **true**.
touched	Pickup was modified but not composed if **true** (pickup stream only).
tplaced	Pickup is top placed if **true** (pickup stream only).
bplaced	Pickup is bottom placed if **true** (pickup stream only).
firstack	Pickup is first in stack if **true** (pickup stream only).
lastack	Pickup is last in stack if **true** (pickup stream only).
balup	Block balancing moved up if **true**.

Attribute	*Description*
baldown	Block balancing moved down if **true**.
balrunaway	Composition handled a block balancing runaway situation if **true**.
predit	Presentation modifications were made if **true** (not currently used).

<block>

Parent(s) **stream**

Children **group, table, image**

Description The **block** tag is a container for a group of related lines. It can best be thought of as a column of text (although page column overrides can modify this behavior.) Geometry and information relating to vertical pagination are available as attributes. Footnote and graphic blocks contain an "id" attribute for references to footnotes and images respectively.

An image block is a special type of block that does *not* contain any lines. It is a placeholder for a reference to an image, with attributes describing the placement and clipping values of the image. A **block** tag contains **group, table,** or **image** children. An image block contains only an **image** child.

Attribute	*Description*
type	*main*, *frame*, *frill*, *layout*, *story*, *footnote*, *caption*, *graphic*, *annotation*, or *title*.
id	ID of the block (frame, frill, or layout block ID number, footnote or image name, if applicable).
bx	x-position of the block relative to the start of the stream in units.
by	y-position of the block relative to the start of the stream in units.
biy	Initial y-position of the block relative to the start of the stream in units.
bsx	Block width in units.
bsy	Block depth in units.
bisy	Initial block depth in units (from the page layout definition of computed depth in "pc" blocks).
background-color	Color of the block background in RGB as #RRGGBB *[optional]*.

Attribute	*Description*
background-color-cmyk	Color of the block background in a percentage of CMYK as a space delimited array; i.e “1.0 0.0 0.0 0.0” *[optional]*.
background-color-spot	Color of the block background as a color name if it is a spot color *[optional]*.
btextlen	Length of text in block w/o VJ in units.
bslack	Amount of white space added by vertical justification in units.
vjerr	Vertical justification error in units.
justified	Block is vertically justified if **true**.
vjmode	Block vertical justification from layout (**top, bottom, center, justify**).
svjmode	Actual Block vertical justification (**top, bottom, center, justify**).
ipcnum	Active PC number (**0** = none, **span**, or **1–254**).
ipcgnum	IPC group number. Blocks with the same number are balanceable.
fipcblk	First block in IPC group if **true**.
lipcblk	Last block in IPC group if **true**.
ebvjmode	Block is ended via an “eb” (**top, bottom, center, justify**).
pgtbl	PT table number.
trymode	Block justified text direction (**squeeze, expand**).
original	This is an original (non-pc) block defined by the layout spec if **true**.
bottom	This block is at the bottom of the stream/page if **true**.
rtl	This block is right-to-left “bidi” if **true**.
xfloat	Floating x-position of the block: **0**=not floating, **1**=stack, **2**=left, **3**=c enter, **4**=right, **5**=lalign, **6**=calign, **7**=ralign, **8**=match, **9**=flip (pickup only).
yfloat	Floating y-position of the block: **0**=not floating, **1**=stack, **2**=left, **3**=c enter, **4**=right, **5**=lalign, **6**=calign, **7**=ralign, **8**=match, **9**=flip (pickup only).
priority	Placement hierarchy (-127 <= priority <= 127 – pickup block definition only).
direction	Rotation direction (pickup block only).
xright	This block is right stacked if **true** (pickup only).
ybottom	This block is bottom stacked if **true** (pickup only).
caldepth	This block’s depth is computed if **true** (pickup only).

Attribute	*Description*
fulldepth	This block's depth is full if **true** (pickup only).
calwidth	This block's width is computed if **true** (pickup graphic block only).
defwidth	This block's width is full if **true** (pickup only).

<group>

Parent(s) **block, entry**

Children **line**

Description Lines of similar type are grouped together and are marked by composition as belonging to the same class. The class name is derived from the active tag name or the class name in the active Item Format rule. If any parent tags have a classification name in the active Item Format rule, the nested parent hierarchy will be reflected in the *pclass* attribute.

