



The Type 42 Font Format Specification

Adobe Developer Support

Technical Note # 5012

1 March 1993

Adobe Systems Incorporated

Corporate Headquarters
1585 Charleston Road PO Box 7900
Mountain View, CA 94039-7900
(415) 961-4400 Main Number
(415) 961-4111 Developer Support
Fax: (415) 961-3769

Adobe Systems Europe B.V.
Europlaza
Hoogoorddreef 54a
1101 BE Amsterdam Z-O, Netherlands
+31-20-6511 200
Fax: +31-20-6511 300

Adobe Systems Eastern Region
24 New England
Executive Park
Burlington, MA 01803
(617) 273-2120
Fax: (617) 273-2336

Adobe Systems Japan
Swiss Bank House 7F
4-1-8 Toranomon, Minato-ku
Tokyo 105, Japan
+81-3-3437-8950
Fax: +81-3-3437-8968

Copyright © 1993 by Adobe Systems Incorporated. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher. Any software referred to herein is furnished under license and may only be used or copied in accordance with the terms of such license.

PostScript is a registered trademark of Adobe Systems Incorporated. All instances of the name PostScript in the text are references to the PostScript language as defined by Adobe Systems Incorporated unless otherwise stated. The name PostScript also is used as a product trademark for Adobe Systems' implementation of the PostScript language interpreter.

Any references to a "PostScript printer," a "PostScript file," or a "PostScript driver" refer to printers, files, and driver programs (respectively) which are written in or support the PostScript language. The sentences in this book that use "PostScript language" as an adjective phrase are so constructed to reinforce that the name refers to the standard language definition as set forth by Adobe Systems Incorporated.

Adobe, PostScript and the PostScript logo are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions. TrueType is a trademark and Apple, Macintosh, and LaserWriter are registered trademarks of Apple Computer, Incorporated. Windows is a trademark and Microsoft is a registered trademark of Microsoft Corporation. Other brand or product names are the trademarks or registered trademarks of their respective holders.

This publication and the information herein is furnished AS IS, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes and noninfringement of third party rights.



Contents

List of Tables v

The Type 42 Font Format Specification 1

- 1 Introduction 1
- 2 The Type 42 Font Format 1
 - Type 42 Font Comment Lines 2
 - The Type 42 Font Dictionary 3
 - Implications of The Glyph Coordinate System 5
- 3 Identifying PostScript Language Interpreters with TrueType Rasterizers 5
- 4 Conversion Issues 6
 - The **FontInfo** Dictionary 6
 - The **sfnts** Array 6
 - Generating The **CharStrings** Dictionary 8
 - Generating the **Encoding** Vector 8
 - Generating Unique Identifiers 9
 - Required TrueType Tables 10
 - Known Bugs 10
 - Example Type 42 font program 10

Index 13



List of Tables

Table 1	Entries in all types of font dictionaries	3
Table 2	Additional entries in all base fonts (FontType not 0)	4
Table 3	Additional entries in Type 42 fonts	4

The Type 42 Font Format Specification

1 Introduction

This document describes a PostScript™ font format which can be used to download TrueType™ fonts to PostScript language interpreters which contain a TrueType rasterizer.

A Type 42 font dictionary contains the **sfnts** keyword, whose value is a PostScript language representation of the data in a TrueType font. Other entries in the Type 42 font dictionary permit the PostScript language interpreter to handle the font in a manner similar to a Type 1 font, and to make the TrueType font data in the **sfnts** entry available to the TrueType rasterizer.

This method yields better performance and quality of output than can be achieved by converting a TrueType font to a Type 1 or 3 font program, since the translation cannot be exact.

This document describes only the format of a Type 42 font program and how it may be created from a TrueType font. The TrueType specification is available in the document: *The TrueType Font Format Specification 1.0*; #M0825LL/A, available from the Apple® Programmers and Developers Association (APDA). Microsoft's TrueType specification is available electronically in two locations (as of February 1993): via anonymous FTP on the Internet on the ftp.uu.net host, in the /vendor/Microsoft®/TrueType-Info directory; and on CompuServe in the Microsoft Developer's Support forum. The documents are in two formats: a Windows™ 3.1 winhelp version and in a Word for Windows 2.0 version.

2 The Type 42 Font Format

The TrueType font format was originally developed by Apple Computer, and is currently supported by the Macintosh® and Windows 3.1 operating environments. Until recently, documents containing TrueType fonts could be sent to PostScript language interpreters by downloading the TrueType rasterizer to interpreters with 680X0-class controllers, or by converting the font into a

PostScript Type 1 or 3 font program. Both of these methods have disadvantages: the TrueType rasterizer is large, and conversion to Type 1 or Type 3 is not exact.

