Technical Working Group Reports

A proposed standard for specials

Tomas G. Rokicki

Introduction

This note presents the current state of the draft standard, as presented by the author and Michael Sofka at the standards session at TUG'95.

1 Identifying syntax

Standard specials shall be syntactically identified by beginning with a colon (':'). All specials beginning with a colon shall follow the guidelines established here.

Any special beginning with a colon, followed by an agreed keyword with agreed semantics, shall be interpreted according to the rules set out in this document and according to the agreed semantics of that keyword.

Any special beginning with two consecutive colons shall be considered an experimental special. It will be interpreted following the syntax and scoping semantics specified in this document, but individual drivers are free to interpret these specials however they wish. This convention allows experimentation with specials in conjunction with the scoping mechanism described here.

2 Syntax

Standard specials shall consist of a sequence of the 95 printable ASCII characters plus the tab character. Tabs will always be interpreted as spaces. Each standard special shall begin with a colon, optionally followed immediately by another colon. Following this shall be a sequence of elements separated by whitespace. Whitespace is also allowed between the colon (or optional colon) and the first element.

Whitespace shall consist of any number of tab or space characters.

The elements shall fall in the following categories: symbol, keyword, key/value pair.

We also define the syntax of numbers, dimensions, and lists, for convenience.

A symbol is any sequence of characters. If the symbol consists of only the 95 printable ASCII characters, and does not contain a double quote, equals sign, space, tab, backslash, or comma, it can be

specified by the sequence of characters that comprise it; when a symbol is so specified, it is called a simple symbol. Otherwise, it can be specified by enclosing the exact characters in a pair of double quotes. Any double quote or backslash within the symbol must be preceded by a backslash. When a symbol is specified through the use of double quotes, it is a quoted symbol.

A keyword is a simple symbol.

A number is a simple symbol that consists of an optional negative sign followed by a sequence of at least one digit, with one optional period ('`.`') included in the sequence of digits.

A dimension is a symbol that obeys the rules of a number except that it is followed immediately (with no intervening whitespace) by one of the following pairs of characters: '`bp`', '`cm`', '`dd`', '`in`', '`mm`', '`pc`', '`pt`', '`sp`'.

An equals sign shall appear only as part of a quoted symbol, or to separate a keyword and some other symbol forming a key/value pair.

A list is a sequence of symbols separated by commas with no intervening whitespace.

Figure 1 gives a BNF formulation for standard specials. Square brackets enclose optional components. Quotes enclose literal characters. Parentheses group. Angle brackets enclose nonterminals. An asterisk represents zero or more occurrences; a plus sign indicates at least one and possibly more occurrences. Vertical bars indicate choice.

In general, case is significant in specials.

3 Specials scoping and infrastructure

Specials exist in the DVI file as just another sequential command. Yet, often we desire a special to have a scope. One use of specials is to modify the way certain commands in the DVI file are interpreted. For instance, a special might indicate that all subsequent characters until an overriding special shall be rendered in a specific color. Another special might indicate that a certain set of pages is to be printed on different media. Yet a third special might indicate that the background color of the entire document should be mauve.

For this reason, we are introducing standard scoping semantics for standard specials.

3.1 Flat DVI files

Ideally, scoping could always be handled simply by placing an appropriate special at the beginning and end of each scoped region, and no further action would need to be taken. For various reasons, as we shall describe, this is not always possible or convenient. Nonetheless, such a 'flat' scoping would serve as an ideal model for the driver writer, and its semantics should form a base on which more complex scoping rules can be built.

In this section, we describe how flat DVI files are interpreted. At the minimum, each driver should be capable of handling flat DVI files.

A flat DVI file is one that can be processed (with respect to specials) with no prescanning or preprocessing. Each special is located syntactically where it belongs semantically. In addition, assuming the beginning of the first page has been scanned for document global specials, each page can be processed and reprocessed independently of any other and in random order, skipping arbitrary sequences of pages. Thus, a flat DVI file is ideal for quick browsing and previewing.

3.1.1 Object specials

The first category of specials, called *object* specials, is those specials that themselves render objects to the page, but do not affect the rendering of other objects. One such example is a special that indicates that a particular graphical figure should be rendered at a particular location on the page.

All object specials shall begin with the keyword '`object`'.

Object specials take several implicit parameters that affect how they are rendered. These implicit parameters include the current DVI location on the page, and the DVI magnification.

Unless otherwise specified for a particular object special, all object specials shall be interpreted such that their lower left-hand corner is rendered at the current DVI location. In addition, all object specials shall be scaled by the DVI magnification in effect.

Object specials cannot take the optional scoping keywords described in section 3.2.