A group is a way to logically combine lines that have the same attributes. For example, a head and paragraph are basic building blocks of style development. All lines within a head will be contained within a group. The paragraph will be combined into its own group.

Attribute	*Description*
class	The user derived class name of the lines with the same style. Composition will group successive lines that are marked with the same classification. This may be the active tag name of the "class" name assigned to the tag in the Item Format rule.
pclass	Parent class structure. Contains a description of parent tags of the form "x/y/z" for the document hierarchy **<x><y><z>**. Each class name is also the active tag name or an assigned name from the Item format spec.
dh	Document Hierarchy. Shows the document structure of the nested tag *before* the start of the text of the first line within the group. This will be of the form "/top/section/body/para" and can be used by translation services that require additional hints of the structure of the document.
style	Name of the style describing the font and line information pertaining to the start of the group of lines (see the <style> tag). This is computed from the active style before the first text character on the line.

Attribute	*Description*
cont	The active group has been continued from a previous column (if equal to **true**).

<line>

Parent(s) **group**

Children **t, image, xref, tag, endtag, rule, shape, note**

Description The **line** tag contains all of the text and other displayable objects that have associated composition parameters. A line is horizontally justified and has an X and Y position, width, left and right indent, and measure. The line space width, letter spacing, quad mode, and other parameters are computed by composition and are also available as attribute values. A line tag is always located inside a group.

Attribute	*Description*
xfinal	X-position of line in units (includes indent values).
yfinal	Y-position of line in units (includes leading and pre-lead value).
lnwidth	Width of the line in units.
lnmeas	Line measure in units.
lindent	Left indent in units.
rindent	Right indent in units.
qdtype	Quad type of the line (**left, right, center, justify, forcej**).
postlead	Postead of the line in units.
prelead	Prelead of the line in units.
bandwidth	Width of the space band in units.
letspace	Letter spacing amount in units.
maxasc	Maximum ascender lead on the line in units.
maxdesc	Maximum descender lead on the line in units.
ldasc	Ascender line lead
lddesc	Descender line lead
ldextra	Extra lead of the line in units.
sobprelead	Start of block pre-lead if **true**.
bmline	Block merge text line if **true**.
ljerrs	Line justification errors if **true**.

Attribute	*Description*
loverset	Line oversets available measure if **true**.
cerrors	Composition errors if **true**.
edtrcds	Edit traces if **true**.
chngbar	Change Bar active for the line if **true**.
first	First line of a head/para if **true**.
last	Last line of a head/para if **true**.
vkpstart	Variable keep start (via the Item Format head/para rules) if **true**.
vkpsa	Variable keep start if **true** (via the Item Format head/para rules) if **true**.
kptype	Page keep type if **true** (for example, a head).
kpstart	Hard Keep start (via the ⟨**ks**⟩ ⟨**ke**⟩) if **true**.
kpsa	Hard Keep is active (via the ⟨**ks**⟩ ⟨**ke**⟩) if **true**.
kpse	End of hard keep (via the ⟨**ks**⟩ ⟨**ke**⟩) if **true**.
bkpt	Vertical justification break point if **true**.
nobkpt	Disallow vertical break here if **true**.
ebinline	There is an ⟨**eb**⟩ in the line if **true**.
lepgraf	Line ends a “pgraf” if **true**.
leitm	Line ends an item if **true**.
quadset	Quad ended line if **true**.
wrdbrk	Line ends with a ″word break″ line end if **true**.
tbstart	Start of table if **true**.
tbsa	Table is active if **true**.
tbend	Last line of table if **true**.
tbrhd	Tabular running head if **true** (continuation table only).

< t >

Parent(s) **line**

Children **dt**

Description The **t** tag is the basic building block for text on a line and describes a sequence of Unicode characters with the same font characteristics and color. When a font name changes, a new **t** tag will be generated to contain the next sequence of Unicode characters. The **t** tag contains text as its element node and never breaks across lines. Each font transition on a line will generate a new **t** tag. Sequential text that is marked as computer generated or is suppressed will also be grouped together.

