

Adobe Acrobat 7.0.5




Font Embedding Guidelines for Adobe Third-party Developers

July 27, 2005



Adobe Solutions Network — <http://partners.adobe.com>



Copyright 2005 Adobe Systems Incorporated. All rights reserved.

NOTICE: All information contained herein is the property of Adobe Systems Incorporated. No part of this publication (whether in hardcopy or electronic form) may be reproduced or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the Adobe Systems Incorporated.

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.

Except as otherwise stated, any reference to a "PostScript printing device," "PostScript display device," or similar item refers to a printing device, display device or item (respectively) that contains PostScript technology created or licensed by Adobe Systems Incorporated and not to devices or items that purport to be merely compatible with the PostScript language.

Adobe, the Adobe logo, Acrobat, the Acrobat logo, Distiller, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Apple and Macintosh are trademarks of Apple Computer, Inc., registered in the United States and other countries. Microsoft, Windows, and OpenType are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners.

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.



Preface

This document provides guidelines for Adobe® third-party application developers who are writing applications capable of embedding fonts in a file or accessing embedded fonts within a file. Most of the information is geared toward either plug-in developers working with the Acrobat® SDK, or Adobe PDF Library developers using the Adobe PDF Library SDK. Examples are based on Distiller® for Adobe Acrobat and the Acrobat SDK APIs.

There are multiple levels of embedding associated with font usage. For OpenType® and TrueType fonts, usage for embedding in documents is specified by **fsType** information in the font. The PostScript® Type 1 font format does not provide for this type of explicit information, so it is especially important to follow clear guidelines with regard to these fonts to ensure that you respect the license agreements established by font vendors for these fonts. Note, however, that it is possible to add **fsType** information to Type 1 fonts—see [“Fonts that Use FSType Instead of fsType” on page 19](#) for more information.

The goal of this document is to provide font usage guidelines—along with examples—that are followed by Adobe application developers. With this information, Adobe third-party developers can understand how Adobe applications handle font embedding issues. Adobe recommends that third-party developers follow these guidelines when writing software that works in conjunction with Adobe software products.

The **fsType** entry in a TrueType or OpenType font is the primary mechanism used by font developers to specify what type of embedding is allowed. The **fsType** specification was first developed for TrueType fonts and later used in the OpenType specification. In the absence of **fsType** information, such as with most PostScript Type 1 fonts, other mechanisms for determining embedding levels must be employed. This document discusses how Adobe uses **fsType** information when deciding whether and how to embed fonts. It also discusses how Adobe handles cases where **fsType** information is not available (see the tables in [Chapter 2, “Specific Guidelines”](#), for these cases).

To use this document, you need to be familiar with basic font formats and the PostScript printing process. Most importantly, you need to understand which formats are used to represent fonts when used on the host, when sent to a printer, and when embedded in a PDF file.

NOTE: The policies presented in this document do not guarantee that font usage will be in legal compliance with font vendor license agreements. For example, while the embedding information within a font’s **fsType** entry indicates that the font may be embedded to a certain level, a license with the font vendor may be separately required to use the font in this way. Third parties should seek the advice of their own legal council in all matters relating to font usage and embedding, regardless of what the policies presented in this document may imply.

Contents

This document contains the following chapters:

- This Preface provides an overview of the document, a list of key terms, and related publications.
- [Chapter 1, “General Guidelines”](#) provides guidelines that relate to the embedding of any font into a file.
- [Chapter 2, “Specific Guidelines”](#) provides guidelines that relate to the embedding of a specific font into a file. It primarily covers embedding a font for Preview & Print as well as for forms fill-in or free-text annotations. It also provides guidelines for when a font can be used for full editing capabilities.
- [Chapter 3, “Using fsType/FSType”](#) describes how to interpret the embedding information in the **fsType** and the **FSType** entries when included in a font.
- [Chapter 4, “Embedded Font Operations Using the Acrobat SDK”](#) provides simple examples of how the Acrobat SDK can be used to write plug-in code that implements the core policies presented in this document.