Some PostScript language interpreters now include a TrueType rasterizer; they can be identified from the PostScript Printer Description (PPD) file (see section 3). For a TrueType font to be recognized by a PostScript interpreter, it must be enclosed in a PostScript font dictionary, with the binary TrueType font expressed as an array of strings as the value for an **sfnts** keyword.

Note Although the keyword name **sfnts** is derived from the Macintosh resource type used for TrueType fonts, TrueType fonts from the Windows environment can be converted in an identical manner.

A Type 42 font is a base font and shares all the properties of base fonts as documented in the *PostScript Language Reference Manual, second edition*. In particular, presence of a unique identifier such as an **XUID** facilitates bitmap caching for a Type 42 font just as it does for any other type of base font. When Type 42 fonts are permanently downloaded to a hard disk connected to a PostScript language interpreter, only the *charstrings* actually referenced in the print job will be read into VM, thus saving memory (see section 4.2). Also, glyphs can be modified or added to the Type 42 font by using user-defined PostScript language procedures (See section 5.6.3 in *The PostScript Language Reference Manual, second edition*).

2.1 Type 42 Font Comment Lines

The first line of a Type 42 font program shall be:

```
%!PS-TrueTypeFont-TTVersion-MfrRevision
```

where *TTVersion* is the TrueType version number of the font (specified in the header); *MfrRevision* is the font manufacturer's revision number of the font. This line helps downloaders to easily identify TrueType/Type42 fonts on the printer's hard disk.

The first portion of the comment line: *%!PS-TrueTypeFont*, is required for compatibility with the Apple LaserWriter[®] IIf and IIfg, and NTR printers' requirements for disk-based Type 42 fonts. If the font has the required portion of the comment line and it conforms to the requirements listed in section 4.2, the embedded TrueType glyph data will be accessed from disk on demand rather than reading the entire font into VM.

Another useful and recommended comment line specifies the VM usage:

```
%%VMusage: MinMemory MaxMemory
```

where *MinMemory* and *MaxMemory* specify the minimum memory needed for the font and how much is needed if the font is downloaded first (these numbers are not necessarily the same; see *The Adobe™ Type 1 Format, version 1.1*, Addison Wesley, 1990). This comment is not used by the PostScript interpreter, but is useful for application programs. The PostScript language operator **resourcestatus** can be used to obtain the VM requirements for a font resource.

The values for the **VMusage** comment can be derived from a TrueType font which contains a post table (which contains information useful for PostScript language printing). For downloading purposes, the size of the font can be used as an estimate for the values for **VMusage**. However, a properly constructed Type 42 font on a printer's hard disk will generally require only a percentage of the VM required for the downloadable version of the font.

2.2 The Type 42 Font Dictionary

Table 1 lists entries common to all types of font dictionaries. Table 2 lists additional key-value pairs that are meaningful in all *base* fonts. Table 3 lists additional key-value pairs that are meaningful in Type 42 fonts. See the corresponding tables 5.1 through 5.3 in *The PostScript Language Reference Manual, second edition*.

Table 1 *Entries in all types of font dictionaries*

<i>Key</i>	<i>Type</i>	<i>Description</i>
FontType	integer	<i>(Required)</i> Value must be 42.
FontMatrix	array[6]	<i>(Required)</i> Transforms the glyph coordinate system into the user coordinate system. Type 42 fonts, unlike Type 1 fonts, are usually defined in terms of an identity transform, so the value of FontMatrix should be [1 0 0 1 0 0]. See section 2.3 for a discussion of the implications of this choice of coordinate system. FontMatrix must be a literal array.
FontName	name	<i>(Optional)</i> The font program's name, derived from the TrueType name table.
FontInfo	dictionary	<i>(Optional)</i> See <i>The PostScript Language Reference Manual, second edition</i> , Table 5.4, page 268.

Table 2 Additional entries in all base fonts (**FontType** not 0)

Key	Type	Description
Encoding	array[256]	(Required) An array of 256 glyph names ordered by glyph code value. The encoding is most likely to be either the Apple standard encoding or the Windows ANSI encoding, but other encodings will occur. See section 4.4. Encoding must be a literal array.
FontBBox	array[4]	(Required) See description in <i>The PostScript Language Reference Manual, second edition</i> , Table 5.2. Derived from the TrueType head table.
UniqueID	integer	(Optional) See section 4.5.
XUID	array	(Optional) Array of integers that uniquely identifies this font or any variant of it. See section 5.8 of <i>The PostScript Language Reference Manual, second edition</i> . XUID must be a literal array.