Attribute	***Description***
x	Relative horizontal move in units. Any horizontal moves that precede the text will be rolled up into a horizontal position change for the text. This does *not* include moves associated with character kerning, track kerning, and letter spacing.
y	Y-offset of text relative to the base line (yfinal) in units. Any vertical base line shift from positioning commands that precede the text will be rolled up into a vertical change for the text. Changes in the vertical position will generate new **t** tags.
style	Name of the font style (size, point size, font, and color) as defined in the **style** tag.
suppress	If **true**, text is suppressed *[optional]*.
cgt	If **true**, text is computer generated *[optional]*.
trace	Text is traced (**insert, strike-thru, delete**) *[optional]*.
trace-session	Session number of the trace *[optional]*.
trace-suppress	Suppress trace (**underline, changebar, all**) *[optional]*.
ul#=*"weight color"*	Underline rule, where # is 1-10 if there are underline rules active. This has a two value list with the first item containing the rule *weight* and the second item containing the rule *color* as an RGB value. Example: **《t ul1=".5 #FF0000"》** would be used for a half point red rule that is specified as **《ul;1》**.

<dt>

Parent(s) **t**

Children none

Description The **dt** tag is a container for discretionary text. It is mainly used to hold the computer generated hyphen character that has been inserted by composition, though this may be extended in the future. The user is responsible for deciding whether to include this text. If line endings are preserved, then this element should be processed. If text is flowing across lines, then the element should be ignored to prevent hyphenated words in the middle of text.

Note: *There are no attributes associated with this tag.*

<xref>

Parent(s) **line**

Children none

Description The **xref** tag refers to a named cross-reference mark. This includes marked spots, pickups and footnotes. The *@id* attribute references the object. For pickups and footnotes, **xref** always refers to the associated content on this (or another) page.

Attribute	***Description***
id	Reference name.
type	Reference type (**footnote, pickup, main, story, frills, layout**).

<tag>

Parent(s) **line**

Children **attr**

Description The **tag** tag describes the original XPP content tag locations and contains the name of the original content tag with an optional parameter list (using the **attr** tag).

Attribute	***Description***
name	The name of the original content tag.

<attr>

Parent(s) **tag**

Children none

Description A list of attribute values that are part of the original content.

Attribute	*Description*
name	The name of the original attribute.

<endtag>

Parent(s) **line**

Children none

Description The **endtag** tag describes the original content end tag locations. It contains the name of the original content end tag.

Attribute	*Description*
name	The name of the original content end tag.

<note>

Parent(s) **line**

Children none

Description A trace marker that locates a “note”.

Attribute	*Description*
id	Name of the note.
type	Type of note (text, link).
owner	Creator of the note.

<image>

Parent(s)	**line, block**
Children	none

Description

The **image** tag exports an image reference. The named image is referenced at the current position. When the tag is part of a text line, It is assumed to have “0” width; i.e. the “cursor” position is in the same location after output of the graphic. All positioning is done by means of horizontal and vertical moves before and after the image reference (reflected in the x,y offset values). The image tag also contains user supplied clipping values.

An **image** tag can also be included directly within a **block** tag when an image block is used within a pickup. In this case, the x,y values of 0 indicate that the image position is located at the x,y location of the **block** tag.

Attribute	*Description*
id	Image name.
path	Path of image. Full system path location to the master graphic image. ***Note:*** *This path is system dependent.*
type	Type of image. Values can be: **xyf** (xyraster), **tif**, **eps**, **jpg**, **pdf**, **svg**, **cgm**, **png**, **gif**, **bmp**, **unknown**.
x	X-offset from current cursor position in units (0 for a pickup image block).
y	Y-offset of image relative to the base line (yfinal) in units (0 for a pickup image block).
w	Width of the image in units.
h	Height of the image in units.
rotation	Image rotation (**0 +90, +-180, +- 270**).
color	Color of the image (xyraster) in RGB #RRGGBB *[optional]*.
color-cmyk	Color of the image (xyraster) in a percentage of CMYK as a space delimited array: i.e “1.0 0.0 0.0 0.0” *[optional]*.
color-spot	Color of the image (xyraster) as a color name if it is a spot color *[optional]*.
scale	Image scale factor (**1–100**).
priority	Placement hierarchy (-127 <= priority <= 127).
clipx	x-offset for start of clipping area in units.
clipy	y-offset for start of clipping area in units.
clipw	Clipping area width in units.
clipd	Clipping area depth in units.