Key Terms

- **fsType/FSType** refers to the **fsType/FSType** entry in a TrueType, OpenType, CFF, or CID-keyed font. See [Chapter 3, “Using fsType/FSType”](#) for details.
- **WasEmbedded** refers to a new key written out in a PostScript stream to indicate that a font in the PostScript stream/file was already embedded and thus can be re-embedded if necessary. This key is added to the **FontInfo** dictionary of an embedded font by the application or printer driver. This key is found only in the **FontInfo** dictionary of a font embedded in a PostScript print stream. It is never found in host fonts. See [Chapter 1, “General Guidelines”](#).
- **OrigFontType** refers to a new key written out into the PostScript stream that identifies the original font type in the case where a font is converted from one font type to another. The intent is to enable developers to follow the guidelines that apply to the original font regardless of the transformations it has undergone. This key is found only in the **FontInfo** dictionary of a font embedded in a PostScript print stream. It is never found in host fonts.
- **XUID** refers to the extended unique identifier for each font. An **XUID** is an optional font element in a name-keyed Type 1 or PostScript OpenType font, but is required in a CID-keyed Type 1 font. The font vendor identifier portion of each **XUID** number is provided by Adobe. Third-party font developers submit a request to Adobe for an **XUID** font vendor identifier. Adobe simply provides the identifier; it does not associate the identifier with a specific font. The **XUID** approved list (see [Chapter 2, “Specific Guidelines”](#)) refers to a list compiled by Adobe (for internal use) of fonts that vendors

allow to be embedded. Fonts on this list were, in general, published before the **fsType** specification was developed. See *PostScript Language Reference, third edition*, for details.

- PDF is the Portable Document Format. Embedding issues are of particular importance to PDF since the embedding of fonts allows the look and feel of the document to be fully portable across platforms with no font substitution.
- EPS is Encapsulated PostScript. EPS is a standard format for importing and exporting PostScript language graphics among applications in a variety of heterogeneous environments. An EPS is basically a “single-page graphic,” as opposed to a multi-page PostScript document.
- SVG is an acronym for Scalable Vector Graphics. SVG is a language for describing two-dimensional vector and mixed vector/raster graphics in XML. SVG is an industry standard defined by the World Wide Web Consortium (W3C). See <http://www.w3.org>.
- OpenType is a cross-platform font format developed jointly by Adobe and Microsoft[®]. It is an extension to the TrueType font format, adding support for PostScript font data in CFF format. The two main benefits of the OpenType format are its cross-platform compatibility (the same font file works on Macintosh[®] and Windows[®] computers), and its ability to support expanded character sets and layout features. See <http://partners.adobe.com/asn/tech/type/opentype/index.jsp>.
- TrueType is a digital font technology designed by Apple[®] Computer. It is now used by both Apple and Microsoft in their operating systems. See <http://www.microsoft.com/typography/users.htm>.
- Type 1 is a font format originally designed for single-byte fonts for use on host systems and with PostScript printers. The Type 1 format is specified in *Adobe Type 1 Font Format*.
- Type 42 fonts consist of a PostScript language “wrapper” around a TrueType font. A Type 42 font is usually generated by a printer driver to download TrueType fonts to a PostScript printer that includes a TrueType rasterizer. See *The Type 42 Font Format Specification (Technical Note #5012)* for details.
- CID is a Character IDentifier. Fonts based on CID (CID-keyed fonts) provide a convenient way for defining multiple-byte character encodings. See *PostScript Language Reference, third edition*, for details.
- CFF is the Compact Font Format. CFF is suitable for compactly representing one or more Type 1 or CID-keyed fonts. Raw (unwrapped) CFF fonts are not generally used on host systems. The main uses of raw CFF fonts are:
 - by Distiller for embedding fonts in PDF files,
 - as a format for internal use in PostScript output devices.

Additionally, a table containing a CFF font is the means for supporting Type 1 structures within the OpenType format. See *The Compact Font Format Specification (Technical Note #5176)*.
- CJK is an acronym used to identify the Chinese, Japanese, and Korean languages (the main Asian multi-byte languages).
- CEF (Compact Embedded Font) is a font format used by SVG.

- OCFs (Original Composite Fonts) are an older type of CJK PostScript font, pre-dating CID-keyed Type 1 fonts.
- CoolType is used by Adobe application developers to handle fonts in Adobe applications. It is a font-server component used by most Adobe applications. It supports font selection, encodings, glyph selection, metrics, rasterization, and font conversion. “CoolType” is also the official name of Adobe’s device-dependent rendering technology (e.g., for LCD screens) as used by Acrobat. For the purpose of this document, the former definition applies.

Related Publications

The documents below are available on the [partners.adobe.com Web site](https://partners.adobe.com).

PostScript Language Reference, third edition is the definitive programmer’s reference for the syntax and semantics of the PostScript language, the imaging model, and the effects of the graphics operators.

Enabling PDF Font Embedding for CID-Keyed Fonts (Technical Note #5641) describes how developers of CID fonts can specify whether their fonts can be embedded via the **FSType** entry in the **FontInfo** dictionary.

