Font Embedding Guidelines for Adobe® Third-party Developers

Adobe® Acrobat® DC SDK

May 2015

Adobe® Acrobat® DC SDK Font Embedding Guidelines for Adobe Third-party Developers for Microsoft® Windows® and Mac OS®

Edition 1.0, May 2015

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contents

List of Examples

1 Preface

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 following topics are covered in this guide:

- General Guidelines
- Specific Guidelines
- Using fsType/FSType
- Embedded Font Operations Using the Acrobat SDK

The glossary on page 4 clarifies the terminology used in this guide.

Who should read this guide?

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

To use this document, you must be familiar with basic font formats and the PostScript® printing process. Most importantly, you must 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, although 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 counsel in all matters relating to font usage and embedding, regardless of what the policies presented in this document may imply.

Related documentation

The following documents are available on the Adobe Developer Connection Web site at www.adobe.com/devnet/

For information about	See
The definitive programmer's reference for the syntax and semantics of the PostScript language, the imaging model, and the effects of the graphics operators.	*PostScript Language Reference, third edition*
How developers of CID fonts can specify whether their fonts can be embedded via the `FSType` entry in the `FontInfo` dictionary.	*Enabling PDF Font Embedding for CID-Keyed Fonts (Technical Note #5641)*
The full specification for the `fsType` entry in the OS/2 table.	*OpenType Specification, version 1.4*
The specification of a font format that is suitable for compactly representing one or more Type 1 or CID-keyed fonts.	*The Compact Font Format Specification (Technical Note #5176)*
The specification for EPS, a PostScript language program that describes the appearance of a single page.	Encapsulated PostScript File Format Specification, Version 3.0
The PostScript Type 42 font format, which can be used to download TrueType fonts to PostScript printers (or PostScript compatible printers) that contain a TrueType rasterizer.	*The Type 42 Font Format Specification (Technical Note #5012)*
A detailed description of the PDF file format.	*PDF Reference*
Detailed descriptions for the APIs that can be used to develop plug-ins for Acrobat and Acrobat Readert DC®, as well as PDF Library applications.	*Acrobat and PDF Library API Reference*
The organization of the Adobe Type 1 font format and how to create a Type 1 font program.	*Adobe Type 1 Font Format*

1 General Guidelines

This chapter provides guidelines that relate to embedding any font into a file. These guidelines are followed whenever Adobe software considers whether to embed fonts in PDF or EPS files, whether through initial creation or transformation of existing files. Adobe may not co-market third-party software that does not meet these guidelines.

Multiple levels of embedding are 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 their vendors. Note, however, that it is possible to add `fsType` information to Type 1 fonts. (See "Fonts that use FSType instead of fsType" on page 5 for more information.)

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 types of embedding associated with a font:

No embedding allowed: The font may not be embedded into an electronic document for any purpose.

Embedding for preview and print only: Fonts can be embedded for preview and print, either the full character set or only a subset of characters can be embedded in an electronic document solely for the purpose of viewing that document on-screen or printing. Although a font that can be embedded for preview and 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, to edit or create other documents, or to fill forms fields.

Editable embedding: Fonts that are available for editable embedding can be embedded in electronic documents. The recipient of the electronic document can then use the fonts to view, print, and modify the text and structure of the document in which it is embedded. These changes can then be saved in the original document. Editable embedding is the type of embedding done for filling forms and editing free-text annotations. This type of embedding is different from using embedded fonts for editing, which provides full text editing capabilities using an embedded font. Adobe products currently support editable embedding fonts for filling forms and editing free-text annotations but does not support their use for general purpose full text editing.

Installable embedding: The font may be provided with an electronic document and installed on the recipient's computer for use in creating new documents. Note, however, that currently no Adobe software installs 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 that may exist in the embedded file to determine embedding-level information.

PostScript streams to be sent to a printer may include fonts that cannot otherwise be embedded in other files, including those fonts whose `fsType` information is set to *no embedding allowed*, because this is necessary for printing. Creation of all other files (including EPS) must follow all the guidelines. EPS files should be handled like PDF and SVG files (*not* like PostScript streams).

Note: Original Composite Fonts (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 in PDF and EPS files whose `fsType` embedding information does not allow embedding except for PostScript print streams. When a font is embedded for use in filling forms or editing free-text annotations in PDF, Adobe software ensures that only fonts having `fsType` information set to editable embedding are used. 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 uses only the 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 "Embedded Font Operations Using the Acrobat SDK" on page 4.

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.