<rule>

Parent(s) **line**

Children none

Description The **rule** tag describes a horizontal or vertical rule. Although the rule is originally created by macros, in the XML output it is reflected in a tag structure in order that rules can be identified.

Note: *The rule tag is NOT used for underlines and/or trace marks. They are described as attributes to the* **t** *tag.*

Attribute	*Description*
x	X-offset from current cursor position in units.
y	Y-offset of rule relative to the base line (yfinal) in units.
w	Width of the rule in units.
d	Depth of the rule in units.
color	Color of the rule (#RRGGBB).
color-cmyk	Color of the rule in a percentage of CMYK as a space delimited array i.e “1.0 0.0 0.0 0.0”.
color-spot	Color of rule as a color name if it is a spot color *[optional]*.

<shape>

Parent(s) **line**

Children none

Description The **shape** tag describes a drawing shape that has been created by macros/ tabular for tinting and coloring boxes and other assorted shapes.

Note: *The attributes listed below apply to all types of shapes unless otherwise noted.*

Attribute	*Description*
name	Type of shape (rectangle, oval, gradient, lineto, custom).
x	X-offset from current position on line.
y	Y-offset of shape relative to the base line (yfinal) in units.
color	Shape color in RGB (#RRGGBB) *[optional]*.
color-cmyk	Shape color in a percentage of CMYK as a space delimited array; i.e. “1.0 0.0 0.0 0.0” *[optional]*.

Attribute	*Description*
color-spot	Shape color as a color name if it is a spot color *[optional]*.
priority	Visible placement order priority (-127 <= x <= 127) with text at "0".
altcolor	Alternate color in RGB (rectangle or oval shapes only) *[optional]*.
altcolor-cmyk	Alternate color in a percentage of CMYK (rectangle or oval shapes only) as a space delimited array; i.e. "1.0 0.0 0.0 0.0" *[optional]*.
altcolor-spot	Alternate color as a color name (rectangle or oval shapes only) if it is a spot color *[optional]*.
outline-type	Type of the outline: **non, sn, sni, sno, db, dbi, dbo, odb, odbi, odbo, idb, idbi, idbo, dsh, dshi, dsho** (rectangle or oval shapes only).
outline-size	Rule thickness of outline in units (rectangle or oval shapes only).
dash-length	For dashed outlines, length of the dash in units (rectangle or oval shapes only).
dash-space	For dashed outlines, length of the space between dashes in units (rectangle or oval shapes only).
corner-radius	Radius of the all four corners for rounded corners as a positive number (rectangle or oval shapes only).
angle	Halftone screen angle (rectangle or oval shapes only).
frequency	Halftone screen frequency (rectangle or oval shapes only).
x-radius	X-radius of the ellipse/circle in units (oval shape only).
y-radius	Y-radius of the ellipse/circle in units (oval shape only).
direction	Line Drawing direction: **left-right, right-left** (lineto only).
rule-size	Width of the rule in units (lineto only).
direction	Type and direction of gradient: **rect-left-right, rect-top-bottom, half-oval-left-right, half-oval-top-bottom, oval-left-right, oval-top-bottom** (gradient only).
start	Starting tint percentage (gradient only).
end	Ending tint percentage (gradient only).

<table>

Parent(s) **block**

Children **tgroup**

Description The **table** tag describes a table and is made up of a group of tags that declare the structure of a table in rows and columns. It is modeled after the CALS standard. Text is ultimately generated in the **entry** tag.

Note: *The* **table** *tag and all of the related sub-tags will be generated based on the structure of XPP tables, not on the original tag coding that produced the table.*