The initial syntax for object specials is as follows:

```
<object-specifier> := ':' [':'] <w> 'object' <w>
```

3.1.2 Attribute specials

The second category of specials is attribute specials. These specials affect the way the page is rendered. Normally, an attribute special affects the rendering state for subsequent DVI commands until overridden by another attribute special that affects the same rendering attribute. We shall discuss how our scoping mechanism can change these rules in section 3.2.

All attribute specials begin with the keyword '`attribute`'.

```
<standard-special> := ':' [':'] [<whitespace>] (<element> <whitespace>)+
<element>          := <key-value>
                   |  <simple-symbol>
<whitespace>       := (tab | ' ')+
<w>                := <whitespace>
<key-value>        := <simple-symbol> ['=' <symbol-or-list>]
<symbol-or-list>   := <symbol> | <list>
<list>             := <symbol> (',' <symbol>)*
<symbol>           := <simple-symbol>
                   |  <quoted-symbol>
<simple-symbol>    := (<printable-char-except-space,tab,comma,backslash,equals,doublequote>)+
<quoted-symbol>    := '"' (<quoted-char>)* '"'
<quoted-char>      := <space-or-printable-char-except-backslash,doublequote>
                   |  '\' '\' | '\' '"'
<number>           := ['-'] (<digit>)+ [. (<digit>)*]
                   |  ['-'] . (<digit>)+
<dimension>        := <number> <unit>
<unit>             := 'bp' | 'cm' | 'dd' | 'in' | 'mm' | 'pc' | 'pt' | 'sp'
```

Figure 1: A BNF grammar for specials

One interesting case should be mentioned here. In a flat DVI file, it is possible for an attribute special to contain the scoping keyword '`pop`'. If the '`attribute`' keyword in a flat DVI file is followed immediately by the keyword '`pop`', then that corresponding attribute in the rendering state is set (back) to its initial state. (This is necessary because, when flattening scoped specials, the initial state might not be known for all attributes; this provides a convenient way to access that default initial value.)

The initial syntax for attribute specials is as follows:

```
<attribute-specifier> := ':' [':'] <w>
                         'attribute'
                         (<w> <scope>)* <w>
<scope>               := 'push' | 'pop'
                      |  'page' | 'global'
```

3.2 Scoped specials

Sometimes a special must occur at a syntactic location different from where it semantically affects the rendering state. One example of this is where an attribute affecting the background color of the paper is specified as part of the running text of a document, in a document with headers. Normally, this special will follow the headline in the DVI file, because TEX's output routine typically ships the header before the whatsits attached to the body text. So by the time the DVI driver sees the special, it has probably already rendered the header, so it may be difficult or inconvenient to change the background color of the sheet at this point.

Another example is when a colored paragraph is broken across a page boundary, and the DVI driver wishes to render only the second page, without scanning the first page. In the absence of specials, this is easily done. However, if there is only a single special specifying the red color at the beginning of the paragraph (on page one), there is no indication in page two that the color should still be red.

As a final example, consider the use of nested attribute specials. One word in a blue paragraph is to be colored green. The special at the end of the green word indicates that the 'previous' color state should be restored. In this case, the special at the beginning of the paragraph, indicating that the paragraph should be blue, is also semantically visible at the end of the green word.

The scoping rules we introduce in this section introduce a scoping mechanism, and define how specials that use this mechanism, and thus cause a DVI file to be scoped (rather than flat), can be transformed into flat specials for easier rendering.

3.2.1 Stacks

The first mechanism is a convenient 'stack' mechanism that allows the previous state of a particular attribute to be easily restored. The two keywords that indicate this mechanism should be used are '`push`' and '`pop`'.

If the keyword '`attribute`' in a special is followed by the keyword '`push`', then the current state of that attribute is pushed onto a stack internal to the DVI processor. Then, the remainder of the special is used to determine the new value of the attribute.

If the keyword '`attribute`' in a special is followed by the keyword '`pop`', then the previously saved value of the attribute is restored. Any actual attribute value specified in the special is ignored.

If the stack is empty when a '`pop`' special is encountered, then the value of the attribute is set to the default initial value for that attribute.

Attribute specials that use neither push nor pop are still fully legal; they affect the current setting of the attribute, but do not affect the stored stack.

Note that each attribute has its own independent stack. Thus, the following sequence of specials is perfectly legal:

```
:attribute push color red
:attribute push trap true
:attribute pop color
:attribute pop trap
```

DVI drivers should support a stack depth for each attribute of at least twenty levels.

3.2.2 Page

Some specials affect the page, or paper, or media, rather than the rendering of characters or rules. Specials that specify the paper size or background color are examples of these. Such specials should ordinarily occur before any characters or rules or object specials on the page itself; such is the case for flat DVI files.

As discussed before, it is sometimes inconvenient or difficult to ensure that these specials actually occur at the beginning of the page itself. Thus, we define the scoping keyword '`page`' that indicates this special should semantically appear at the beginning of the page. Thus, if a page contains a single (hypothetical) background color attribute, specified as

```
:attribute page backgroundcolor mauve
```

anywhere on the page, then the page should be rendered with a mauve background.