OpenType Specification, version 1.4 contains the full specification for the **fsType** entry in the **OS/2** table.

The Compact Font Format Specification (Technical Note #5176)

Encapsulated PostScript (EPS) File Format Specification Version 3.0 (Technical Note #5002)

The Type 42 Font Format Specification (Technical Note #5012)

PDF Reference, Fourth Edition, version 1.5

Acrobat Core API Reference

Adobe Type 1 Font Format

1

General Guidelines

Introduction

This chapter provides guidelines that relate to the embedding of any font into a file. These guidelines are followed whenever Adobe software considers whether to embed fonts in files, whether through initial creation or transformation of existing files. It is possible that Adobe will not co-market third-party software that doesn't meet these guidelines.

Fonts, of course, are used to create and view documents, or print and modify them. Fonts can be subsetted when embedded to use only those glyphs required for display and printing, reducing overall file size. In general, the TrueType and OpenType specifications define the following four types of embedding associated with a font:

1. *No embedding allowed.* The font may not be embedded into an electronic document for any purpose.
2. *Embedding for Preview & Print only.* A font that is embeddable for Preview & Print allows the font, either the full character set or only a subset of characters used, to be embedded in an electronic document solely for the purpose of viewing that document on-screen and/or printing that document. While a font that is embeddable for Preview & Print may be embedded in an electronic document, the embedded font may not be used to further edit the document in which it is contained, or to edit or create other documents, or to fill in forms fields.
3. *Editable embedding.* Fonts that are embeddable for Editable embedding can be embedded in electronic documents for use by the recipient of the electronic document to view, print and further edit or modify the text and structure of the document in which it is embedded. These changes or edits can then be saved in the original document. Editable embedding is the type of embedding done for forms fill-in and free-text annotations. This type of embedding is to be distinguished from use of embedded fonts for editing, which provides full text editing capabilities with an embedded font (an operation which Adobe does not currently support). Adobe products currently support the use of embedded Editable embedding fonts for forms fill-in and free-text annotations but does not support their use for general purpose full text editing.
4. *Installable embedding.* The font may be provided with an electronic document and installed on the recipient's computer for use in creating/authoring new documents. Note, however, that Adobe software will not, as a matter of policy, install an embedded font onto a user's system, regardless of the information specified in the font by the vendor.

When to Check Embedding Information

When Adobe software embeds a whole file within another file (such as embedding an EPS within a multi-page PostScript document) in such a fashion that the file is included without modification, Adobe software does not look for fonts which may exist in the embedded file to determine embedding-level information.

PostScript streams generated with the intention that they will be sent to a printer may include fonts not otherwise embeddable in other files (i.e., including those fonts whose **fsType** information is set to “No Embedding”), since this is necessary for printing. Creation of all other files (including EPS) must follow all of the guidelines. EPS files should be handled like PDF and SVG files (*not* like PostScript streams).

NOTE: OCFs are never embedded, even for PostScript streams sent to a printer. To print using OCFs, the fonts must be resident on the printer.

Adobe software does not allow a font to be embedded whose **fsType** embedding information doesn't allow embedding (except for the case of PostScript print streams). When a font is embedded for use in forms fill-in or free-text annotations, Adobe software ensures that only fonts having **fsType** information set to Editable embedding are utilized. Despite the definition of Editable embedding in the True Type and OpenType specifications, for actual editing, or for adding comments in notes to the document, Adobe software only uses fonts currently installed on the user's system.

In Adobe Acrobat, for example, fonts installed on a user's system are required if the font is to be used with the TouchUp Text Tool and the Note Tool. If the user attempts to perform editing operations in a PDF document with the TouchUp Text Tool or the Note Tool by using an embedded font that is not installed on the user's system, the operation fails (the user is unable to enter text from his keyboard at the cursor insertion point). The operation is successful only if the font is already installed on the user's system. For an example of how to check for the existence of fonts on a user's system, see [Chapter 4, “Embedded Font Operations Using the Acrobat SDK”](#).

NOTE: Adobe does not object to products that use embedded fonts for full editing, assuming that the **fsType** information is set to Editable embedding. Adobe simply chooses not to do so in its products at this time.

Information that Must be Preserved When Embedding

Always embed (retain) copyright and trademark information, if it exists, when embedding a font. In Type 1 and derivative formats, this is most commonly a **/Notice** key in the **FontInfo** dictionary. Sometimes a **/Copyright** key exists in addition to the **/Notice** key, in which case both should be retained. For TrueType and OpenType fonts, the copyright and trademark information is stored in the **name** table (copyright is **nameID 0** and trademark is **nameID 7**).