Attribute	*Description*
tabstyle	Table style (only if table is CALS; see *cals* attribute below) *[optional]*.
frame	Table frame (**all, none, top, bottom, topbot, sides).**
topbox	Thickness of top box rule in units *[optional]*.
botbox	Thickness of bottom box rule in units *[optional]*.
lsidbox	Thickness of left side box rule in units *[optional]*.
rsidbox	Thickness of right side box rule in units *[optional]*.
blgutter	Size of left (start) box gutter in units.
brbutter	Size of right (end) box gutter in units.
btgutter	Size of top box gutter in units.
bbgutter	Size of bottom box gutter in units.
cals	Indicates whether this is a CALS table (**true**, **false**).
colsep	Column separator (**0**=no, **1**=yes) *[optional]*.
rowsep	Row separator (**0**=no, **1**=yes) *[optional]*.
btcolor	Color of top box rule (#RRGGBB) *[optional]*.
btcolor-cmyk	Color of top box rule in a percentage of CMYK as a space delimited array i.e "1.0 0.0 0.0 0.0" *[optional]*.
btcolor-spot	Color of top box rule as a color name if it is a spot color *[optional]*.
bbcolor	Color of bottom box rule (#RRGGBB) *[optional]*.
bbcolor-cmyk	Color of bottom box rule in a percentage of CMYK as a space delimited array i.e "1.0 0.0 0.0 0.0" *[optional]*.
bbcolor-spot	Color of bottom box rule as a color name if it is a spot color *[optional]*.
blcolor	Color of left side box rule (#RRGGBB) *[optional]*.

Attribute	*Description*
blcolor-cmyk	Color of left side box rule in a percentage of CMYK as a space delimited array i.e "1.0 0.0 0.0 0.0" *[optional]*.
blcolor-spot	Color of left side box rule as a color name if it is a spot color *[optional]*.
brcolor	Color of right side box rule (#RRGGBB) *[optional]*.
brcolor-cmyk	Color of right side box rule in a percentage of CMYK as a space delimited array i.e "1.0 0.0 0.0 0.0" *[optional]*.
brcolor-spot	Color of right side box rule as a color name if it is a spot color *[optional]*.

<tgroup>

Parent(s) **table**

Children **colspec, thead, tbody**

Description The **tgroup** tag performs the same function as in the CALS model. It logically groups the thead and tbody sections.

Attribute	*Description*
tgroupstyle	XPP tabular style bundle name.
cols	Number of columns in the table.
mx_rows	Number of rows in the table.
tbox_row	Top of box above this row.
cals_intertgrule	CALS inter-tgroup rule (**norule, rowrule, boxruletop, boxrulebottom, boxruleboth**).
hj_mode	Horizontal position of table within block. Unitized value or one of: **left, right, center, inside, outside**.
lnmeas	Line measure in effect at beginning of table.
mx_rtl	**1**=columns flow right-to-left; **0**=columns flow left-to-right.
tbwidth	Table width.
tbdepth	Table depth (if table is broken, depth of table 'chunk' on this page).
tbtotvj	Table total vertical justification.
tbxposn	X-position of table relative to containing block.
tbyposn	Y-position of table relative to containing block.
tgroup	Tgroup number.

Attribute	*Description*
hdstyle_rows	Number of header rows (for 'hdrstyle').
min_colwidth	Minimum width of any column.
min_rowdepth	Minimum depth of any row.
hdr_rows	Number of running header rows.
colstyle	**0**=text width **1**=column width.
tblbrk	Break table at: (**row, cell, none**).
max_colwidth	Maximum width of any column.
ctbl_depth	Computed table depth.
yshift	yfinal shift due to pickups.
pgh_break	Continue headers on (**left, right, any**) page.
bkh_break	Continue headers on (**first, any**) block.
hdr_sup_rows	Suppress header rows (**true**, **false**).
lim_bywid	Limit widths by requested measure (**true**, **false**).
bbxcol	First column inside box rules.
ebxcol	Last column inside box rules.
bbxrow	First row inside box rules.
ebxrow	Last row inside box rules.
ob_hdr_rows	Continued Rows outside box rules.
tnoutput	Output table notes each: (**block, table**).
tnrenum	Renum table notes each: (**block, table**).
tnrefstyle	Table note reference style (**none, 0, an, 1, rn, 2, abc, 3, RN, 4, ABC, 5, ans, 6, anss, 7, anz, 8, anzz, 9, custom, 11, aio, 12, AIO, 13, ia, 15, ias, 16, iass, 17, iaz, 18, iazz, 19, ie, 20, ies, 21, iess, 22, iez, 23, iezz, 24, id, 25, ids, 26, idss, 27, idz, 28, idzz, 29**).
tnoutorder	Output table notes in reference order: **0**=yes, **1**=no.
eb_ep_row	Row to break on due to **《eb》** or **《ep》** (**0**=no break).
stubcols	Set stub column(s) to remaining measure (**true, false**).

<colspec>

Parent(s) **tgroup**

Children none

Description The **colspec** tag performs the same function as in the CALS model. It specifies the characteristics of a vertical portion (column) of a table.