Note that there is no predefined correlation between attributes that are specified page and those that are local. Thus, a special such as

```
:attribute backgroundcolor mauve
```

that occurs between two characters on a page makes little sense. Indeed, where the special obviously affects the entire page, but it is not specified with a page keyword, the operation of the DVI driver shall be undefined.

If multiple page attribute specials for the same attribute appear on a page, they shall all be semantically moved to the top of the page — in the same order as they occur on the page.

3.2.3 Global

Some specials affect the document as a whole, or it is desired that they affect the document as a whole. Specials that define a paper size are one example of these. Generally, such specials should occur before any characters or rules or object specials on the first page.

As discussed before, it is sometimes inconvenient or difficult to ensure that these specials actually occur at the beginning of the page. Thus, we define the scoping keyword '`global`' that indicates this special should semantically appear at the beginning of the entire document. Thus, if a document contains a single (hypothetical) paper size attribute, specified as

```
:attribute global papersize 8.5in 11in
```

anywhere in the document, then the entire document should be rendered on $8.5'' \times 11''$ paper.

For pragmatic reasons (we may not want to, or be able to, prescan the entire document), we require that such global specials occur somewhere on the first page in order to take effect.

Note that there is no predefined correlation between the attributes that are specified global and those that are local or page-specific.

If multiple global attributes for the same attribute appear in a document, all shall be semantically moved to the front of the document — in the same order as they occur in the document.

3.3 Flattening process

The scoping rules are concentrated on the distinction between a special's syntactic and semantic location. The process of flattening a DVI file resolves these differences, by eliminating all stack operations (except for defaulting pops) and by moving all specials to their logical semantic location. This operation converts a scoped DVI file to a flat DVI file.

When flattening occurs, object specials remain in their original location.

For instance, consider a scoped DVI file that appears as follows:

```
Page 1:
   <text>
   :attribute push color red
   <text>
   :attribute global backgroundcolor mauve
Page 2:
   <text>
   :attribute push color green
   <text>
   :attribute color blue
   <text>
   :attribute page papersize 8.5in 11in
   :attribute pop color
   <text>
Page 3:
   <text>
   :attribute pop color
   <text>
```

This would be flattened into the following flat DVI file:

```
Page 1:
   :attribute global backgroundcolor mauve
   <text>
   :attribute color red
   <text>
   :attribute pop color
Page 2:
   :attribute page papersize 8.5in 11in
   :attribute color red
   <text>
   :attribute color green
   <text>
   :attribute color blue
   <text>
   :attribute color red
   <text>
   :attribute pop papersize
   :attribute pop color
Page 3:
   :attribute color red
   <text>
   :attribute pop color
   <text>
```

If a page reversal program that obeys the specials is run, the following DVI file would result. Note that the only difference is that the global specials are moved from the original first page to the new first page.

```
Page 3:
   :attribute global backgroundcolor mauve
   :attribute color red
   <text>
   :attribute pop color
   <text>
Page 2:
   :attribute page papersize 8.5in 11in
   :attribute color red
   <text>
   :attribute color green
   <text>
   :attribute color blue
   <text>
   :attribute color red
   <text>
   :attribute pop papersize
   :attribute pop color
Page 1:
   <text>
   :attribute color red
   <text>
   :attribute pop color
```

This flattening can be performed with a special DVI to DVI processor (which we hope to provide). The use of such a preprocessor will allow fancy scoped specials to be used in an environment that supports only flat specials. For DVI files that are intended to be widely distributed and portable, such a flattening should probably be done.

This flattening can also be performed dynamically, as a DVI file is being created or read from disk, so long as single-page scanning is available. (We hope to provide code for this as well, that will allow easy integration of scoped specials into previewers and printer drivers with a minimum of effort.)

4 Proposed object specials

The primary and most important proposed object special is that for encapsulated PostScript file inclusion.

4.1 EPSF inclusion

The syntax for the encapsulated PostScript inclusion special is as follows:

```
<epsf-special>  := <object-specifier>
                   ('epsf' | 'psfile') '='
                   <symbol>
                   (<w> <epsf-keyword>)* [<w>]

<epsf-keyword> := 'width=' <dimen>
                | 'height=' <dimen>
                | 'scale=' <number>
                     [',' <number>]
                | 'clip=' ('on' | 'off')
                | 'boundingbox='
                     <number> ',' <number> ','
                     <number> ',' <number>
```

In this special, the symbol following the keyword '`epsf`' or '`psfile`' is interpreted as a filename

containing a encapsulated PostScript file for inclusion. This file should follow the Adobe standards for EPSF files; otherwise the effects of this special are undefined.