Whenever fonts are embedded (in any file including PostScript streams), the following logic should be followed to preserve embedding information in the embedded fonts if the

information exists or can be known at the time the font is embedded. In general, **fsType/FSType** information should not be estimated and added to a font when it did not previously and explicitly exist.

```
If fsType/FSType exists
  embed fsType/FSType
Else
  embed OrigFontType
  embed WasEmbedded (if font came from an EPS, PDF, or SVG file)
  embed XUID
```

The terms used are explained below. Also see [“Key Terms” on page 4](#).

- **fsType/FSType** — The **fsType/FSType** entry. If there is an existing fsType/FSType entry, there must also be one in the embedded font. However, if the format of the font is changed during the embedding process, it may become necessary to store fsType in a different way. See [Chapter 3, “Using fsType/FSType”](#) for details.
- **WasEmbedded** — A new key written out in a PostScript stream to indicate that a font in the PostScript stream/file was already embedded and thus can be re-embedded if necessary. This key is found in the **FontInfo** dictionary in the embedded font. The value for **WasEmbedded** is a boolean (**true** or **false**). If **WasEmbedded** is **true**, then the font is assumed to be embeddable. This key is found only in a font embedded in a PostScript print stream. It is never found in host fonts.
- **OrigFontType** — A new key written out in a PostScript stream that identifies the original font type in the case where a font is converted from any other font type into an embedded PostScript font, and no **fsType/FSType** information is present in the original font. The intent is to enable developers to follow the guidelines that apply to the original font regardless of the transformations it has undergone. This key is found in the **FontInfo** dictionary. The value for **OrigFontType** is a name (**/Type1**, **/TrueType**, **/OCF**, or **/CID**). If **OrigFontType** is **/Type1** or **/TrueType**, then the font is assumed to be embeddable. This key is found only in a font embedded in a PostScript print stream. It is never found in host fonts.
- **XUID** — The extended unique identifier for each font. **XUIDs** are an optional font element. The font vendor identifier portion of each **XUID** number is provided by Adobe. **XUIDs** are optional for Type 1 fonts, but it is required that font developers put them in CID-keyed and PostScript OpenType fonts.

2

Specific Guidelines

Introduction

This chapter provides guidelines that relate to the embedding of a specific font into a file, then presents an example of how an Adobe application (Distiller for Adobe Acrobat) implements the guidelines. The guidelines cover embedding fonts when files are created for Preview & Print, and embedding them for forms fill-in and free-text annotations. Adobe doesn't embed fonts for the purpose of editing PDF documents. If editing is required, Adobe software uses fonts already installed on the system.

NOTE: References to "approved lists" of fonts are lists maintained by Adobe for use in Adobe software. For example, Adobe maintains a list of non-Adobe Type 1 fonts that Adobe has permission to use for Editable embedding even though the fonts themselves carry no embedding level information (have no **fsType/FSType** information). Third parties should compile their own lists for use in their software.

Guidelines for Embedding Fonts During Electronic File Creation

The following guidelines should be followed when an application is *creating* an electronic file for Preview & Print.

TABLE 2.1 *Embedding Guidelines for Electronic File Creation*

Font	Minimum Required	Highly Recommended
PostScript	<ul style="list-style-type: none">● If embedding in PostScript stream, OK to embed.	<ul style="list-style-type: none">● Default should be to subset font when author is using less than 100% of the characters in the font. (Author can change this preference.)
Type 1 (Western)	<ul style="list-style-type: none">● Else if fsType/FSType exists, respect fsType/FSType settings.	
CEF	<ul style="list-style-type: none">● Else if the font WasEmbedded, OK to embed.	
CFF	<ul style="list-style-type: none">● Else if OrigFontType exists, follow the rules in this document for the font as identified by OrigFontType.● Else, allow the font to be embedded.	