Attribute	***Description***
colnum	Column number.
colname	Column name.
colwidth	Column width.
tbclwsp	Whitespace to the left of the column.
tbcrwsp	Whitespace to the right of the column.
tbclgut	Gutter to the left of the column.
tbcrgut	Gutter to the right of the column.
tbclrule	Width of rule to the left of the column.
tbcrrule	Width of rule to the right of the column.
tbcmeas	Width of the text measure in this column.
tbmxalnw	Maximum width of aligned text in this column.
tbctpos	X position of the text (measure) relative to the start of the table.
tbcxpos	X position of the column relative to the start of the table.
tbcr_rcolor	Color of rule to right of the column (#RRGGBB).
tbcr_rcolor-cmyk	Color of rule to right of the column in a percentage of CMYK as a space delimited array i.e "1.0 0.0 0.0 0.0".
tbcr_rcolor-spot	Color of rule to right of the column as a color name if it is a spot color *[optional]*.
tbcl_rcolor	Color of rule to left of the column (#RRGGBB).
tbcl_rcolor-cmyk	Color of rule to left of the column in a percentage of CMYK as a space delimited array i.e "1.0 0.0 0.0 0.0".
tbcl_rcolor-spot	Color of rule to left of the column as a color name if it is a spot color *[optional]*.

<thead>

Parent(s) **tgroup**

Children **row**

Description The **thead** tag performs the same function as in the CALS model. It logically groups the table header rows.

Note: *There are no attributes associated with this tag.*

<tbody>

Parent(s) **tgroup**

Children **row**

Description The **tbody** tag performs the same function as in the CALS model. It logically groups the table rows that are the main body.

Note: *There are no attributes associated with this tag.*

<row>

Parent(s) **thead, tbody**

Children **entry**

Description The **row** tag performs the same function as in the CALS model. It logically groups all the cells within one table row.

Attribute	*Description*
row	Row number (excluding header rows).
rowrel	Row number (within table on this page, including header rows).
tbrdepth	Row depth.
row_pos	Row top position, relative to the tbyposn value. The row top is the position just below the gutter above the row.
row_gutter	Size of row gutter (below this row).
x_space	Expansion space due to vertical justification.
tbntdepth	Depth of composed table notes.

The following row attributes are deprecated and may be removed in the future. The rule values may not be correct for all cells in the row (they might only be correct for the first cell). Instead, use the rule attribute values output with the **entry** tag.

Attribute	*Description*
rthk	Thickness of (topmost/only) row rule *[optional]* **(deprecated)**.
roff	Offset from top of row gutter to (topmost/only) row rule *[optional]* **(deprecated)**.
rcolor	Row rule color (RGB) *[optional]* **(deprecated)**.
rcolor-cmyk	Row rule color as a percentage of CMYK as a space delimited array; i.e "1.0 0.0 0.0 0.0" *[optional]* **(deprecated)**.
rcolor-spot	Row rule color as a color name if it is a spot color *[optional]* **(deprecated)**.
rthk1	Thickness of bottom row rule of double rule *[optional]* **(deprecated)**.
roff1	Offset from top of row gutter to bottom row rule *[optional]* **(deprecated)**.

<entry>

Parent(s) **row**

Children **group**

Description The **entry** tag is the basic building block of a table. It contains all of the lines that make up a table cell. Each entry tag will only contain group children.

Attribute	*Description*
col	Column number.
colsep	Column separator (**0**=no, **1**=yes) *[optional]*.
rowsep	Row separator (**0**=no, **1**=yes) *[optional]*.
namest	Name of leftmost column of span (name must match colname in **tgroup** or **colspec** tag) *[optional]*.
nameend	Name of rightmost column of span (name must match colname in **tgroup** or **colspec** tag) *[optional]*.
morerows	Number of additional rows in vertical span *[optional]*.
align	Text position within column or span (**left, center, right, justify**). Equivalent to tabular composition jstyle value (and similar to the **qdtype** attribute for the **line** tag).
halign	Horizontal cell alignment (**left, right, center**). Equvialent to tabular composition hstyle value.