Positioning occurs as follows. First, a bounding box is determined. If one is specified in the special it is used. Otherwise, if none is specified in the special and the keyword is given as '`epsf`', the EPSF document itself is scanned for a bounding box.

The bounding box values at this point, interpreted in PostScript units, map the region of the illustration that will be included.

The TEX width is set to the horizontal size of the bounding box, and the TEX height is set to the vertical size of the bounding box, both interpreted as 72 units to the inch. Consideration of any width, height, and scale parameters may further affect these values, as described below.

Next, the optional width and height specifiers are considered. If neither is given, this step is omitted. If both are given, their values replace the TEX height and width set before. If only one is given, it replaces the corresponding TEX value, and the other TEX value is set to preserve the aspect ratio.

Next, if the '`scale`' keyword is given, the TEX height and width are further modified. If only one numeric parameter is given, both the width and height are multiplied by this parameter. Otherwise, the width is multiplied by the first parameter and the height is multiplied by the second parameter.

Finally, the width and height values are multiplied by the current DVI magnification in effect.

The resulting values describe the size of a rectangle on the DVI page. The lower left hand corner of this rectangle is positioned at the current DVI location. The geometric mapping from the original bounding box to this rectangle defines the scaling that is performed on the EPSF file when it is rendered.

If the clipping keyword is specified to be '`on`', or if no clipping keyword is specified, the rendering of the EPSF file will be constrained to fall within the DVI rectangle calculated above. Otherwise, no clipping will be performed, and if the EPSF file renders outside its bounding box borders, portions of the image will also be rendered outside the DVI rectangle.

4.1.1 PS *vs.* EPSF

The effects of the keyword '`epsf`' and the keyword '`psfile`' are almost identical — with one minor difference. If '`epsf`' is specified, then the bounding box keyword and values are optional; if they are not specified, the bounding box is read from the EPSF file. If '`psfile`' is specified, the bounding box must be specified; the operation of the special is undefined if no bounding box is specified. (The reasoning behind this seeming inconsistency is partially political and partially religious.)

In order to include a normal EPSF image in its entirety, either the '`psfile`' or the '`epsf`' keyword can be used; if a bounding box value is specified in the special command, the semantics are equivalent.

If only a rectangular subportion of an EPSF image or PS page is to be rendered, then the '`psfile`' keyword should be specified, along with a bounding box describing precisely what portion of the image should be included. Clipping should be turned on to ensure that the rest of the image does not show up outside the DVI rectangle.

4.1.2 Bounding box

The bounding box is specified as a comma-separated list of numbers. These numbers are interpreted as PostScript units. There are 72 PostScript units to the inch. The four numbers are, in sequence, the x-coordinate of the lower left corner, the y-coordinate of the lower left corner, the x-coordinate of the upper right corner, and the y-coordinate of the upper right corner. All four must be specified.

Note that we do not restrict these numbers to be integers. While Adobe requires these to be integers in their Document Structuring Conventions, some applications generate floating point numbers. In addition, the higher precision afforded by floating point might be useful in some circumstances.

4.1.3 Scaling

The scaling parameter consists of one or two numbers separated by a comma. These are used to specify a scaling ratio for an EPSF image. If only one number is specified, the scaling preserves the aspect ratio of the image.

4.1.4 Clipping

Clipping can be set to either '`on`' or '`off`'. The default is '`on`'.

4.1.5 Rotation

Rotation is not currently supported, although it is currently under discussion.

5 Proposed attribute specials

5.1 Color

Under development.

5.2 Background color

Under development.

5.3 Paper size

Paper size and orientation are important characteristics of the document, and should be specified in the document itself rather than on the driver command line. The syntax for the paper size special is:

```
<papersize-special> := <attribute-specifier>
                       'papersize'
                       [<w> <dimension>
                        <w> <dimension>] [<w>]
```

The value (composed of two dimensions) may be omitted only if one of the specified scoping operators is '`pop`'.

The dimensions specify horizontal and then vertical size.

As is conventional, the DVI origin is located one inch down and one inch from the left of the top left corner of the paper.

A typical papersize special is

```
:attribute global papersize 8.5in 11in
```

To specify that landscape letter size is in effect for the current page forward, use

```
:attribute page push papersize 11in 8.5in
```

followed by

```
:attribute page pop papersize
```

on the page you wish to return to portrait.

6 Document history

This section shall record a history of the changes to this document.

22 September 1995: Originated by Tomas Rokicki on the basis of extensive discussions at TUG'94 and TUG'95, discussion on the TWG-DVI mailing list, and discussion at a meeting at MSRI in December of 1994.

10 January 1996: Edited for publication in *TUGboat* by Robin Fairbairns, Barbara Beeton, and Tomas Rokicki.

◇ Tomas G. Rokicki
725B Loma Verde
Palo Alto, CA 94303
USA
Email: `rokicki@cs.stanford.edu`