Table 3 Additional entries in Type 42 fonts

Key	Type	Description
PaintType	integer	(Required) 0 for filled glyphs; 2 for stroked glyphs.
StrokeWidth	number	(Optional) The width of the line used to stroke outline fonts (PaintType = 2) in glyph coordinates. This is interpreted in glyph space; see section 2.3.
Metrics	dictionary	(Optional) Width and sidebearing information for writing mode 0. Not normally present in the original definition of a font; adding this dictionary to a font overrides the widths and sidebearings encoded in the glyph definitions in the TrueType font. This dictionary will only affect Type 42 fonts in version number 2013 and greater of the PostScript interpreter. The values in this dictionary are interpreted in glyph space; see section 2.3.
Metrics2	dictionary	(Optional) Width and sidebearing information for writing mode 1. In general this dictionary is only interpreted by Level 1 devices with composite font extensions and all Level 2 devices; for Type 42 fonts it is only recognized by PostScript interpreter version 2013 and greater. The values in this dictionary are interpreted in glyph space; see section 2.3.
CDevProc	procedure	Algorithmically derives global changes to a font's metrics. See <i>The PostScript Language Reference Manual, second edition</i> , p. 277. CDevProc works the same in a Type 42 font as in a Type 1 font, aside from the different glyph coordinate system; see section 2.3.

CharStrings	dictionary	<i>(Required)</i> Associates glyph names with glyph descriptions. If an entry's value is an integer, it is used as an index into the TrueType loca table, which contains the byte offsets of glyph definitions in the TrueType glyf table. If the value is a procedure (executable array or packed array), it is interpreted as described in section 5.6.3 of <i>The PostScript Language Reference Manual, second edition</i> . This dictionary must have an entry whose key is /.notdef .
sfnts	array	<i>(Required)</i> An array of one or more PostScript language string objects containing the binary TrueType font. (see section 4.2 for information on the constraints and format).

2.3 Implications of The Glyph Coordinate System

As indicated in Table 1, a Type 42 font's glyph coordinate system is typically defined as an identity transform. This is in contrast to a Type 1 font, whose glyph coordinate system is typically defined at a 1000 unit scale relative to user space.

This difference has implications regarding the interpretation of font dictionary entries whose values are defined in glyph space. If a PostScript language program adds or changes such entries in a font dictionary, it must choose values that are appropriate to the font's glyph coordinate system. In particular, values that would be appropriate for a Type 1 font will be 1000 times too large for a Type 42 font.

The font dictionary entries for which this issue arises include:

- The value of **StrokeWidth** (when **PaintType** has been set to 2);
- The contents of the **Metrics** and **Metrics2** dictionaries;
- The operands and results of the **CDevProc** procedure;
- The values of **UnderlinePosition** and **UnderlineThickness** in the **FontInfo** dictionary.

3 Identifying PostScript Language Interpreters with TrueType Rasterizers

PostScript language interpreters with TrueType rasterizers can be identified from the following entry in the device's PPD file (version 4.1 of the specification):

```
*TTRasterizer: RasterizerOption
```

where *RasterizerOption* can be any of the following:

None	No TrueType rasterizer is present, and the device is not capable of receiving a downloadable rasterizer. To use a TrueType font on this interpreter, it must be converted to a Type 1 or Type 3 font.
Accept68K	No TrueType rasterizer is built-in, but the device has a 680X0-based controller and enough memory to accept a downloadable TrueType rasterizer. (The code to accomplish this is proprietary to Apple Computer, and is not generally available).
Type42	The device has a Type 42 TrueType rasterizer in ROM.

A PostScript language program can determine whether a Level 2 device supports Type 42 fonts (no Level 1 devices support Type 42) by executing:

```
42 /FontType resourcestatus {pop pop true} {false} ifelse
```

which pushes true or false on the stack depending on whether Type 42 font support is present.

4 Conversion Issues

The following sections discuss issues related to converting a TrueType font into a Type 42 font.

4.1 The FontInfo Dictionary

The optional **FontInfo** dictionary may be constructed from entries in the name and post tables in the TrueType font. It is not used by the PostScript interpreter, but some PostScript language programs may utilize entries such as **UnderlinePosition** and **UnderlineThickness**.