Attribute	*Description*
valign	Vertical cell alignment (**top, bottom, middle**). Equivalent to tabular composition vstyle value.

rule_info	Attribute contains three space-delimited values in the form “style type nogut“ where *[optional]*:
	style = **1, 2** (single rule), **3** (double rule)
	type = **0** (urule), **1** (hrule), **2** (trule)
	nogut = **0** (add to row gutter), **1** (don’t add to row gutter)

rthk	Thickness of (topmost/only) row rule *[optional]*.
roff	Offset from top of row gutter to (topmost/only) row rule *[optional]*.
rcolor	Row rule color (RGB) *[optional]*.
rcolor-cmyk	Row rule color as a percentage of CMYK as a space delimited array; i.e “1.0 0.0 0.0 0.0” *[optional]*.
rcolor-spot	Row rule color as a color name if it is a spot color *[optional]*.
rthk1	Thickness of bottom row rule of double rule *[optional]*.
roff1	Offset from top of row gutter to bottom row rule *[optional]*.
alfleft	Flush-left alignment character(s) *[optional]*
al2	Second alignment position character(s) *[optional]*
al3	Third alignment position character(s) *[optional]*
al4	Fourth alignment position character(s) *[optional]*
alhright	Right-hang alignment character(s) *[optional]*

Appendix B

Samples

A Sample XML Export Output File

This section contains a one-page XPP division with the following content, in which a **<head>** is classified as a *h1* and a **<para>** is classified as a *p*:

```
<top>
    <head>This is Bold</head>
    <para>red</para>
</top>
```

XML Export will produce the following file:

```
<?xml version=″1.0″ encoding=″UTF-8″?>
<document version=“1.0”>
    <pages>
        <page relpgnum=″1″ psrnum=“1” p1=″0″ p2=″0″ p3=″0″
          p4=″1″ p5=″0″ p6=″0″ pagetype=“right”
          plname=″envintro″ px=″0″ py=″0″ fsx=″66″ fsy=″80″
          lastpage=″true″ pfexcept=″true″>
            <stream type=″main″ sx=″0″ sy=″0″ ssx=″0″ ssy=″0″4
              groupnum=″1″ svjmode=″0″ epfilltype=″0″>
                <block bx=″4″ by=″4″ bsx=″44″ bsy=″64″ bisy=″64″
                  btextlen=″63.2″>
                    <group class=“h1”pclass=“” style=“h1
                      dh=“/top/head” style=“h1”>
                        <line xfinal=″0″ yfinal=″10″ prelead=″4″
                          lnwidth=″43″ qdtype=″center″>
                            <tag name=“top”/>
                            <tag name=“head”/>
                            <t style=“h1”>This is Bold
                                <endtag name=“head”/></t>
                        </line>
                        </group>
                        <group class="p" pclass="" dh="/top/para"
                          style="p">
                            <line xfinal="15" yfinal="18" prelead="0"
                              lnwidth="42" qdtype="left">
                                <tag name="para"/>
                                <t style="p">red</t>
                                <tag name="/para"/>
                                <tag name="/top"/>
                            </line>
                        </group>
                    </block>
                </stream>
            </page>
        </pages>

        <styles>
            <style name="p" font="Helvetica" size="10 color="#FF0000"
              cm="normal"/>
            <style name="h1" font="HelveticalBold" size="16"
              color="#FFFFFF" cm="normal"/>
        </styles>
    </document>
```

Sample XSL Fragments

This section demonstrates how to process various XML components described in the XML Output Format.

To reference the "font size" style value . . .

. . . based on the style name in the "t" tag, use the following XSL template:

```
<xsl:template match="t">
    <xsl:variable name="myname select="@style"/>
    <xsl:text>font-size:</xsl:text>
    <xsl:value-of select="//style[@name=$myname]/@size"/>
</xsl:template>
```

To generate an XHTML colorized text "font" element. . .

. . based on the XML component:

```
<line>
    <t color="#FF00FF">this is a color</t>
</line>
```

use the following XSL template:

```
<xsl:template match="t">
    <xsl:variable name="myname" select="@style"/>
```

```
        <xsl:element font>
            <xsl:attribute name=“font”>
                <xsl:value-of select=″//style[@name=$myname]/@size″/>
            </xsl:attribute>
        </xsl:element>
    </xsl:template>
```

To change the font size in XHTML . . .