TABLE 2.1 Embedding Guidelines for Electronic File Creation

Font	Minimum Required	Highly Recommended
OpenType (Western & CJK)	<ul style="list-style-type: none"> ● If embedding in PostScript stream, OK to embed. ● Else if fsType/FSType exists, respect fsType/FSType settings. 	<ul style="list-style-type: none"> ● Default should be to subset font when author is using less than 100% of the characters in the font. (Author can change this preference.)
TrueType (Western & CJK)	<ul style="list-style-type: none"> ● Else if the font WasEmbedded, OK to embed. ● Else if OrigFontType exists, follow the rules in this document for the font as identified by OrigFontType. ● Else, allow the font to be embedded. 	
CID (CJK) CID-CFF	<ul style="list-style-type: none"> ● If embedding in PostScript stream, OK to embed. ● Else if fsType/FSType exists, respect fsType/FSType settings. ● Else if the font WasEmbedded, OK to embed. ● Else if OrigFontType exists, follow the rules in this document for the font as identified by OrigFontType. ● Else if XUID exists and is on approved list, OK to embed. ● Else if XUID exists and is <i>not</i> on approved list, do <i>not</i> embed. ● Else, do <i>not</i> embed. 	<ul style="list-style-type: none"> ● Default should be to subset font when author is using less than 100% of the characters in the font. (Author can change this preference.)
OCF (CJK)	<ul style="list-style-type: none"> ● Never embedded. 	N/A

Guidelines for Embedding in Forms and Free-text Annotations

The following guidelines should be followed for the type of embedding done for forms fill-in and free-text annotations. This type of embedding is to be distinguished from use of embedded fonts for editing, which provides full text editing capabilities with an embedded font (capabilities which Adobe does not currently support).

NOTE: Only fonts whose **fsType** or related information indicates that they are Editable can be used for forms text and free-text annotations.

TABLE 2.2 *Embedding Guidelines for Forms Fill-In/Free-text Annotations*

Font	Minimum Required	Highly Recommended
PostScript Type 1 (Western) CEF CFF	<ul style="list-style-type: none"> ● If fsType/FSType exists, respect fsType/FSType settings. ● Else if OrigFontType exists, follow the rules in this document for the font as identified by OrigFontType. ● Else if Type 1 font that meets the following criteria, can be treated as Editable: <ol style="list-style-type: none"> 1. If the string: "...trademark of Adobe" is identifiable in the font, then treat as Editable. 2. Otherwise if "Adobe" is in the notice string <i>and</i> the PostScript font name is on approved list, treat the font as Editable. ● Else (no fsType/FSType), assume embedded font is Preview & Print only. 	<ul style="list-style-type: none"> ● If the user types a character unavailable in the current font, an application should follow one of two options: 1) Use a "fallback font" to display the missing characters (selection of fallback font may be determined by the form creator, the OS, the application or the user); 2) Display blanks or notdef symbols. ● If a fallback font is used and embedded in a form field, it is subject to the same restrictions as any other font being embedded for use in a form field.
OpenType (Western & CJK) TrueType (Western & CJK)	<ul style="list-style-type: none"> ● If fsType/FSType exists, respect fsType/FSType settings. ● Else if OrigFontType exists, follow the rules elsewhere in this document for the font as identified by OrigFontType. ● Else (no fsType/FSType), assume embedded font is Preview & Print only. 	<ul style="list-style-type: none"> ● If the user types a character unavailable in the current font, an application should follow one of two options: 1) Use a "fallback font" to display the missing characters (selection of fallback font may be determined by the form creator, the OS, the application or the user); 2) Display blanks or notdef symbols. ● If a fallback font is used and embedded in a form field, it is subject to the same restrictions as any other font being embedded for use in a form field.

TABLE 2.2 Embedding Guidelines for Forms Fill-In/Free-text Annotations

Font	Minimum Required	Highly Recommended
CID (CJK) CID-CFF	<ul style="list-style-type: none"> ● If fsType/FSType exists, respect fsType/FSType settings. ● Else if OrigFontType exists, follow the rules in this document for the font as identified by OrigFontType. ● Else (no fsType/FSType), assume embedded font is Preview & Print only. 	<ul style="list-style-type: none"> ● If the user types a character unavailable in the current font, an application should follow one of two options: 1) Use a "fallback font" to display the missing characters (selection of fallback font may be determined by the form creator, the OS, the application or the user); 2) Display blanks or notdef symbols. ● If a fallback font is used and embedded in a form field, it is subject to the same restrictions as any other font being embedded for use in a form field.
OCF (CJK)	<ul style="list-style-type: none"> ● No support for forms. 	N/A

Guidelines for Use of Embedded Fonts for Editing

If the font is resident on user's system, then allow editing using the system resident font. As of this writing, no Adobe application allows use of embedded fonts for editing. If the author tries to use the application for full editing capabilities and the font is not resident on the user's system, no text will be input at the cursor insertion point when the user types, indicating that he will have to select another font.

Even if a font is already fully embedded in a file and its **fsType/FSType** indicates Editable embedding, current Adobe software still checks to ensure that the font is installed on the user's system before allowing full editing capabilities (such as for the Acrobat TouchUp Text Tool or Acrobat Note Tool). For an example of how to check whether a font is installed on a user's system or is embedded only, see [Chapter 4, "Embedded Font Operations Using the Acrobat SDK"](#).