4.2 The sfnts Array

In VM, a TrueType font is represented as an array named **sfnts** which is composed of PostScript language string objects which, when concatenated, comprise the entire TrueType font. Multiple strings may be required due to the PostScript language implementation limit of 65535 bytes in a string.

When a TrueType font is divided into multiple strings, the strings must begin at TrueType table boundaries, or at individual glyph boundaries within the *glyf* table. The TrueType file format requires that tables begin at 4-byte

boundaries and that individual glyph descriptions begin at 2-byte boundaries. Therefore, each string will contain an even number of bytes of TrueType data.

For compatibility with Type 42 implementations in PostScript interpreter versions prior to 2013, each string must have one additional padding byte appended by adding “00” to the hex data in the file. That is, the length of each string must be odd. The last byte is not logically part of the TrueType font data and is ignored by the interpreter.

The **sfnts** array is expressed as a series of strings:

```
/sfnts [ <string1> <string2> ... <stringN> ] def
```

In the font file, the strings are made up of lines of hexadecimal characters. The characters in each line may be preceded, followed, and divided by an arbitrary (but consistent) number of white space or control characters (see the additional compatibility constraint in the bullet list below for fonts downloaded to a hard disk).

For Type 42 fonts to be downloaded to a printer’s disk (or other filesystem hardware), there are specific additional constraints on the text representation of that array in the file. Observing these constraints and beginning the file with the correct comment line (see section 2.1) enables the PostScript interpreter to have dynamic access to the font file on an as-needed basis, which has significant implications for saving VM. If a Type 42 font has the correct initial comment line but does not conform to the constraints listed below, the result will be an `invalidfont` error on some interpreters.

Although newer versions of the PostScript interpreter are likely to have fewer restrictions on the format of the **sfnts** array, the following constraints should be used, for backward compatibility purposes, to enable dynamic access to a disk-based font file:

- There may be whitespace and/or control-characters between the **/sfnts**, the “[“, and the “<“, and between any string’s “>” and the next string’s “<”.
- The string should be sub-divided into lines of a constant n characters in length and which may be divided by m characters of white space and/or control characters. The numbers n and m must be constant for the entire **sfnts** array, although it may vary from font to font. Line lengths should also satisfy the Document Structuring Convention constraint: $0 < n \leq 255$ (see Appendix G of *The PostScript Language Reference Manual, second edition*)
- The data encoding technique used in the **sfnts** array, for example, either ASCII-Hex or binary (see below) must be the same for all strings in a particular font, but may vary among fonts.

The strings in the **sfnts** array may be represented in binary in the same way as may be used for Type 1 charstrings (see section 2.4 in the *Adobe Type 1 Font Format* book). However, fonts using this representation cannot be installed on disk in PostScript interpreter versions prior to 2013. Also, they cannot be safely transmitted across non-binary channels, so fonts in this format should not be embedded in documents. Use of this format should be limited to disk font installer utilities that know something about the capabilities of the PostScript interpreter being accessed.

To represent a string in a binary representation, a PostScript language procedure must be defined with the following code:

```
/RD {string currentfile exch readstring pop} executeonly def
```

Each use of RD is followed by exactly one blank character followed by a sequence of binary bytes that are the string contents:

```
n RD ~binary~bytes~ {noaccess def} executeonly def
```

RD itself is preceded by an integer *n* which is the number of binary bytes following the RD (not including the single blank that follows the RD).

The following is an example of a two-element **sfnts** array encoded in this way:

```
/sfnts [  
  62135 RD ~62135~binary~bytes~  
  12093 RD ~12093~binary~bytes~  
] def
```

Each string contains an even number of bytes of TrueType data, followed by one byte of padding which the PostScript interpreter ignores.

4.3 Generating The CharStrings Dictionary

The **CharStrings** dictionary for a Type 42 font is a standard dictionary of *key/value* pairs, where the *key* is the glyph's name (derived from the TrueType font's cmap table), and the *value* is an index number into the TrueType font's loca (glyph offsets) table. The value in the key/value pair may also be a PostScript language procedure (executable array or packed array); see section 5.6.3 of the *The PostScript Language Reference Manual, second edition*.

4.4 Generating the Encoding Vector

TrueType fonts used in the Windows or Macintosh environments will generally use the encoding specific to that system, such as ANSI for Windows and the Apple encoding for the Macintosh. The platform-specific encoding can be determined by the platform ID number in a subtable of the cmap table. The

post table lists glyph names that differ from the platform's standard encoding. If there is no post table in a TrueType font in the Windows environment, the Windows ANSI encoding can be assumed.