. . . based on the <span> tag, use the following XSL template:

```
    <xsl:template match=“t@size]”>
        <xsl:variable name=″myname″ select=″@style″/>
        <xsl:variable name=″mystyle″
          select=″//style[@name=$myname]″/>
        <xsl:element name=“span”>
            <xsl:attribute name=“style”>
                <xsl:text>size </xsl:text>
                <xsl:value-of select=“$mystyle/@size”/>
                <xsl:text>;</xsl:text>
            </xsl:attribute>
        </xsl:element>
    </xsl:template>
```

To start an EDGAR HTML document . . .

. . . based on the XML component:

```
    <doc>
        <page>
            <stream> . . . </stream>
        </page>
    </doc>
```

use the following XSL templates:

```
    <xsl:template match=“/”>
        <html>
            <head>
                <title>EDGAR HTML</title>
            </head>
            <body>
                <xsl:apply-templates/>
            </body>
        </html>
    </xsl:template>

    <xsl:template match=“page”>
        <xsl:apply-templates/>
        <PAGE/>
    </xsl:template>
```

To change color in RTF . . .

based on the XML component:

```
<line>
    <t color="0000ff">This is blue text in the color table #1</t>
</line>
```

In RTF, the foreground and background colors use indexes into the color table to define a color. So, given an RTF color table:

```
{\colortbl;\red0\green0\blue0;\red0\green0\blue255;\red0\green255\
  blue255; . . .}
```

where color #1 is "blue", the desired output would be:

```
{\f1This is blue text in the color table}.
```

The XSL would look like:

```
<xsl:output method="text"/>

<xsl:template match="t">
    <xsl:variable name="myname" select="@style"/>
    <xsl:variable name="mystyle"
      select="//style[@name=$myname]"/>
    <xsl:text>{</xsl:text>
    <xsl:choose>
        <xsl:when test="$mystyle/@color=0000FF">
            <xsl:text>\f1</xsl:text>
        </xsl:when>
    </xsl:choose>
    <xsl:text>
        <xsl:valueof select="."/> </xsl:text>
    <xsl:text>}</xsl:text>
</xsl:template>
```

Appendix C

A Sample XML Export Conversion to ePub

Overview

The EPUB conversion sample (program) may be used as a reference in designing your own targeted output format. It is made up of the following deliverables:

- xy2epub_DEMO.pl, a Perl script which is an XPP job/division tool.
- epub.xsl, an XSLT 2.0 stylesheet which converts divxml output to EPUB Version 2 format.
- epub.css, a CSS file which contains EPUB styles.

Note: This sample provides only the minimum XSLT translation required to produce a sample EPUB document. It is expected that you will customize this sample to meet your own unique implementation requirements.

To launch the conversion program from the SDL XPP Pathfinder, right click on a DIV (or JOB) and select **More Tools-> xy2epub_DEMO.pl**. The program will display progress windows showing each step in the conversion process.

The conversion program will perform the following operations:

- Ensure that the divxml program is available, as well as the XSLT 2.0 engine (xppxslt).
- Use unique filenames for creating the EPUB so that multiple parallel EPUB conversions are possible.
- Execute divxml to create an XML representation of the job/division.
- Run an XSLT transformation to convert the textual content of the job/div to xhtml files; break the job/div into sections if possible,

otherwise convert each page to a separate Section in the EPUB. Each section becomes a separate file in a folder reserved for text content within the EPUB.

- Run a separate XSLT transformation to search the divxml output for images. If found, create a separate folder for image content within the EPUB and copy the images to the image folder.
- Create the EPUB, including appropriate metadata. The EPUB Version 2 format is a zip file with the following contents:

```
<epub.zip>:
  META-INF
    container.xml
  OPS
    content.opf
    cover.png
    toc.ncx
    Images
      *.{jpg, png, gif}
    Style
      epub.css
    Text
      *.xhtml
```

For information regarding the EPUB specifications, see:

http://www.idpf.org/specs.htm

For a plug-in to the Firefox browser which renders EPUB documents, see:

http://www.epubread.com

For a tool which checks the validity of EPUB documents, see:

http://code.google.com/p/epubcheck

For an open-source eBook management tool (including EPUB), see:

http://www.calibre-ebook.com

SDL Group
201 Edgewater Drive
Wakefield, MA 01880-6216
www.sdl.com