Distiller Example

An application of the above guidelines is demonstrated in the following table. Using numbers to indicate relative priority, the table shows the search order used to determine whether and how fonts will be embedded in a PDF file. This is the algorithm currently used in Distiller. Note that for OpenType and TrueType fonts, Distiller does not check any entry

other than the **OS/2 fsType** entry, which is why there are no numbers greater than 1 for these fonts in the columns one and four.

TABLE 2.3 *Embedding Guidelines for Conversion From PS to PDF in Distiller*

	OpenType with PostScript outlines	Type1 and CIDFontType0	Type42 and CIDFontType2 ^a	TrueType and OpenType with TrueType outlines
FSType in FontInfo		1	1	
FSType in Font dictionary		2	2	
OS/2 fsType	1		3	1
WasEmbedded		3	4	
OrigFontType		4	5	
What gets embedded in PDF ^b	CFF: PS code /FSType val def	CFF: PS code /FSType val def if FSType was found	TrueType font: OS/2 table if FSType or OS/2 was found	TrueType font: OS/2 table if fsType or OS/2 was found

- a. Distiller looks at **FSType** before **fsType** in **OS/2** table for Type42 fonts for historical reasons: There were some faulty **OS/2** tables in Mac OS9 system TrueType fonts, so this was a special workaround.
- b. When Distiller embeds Type 1, PostScript-flavored OpenType, and CIDFonts in Compact Font Format (CFF), the embedding information, if present, is preserved as embedded PostScript language code: **/FSType value def** in CFFs (see *The Compact Font Format Specification—Technical Note #5176, Appendix F*) where **value** is the decimal representation of the embedding flags (for example, **/FSType 8 def**). Distiller embeds TrueType and TrueType-flavored OpenType fonts in TrueType format and the embedding information, if present, is preserved in the **OS/2** table.

3

Using fsType/FSType

Introduction

This chapter describes the **fsType** entry and makes the distinction between the **fsType** and **FSType** designations. It describes where the **fsType/FSType** entry can be found in different fonts. Finally, it provides Adobe's guidelines for how the **fsType/FSType** entry is to be interpreted and handled in Adobe applications.

The **fsType/FSType** entry provides information regarding whether you can edit using the font, distribute the font freely, and so on. Adobe doesn't allow the user to perform operations using Adobe software that are not allowed by the font embedding information. In addition, Adobe software end users are also directed in Adobe EULAs to separately confer with font vendors regarding any additional licensing considerations that might apply to font usage. For some fonts, Adobe has agreements with vendors that allow customers to embed and use certain fonts. When someone purchases font software from Adobe, or an Adobe application that includes font software, he is allowed to use the fonts and embed those fonts into documents for the purposes specified in the license. Specific information about Adobe's agreements with font vendors is provided on the Adobe Web site. It is recommended that third parties provide the same information to their customers. See, for example:

<http://www.adobe.com/type/browser/legal/embeddingeula.html>

Description of the fsType Entry

In TrueType and OpenType fonts, embedding information is included in the **fsType** entry of the **OS/2** table. This field contains a number of bits that specify the levels of embedding that are available and is described in the table below. The formal definition can be found at:

<http://partners.adobe.com/asn/tech/type/opentype/index.jsp>

Any file other than a print stream (including EPS, PDF, and SVG) should follow the guidelines provided in this embedding information.

TABLE 3.1 *fsType Bits of the OS/2 table*

Bit	BitMask	Description
	0x0000	Installable Embedding: No fsType bit is set. Thus fsType is zero. Fonts with this setting indicate that they may be embedded and permanently installed on the remote system by an application. The user of the remote system acquires the identical rights, obligations and licenses for that font as the original purchaser of the font, and is subject to the same end-user license agreement, copyright, design patent, and/or trademark as was the original purchaser.
0	0x0001	Reserved, must be zero.
1	0x0002	Restricted License embedding: Fonts that have <i>only</i> this bit set <i>must not be modified, embedded or exchanged in any manner</i> without first obtaining permission of the font software copyright owner. <i>Caution:</i> For Restricted License embedding to take effect, it must be the only level of embedding selected.
2	0x0004	Preview & Print embedding: When this bit is set, the font may be embedded, and <i>temporarily</i> loaded on the remote system. Documents containing Preview & Print fonts must be opened “read-only”; no edits can be applied to the document.
3	0x0008	Editable embedding: When this bit is set, the font may be embedded but must only be installed <i>temporarily</i> on other systems. In contrast to Preview & Print fonts, documents containing Editable fonts <i>may</i> be opened for reading, editing is permitted, and changes may be saved.
4-7		Reserved, must be zero.
8	0x0100	No subsetting: When this bit is set, the font may not be subsetted prior to embedding. Other embedding restrictions specified in bits 0 - 3 and 9 also apply.
9	0x0200	Bitmap embedding only: When this bit is set, only bitmaps contained in the font may be embedded. No outline data may be embedded. If there are no bitmaps available in the font, then the font is considered unembeddable and the embedding services will fail. Other embedding restrictions specified in bits 0 - 3 and 8 also apply.
10 - 15		Reserved, must be zero.