To generate an encoding vector, the i^{th} name in the Type 42 font's **Encoding** array must be associated with the i^{th} entry in the TrueType cmap table. If a glyph is not in the cmap table, it should be given a glyph index of zero (0), which indicates a non-printing glyph.

4.5 Generating Unique Identifiers

The Type 42 font may contain a unique identifier which allows the glyph bitmaps to be cached across print jobs (see also section 5.8 of *The PostScript Language Reference Manual, second edition*). This entry is optional but highly desirable since many users may use the same fonts in every print job.

Bitmaps generated from TrueType fonts in Type 42 format use the same caching system as is used for Type 1 fonts. When a glyph bitmap is needed from a Type 42 font, the glyph cache is checked first. If the bitmap has not been cached, the bitmap is produced from the outline font program.

TrueType fonts do not contain any type of unique number which either corresponds to the PostScript language **UniqueID** entry or could be used for such. Using something like a checksum number as a **UniqueID** value devices would not be advisable since it does not assure uniqueness. Although this approach would work in many situations, there is an increased and unacceptable risk when, as at a service bureau, bitmaps are cached on a hard disk for a potentially long period of time. Hence, the performance gain resulting from caching does not offset the danger of a user getting incorrect bitmaps from the cache.

Since TrueType rasterizers only exist in Level 2 interpreters, the **XUID** operator offers a safer opportunity to cache bitmaps. The **XUID** (extended unique ID) is an array of integers which provides for distributed, hierarchical management of **UniqueID** numbers. The goal is to have a mechanism for generating an **XUID** array of values, on-the-fly, which are unique for every font, yet exactly repeatable since a TrueType font in a user's system may be converted multiple times to a Type 42 for printing.

A recommended method for generating a number for a given font which is both more likely to be unique than a simple checksum and exactly repeatable, is to use the MD5 algorithm from RSA Data Security, Incorporated. Their software can be copied and freely distributed if it is properly identified. The code for this algorithm is readily available from:

RSA Data Security, Inc.
100 Marine Parkway
Redwood City, CA 94065

The goal is to generate an **XUID** array of 5 elements, with the first having the value of 42 (decimal). This value has been registered in the Adobe **XUID** registry for use by software in creating Type 42 fonts. The MD5 algorithm can then be used to generate a 128-bit number, using the font file as input. This number can then be divided into four 32-bit integers to make the other four elements of the array. Some optimization of the algorithm code may be necessary to enhance performance.

4.6 Required TrueType Tables

In creating a Type 42 font from a TrueType font, only a limited subset of all potential tables in the original font are actually used by the rasterizer in the PostScript language device. Following is a list of the names of tables which are actually referenced by the TrueType rasterizer (see the TrueType specification for details):

head	prep
hhea	glyf
loca	hmtx
maxp	fpgm
cvt_	

Since a significant number of tables may be included in a TrueType font (including potentially large kerning tables), performance may be improved only by including the tables actually used by the TrueType rasterizer — in the downloadable Type 42 font.

4.7 Known Bugs

There is a known bug in the TrueType rasterizer included in versions of the PostScript interpreter previous to version 2013. The problem is that the translation components of the **FontMatrix**, as used as an argument to the **definefont** or **makefont** operators, are ignored. Translation of user space is not affected by this bug.

4.8 Example Type 42 font program

```
%!PS-TrueTypeFont-65536-65536-1
11 dict begin
  /FontName /Chicago def
  /Encoding 256 array
  0 1 255{1 index exch/.notdef put}for
  dup 0 /.null put
```


Index

Symbols

.notdef 5

B

bitmap cache 9

bugs 10

C

CDevProc 4

character coordinate system 5

comment lines 2

conversion issues 6

F

FontInfo dictionary 3, 5

FontMatrix 10

FontType 3, 6

G

glyph coordinate system 3, 5

H

hard disk 2, 3

I

invalidfont error message 7

M

makefont 10

MD5 algorithm 9

Metrics 4

Metrics2 5

P

PaintType 4

PPD file 2, 5

R

resourcestatus 3

S

sfnts 2, 7

string

binary representation 8

StrokeWidth 5

T

TrueType rasterizer 2, 5, 6, 10

TrueType specification 1

TrueType table

cmap 8

cvt_ 10

fpgm 10

glyf 5, 10

head 10

hhea 10

hmtx 10

loca 5, 8, 10

maxp 10

name 3, 6

post 3, 6, 9

prep 10

Type 42 Font Dictionary 3

U

UnderlinePosition 6

UnderlineThickness 5, 6

V

VM 2

VMusage comment 2