NOTE: If multiple embedding bits are set, the *least* restrictive combination takes precedence. For example, if bits 1 and 3 are set, bit 3 takes precedence over bit 1 and the font may be embedded as Editable. For compatibility purposes, most vendors that allow Editable embedding also set the Preview & Print bit (0x000C). This will permit an application that supports only Preview & Print embedding to detect that font embedding is allowed.

Fonts that Use FSType Instead of fsType

When TrueType and OpenType fonts are sent to Distiller, they are usually converted into one of the following PostScript representations: Type 1, Type 42, or CIDFont. This has led to the need to represent **fsType** values in formats that did not originally support them. The **FSType** entry (note different capitalization) uses the same values as the **fsType** entry, except that it is found in font types other than TrueType and OpenType fonts. **FSType** is now recognized in Type 1, Type 42, and CIDFonts.

To store an **fsType** value in a CIDFont, store it in the **FontInfo** dictionary under the key **/FSType**. See *Enabling PDF Font Embedding for CID-Keyed Fonts (Technical Note #5641)*. The Type 1 specification provides no mechanism for representing **FSType**, but it can be added anyway. This is done by adding a key/value pair to either the font's **FontInfo** dictionary or to the top-level font dictionary. The key/value pair is of the form **/FSType value def**.

The Type 42 specification provides a way to include **FSType**, by inclusion of the font's **OS/2** table, but the **OS/2** table is not required in Type 42 fonts.

It is thus possible to look for an **FSType** value in both Type 1 and Type 42 fonts. As an example of implementation, for both Type 1 and Type 42 fonts Distiller 6.0 looks for the **/FSType** key in the font's **FontInfo** dictionary and in the top-level font dictionary. If **FSType** is present in both dictionaries, the one in **FontInfo** is used by Distiller. For Type 42 fonts, if **FSType** is not present in either dictionary, then the **fsType** in an included **OS/2** table is used (see footnote to [Table 2.3](#)).

How Adobe Applications Interpret the fsType/FSType Entry

The following algorithm defines how Adobe applications interpret the **fsType** specification outlined in “[Description of the fsType Entry](#)”. Presented are the pseudocode rules that define the phrase “respect **fsType/FSType** settings” that is used in the tables in [Chapter 2](#), “[Specific Guidelines](#)”.

NOTE: In the following algorithm, **x** represents a bit that may be ignored

1. If **fsType/FSType** equals 0000 0000 0000 0010 (**fsType/FSType** = 2 in decimal) then no embedding allowed,

2. Else if **fsType/FSType** bit 9 is 1 [**xxxx xx1x xxxx xxxx**] then no embedding allowed (this setting explicitly allows for bitmap embedding, but CoolType does not currently support bitmap embedding),
3. Else if **fsType/FSType** bit 3 is 1 [**xxxx xx0x xxxx 1xxx**] then Editable embedding is allowed,
4. Else if **fsType/FSType** bit 2 is 1 [**xxxx xx0x xxxx 01xx**] then Preview & Print embedding is allowed,
5. Else Editable embedding is allowed. The font includes no bit setting or a combination of **fsType** bits not defined in the specification; for these cases, the specification indicates that the font should be treated as installable. Adobe products don't provide for installable embedding; thus the next least restrictive bit interpretation is used—Editable embedding.

4

Embedded Font Operations Using the Acrobat SDK

Introduction

This chapter discusses embedded font-related operations using the Acrobat SDK. Examples of common operations are provided. Distiller and the PDF Library add font embedding information to fonts that are embedded in PDF files. With the inclusion of this information, your code can determine how an embedded font can be used. The operations discussed in this chapter also apply, for the most part, to code used with the PDF Library SDK.

Embedded Font-related Operations

Adobe Acrobat plug-in developers can remove and embed fonts within an existing PDF document. They can also use fonts that are already embedded in a PDF document for Preview & Printing, as well as editing. However, allowing editing using embedded fonts is not recommended by Adobe, and in some cases it is impractical. Specifically, CJK fonts potentially include thousands of glyphs, so it is necessary for applications to subset these fonts when embedding them in a PDF file. This precludes embedded CJK fonts from being used for editing by a plug-in.

PDF Library users can perform all of the above operations using an existing PDF document, as well as create a PDF document from scratch that includes embedded fonts. Creating a document from scratch cannot be performed by a plug-in, but this can be done by using PDF library calls from within a compiled application that includes the PDF Library.

Many embedded font-related operations using the Acrobat SDK require use of the **PDEFontGetSysFont** method. This method first checks whether a font (provided as the sole argument to the method) is installed on the system. If it is not, then the method checks whether the font is embedded in the current PDF file. When using **PDEFontGetSysFont** to determine what font operations should be allowed, you need to check whether the returned font is a system font. If it is, then full editing can be allowed. If it is not, then the font is embedded only and the restrictions discussed in this document should be observed.

When checking embedded information, use an **if-else if** structure that checks first for the least restrictive use of the font, then checks progressively for greater and greater restrictions.

Plug-ins should subset fonts when they are used for Preview & Print operations, or when the font is too large to add to a PDF file. Embed the entire font when Editable embedding is used, unless the font is too large.

Example 1

This example demonstrates how to check the embedding information of a system font and then embed as appropriate. The key APIs demonstrated are **PDFFindSysFont**, **PDSysFontGetAttrs**, and **PDEFontCreateFromSysFont**.

NOTE: Do not use **PDEFontGetAttrs** to check the protection field bit in the **PDEFontAttrs** structure. The protection field is only valid when using **PDSysFontGetAttrs**.

```
// Initialize the font descriptor then create the font reference.
PDEFont pdeFont = NULL;
PDEFontAttrs pdeFontAttrs;
PDESysFont sysFont;
ASUns32 fontCreateFlags;
memset(&pdeFontAttrs, 0, sizeof(pdeFontAttrs));

// Find the matching system font.
pdeFontAttrs.name = ASAtomFromString("Verdana");
sysFont = PDFindSysFont(&pdeFontAttrs, sizeof(pdeFontAttrs), 0);

// Get the font attributes.
PDSysFontGetAttrs(sysFont, &pdeFontAttrs, sizeof(pdeFontAttrs));

// Create the PDE font from the system font.
// Check the font policy for Preview & Print, or editing.
// Based on the font permission policy, decide whether to embed/subset
// the font.

if ((pdeFontAttrs.protection & kPDEFontNoEditableEmbedding) == 0) {
// Editing OK. Embed the entire font.
fontCreateFlags = kPDEFontCreateEmbedded;
}
else if ((pdeFontAttrs.protection & kPDEFontNoEmbedding) == 0) {
// Preview & Print embedding OK, editing is NOT OK.
// Subset the embedded font.
fontCreateFlags = kPDEFontCreateEmbedded|kPDEFontWillSubset;
}
else {
// Embedding not allowed.
fontCreateFlags = kPDEFontDoNotEmbed;
}

// Create the PDE font. Embed if embedding information allows.
pdeFont = PDEFontCreateFromSysFont(sysFont, fontCreateFlags);
```

Example 2

The following example shows how to find a font that may be either installed on the system or embedded in a document, then investigate the embedding information in the font. The key APIs demonstrated are **PDEFontGetSysFont** and **PDSysFontGetAttr**. To determine if the font is installed on the system (as opposed to embedded in the document) use the **PDFindSysFont** method as shown in [Example 1](#).

NOTE: Do not use **PDEFontGetAttrs** to check the protection field bit in the **PDEFontAttrs** structure. The protection field is only valid when using **PDSysFontGetAttrs**.

```
ASBool EditingEmbeddedFontOK(PDFont fontP)
{
    CosObj fontObj = PDFontGetCosObj(fontP);
    PDEFont pdeFont = PDEFontCreateFromCosObj(&fontObj);
    PDESysFont pdSysFont = PDEFontGetSysFont(pdeFont);
    PDEFontAttrs attrs;
    if (pdSysFont)
    {
        PDSysFontGetAttrs(pdSysFont, &attrs, sizeof(attrs));
        if ((attrs.protection & kPDEFontNoEditableEmbedding) == 0)
            // Editing OK.
        else if ((attrs.protection & kPDEFontNoEmbedding) == 0)
            // Preview & Print embedding OK.
            //Editing is NOT OK.
        else
            /* Embedding not allowed... */
    }
}
```