Preserving information in embedded fonts

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 if 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
```

General descriptions of the terms used can be found in the glossary on page 4, but additional comments are included below.

`fsType/FSType`: 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 "Using fsType/FSType" on page 4 for details.

`WasEmbedded`: A new key written out in a PostScript stream to indicate that a font in it 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 type (`true` or `false`). If `WasEmbedded` is `true`, it is assumed that the font can be embedded. 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 let developers 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`, it is assumed that the font can be embedded. 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. `XUID`s are an optional font element. The font vendor identifier portion of each `XUID` number is provided by Adobe. `XUID`s are optional for Type 1 fonts, but font developers must put them in CID-keyed and PostScript OpenType fonts.

1 Specific Guidelines

This chapter provides guidelines that relate to the embedding of a specific font into a file, and then presents an example of how an Adobe application (Distiller for Adobe Acrobat) implements the guidelines. The guidelines cover embedding fonts when creating files for preview and print, filling forms, and editing free-text annotations. Adobe does not 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 (no `fsType/FSType` information). Third parties should compile their own lists for use in their software.

Embedding fonts in electronic files creation

The following guidelines should be followed when an application is *creating* an electronic file for preview and print.

Font	Minimum required	Highly recommended
PostScript/Type 1 (Western)/CEF/CFF	If embedding in PostScript stream, you can embed this font. Else if `fsType/FSType` exists, respect `fsType/FSType` settings. Else if the font `WasEmbedded`, you can embed this font. Else if `OrigFontType` exists, follow the rules in this document for the font as identified by `OrigFontType`. Else, allow the font to be embedded.	Default should be to subset font when the author is using less than 100% of the characters in the font. (Author can change this preference.)
OpenType (Western & CJK)/TrueType (Western & CJK)	If embedding in PostScript stream, you can embed this font. Else if `fsType/FSType` exists, respect `fsType/FSType` settings. Else if the font `WasEmbedded`, you can embed this font. Else if `OrigFontType` exists, follow the rules in this document for the font as identified by `OrigFontType`. Else, allow the font to be embedded.	Default should be to subset font when the author is using less than 100% of the characters in the font. (Author can change this preference.)

Font	Minimum required	Highly recommended
CID (CJK)/CID-CFF	If embedding in PostScript stream, you can embed this font. Else if `fsType/FSType` exists, respect `fsType/FSType` settings. Else if the font `WasEmbedded`, you can embed this font. Else if `OrigFontType` exists, follow the rules in this document for the font as identified by `OrigFontType`. Else if `XUID` exists and is on the approved list, you can embed this font. Else if `XUID` exists and is *not* on the approved list, do *not* embed. Else, do *not* embed this font.	Default should be to subset font when the author is using less than 100% of the characters in the font. (Author can change this preference.)
OCF (CJK)	This font is never embedded.	N/A

Embedding fonts in PDF forms and free-text annotations

The following guidelines should be followed for the type of embedding done for filling PDF forms and editing free-text annotations. This type of embedding is different from using embedded fonts for editing, which provides full text editing capabilities using an embedded font, something that Adobe does not currently support.

Only fonts whose `fsType` or related information indicates that they are editable can be used for forms text and free-text annotations.

Font	Minimum required	Highly recommended
PostScript/Type 1 (Western)/CEF/CFF	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 meets the following criteria, the font can be treated as editable: If the string: "…trademark of Adobe" is identifiable in the font, treat the font as editable. Otherwise if "Adobe" is in the notice string *and* the PostScript font name is on the approved list, treat the font as editable. Else (no `fsType/FSType`), assume the embedded font is preview and print only.	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)	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 the embedded font is preview and print only.	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.
CID (CJK)/CID-CFF	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 the embedded font is preview and print only.	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)	Not supported for forms.	N/A

Use of embedded fonts for editing

If the font is resident on a user's system, and may be embedded, allow editing using that font. Currently, 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 appears at the cursor insertion point when the user types . The user must 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 "Embedded Font Operations Using the Acrobat SDK" on page 4.

Distiller example

An application of the previously stated guidelines is demonstated in the following table about converting fonts from PS to PDF in Distiller. Using numbers to indicate relative priority, the table shows the search order used to determine whether and how fonts are 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 columns one and four of the table below.

	OpenType with PostScript outlines	**Type1 and CIDFontType0**	**Type42 and CIDFontType2[a]**	**TrueType and OpenType with TrueType outlines**
`FSType` in `FontInfo`	N/A	1	1	N/A
`FSType` in Font dictionary	N/A	2	2	N/A
OS/2 `fsType`	1	N/A	3	1
`WasEmbedded`	N/A	3	4	N/A
`OrigFontType`	N/A	4	5	N/A

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.

The method Distiller uses to embed fonts depends on the font type:

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`).

TrueType and TrueType-flavored OpenType fonts in TrueType format: The embedding information, if `FSType` or OS/2 is found, is preserved in the OS/2 table.

1 Using fsType/FSType

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` 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 chapter 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 "Specific Guidelines" on page 4.)

The `fsType/FSType` entry provides information regarding whether you can edit using the font, distribute the font freely, and so on. Adobe does not 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 people license font software from Adobe, or an Adobe application that includes font software, they are 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 (see http://www.adobe.com/type/browser/legal/embeddingeula.html). It is recommended that third parties provide the same information to their customers.

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. The bits are 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.

Bit	BitMask	Description
No bit is set.	`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.

Bit	BitMask	Description
1	`0x0002`	Restricted License embedding: Fonts that have *only* this bit set *must not be modified, embedded or exchanged in any manner* without first obtaining permission of the font software copyright owner. **Note:** For Restricted License embedding to take effect, it must be the only level of embedding selected.
2	`0x0004`	Preview and print embedding: When this bit is set, the font may be embedded, and *temporarily* loaded on the remote system. Documents containing preview and 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 *temporarily* on other systems. In contrast to preview and print fonts, documents containing editable fonts *may* 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 and print bit (`0x000C`). This permits an application that supports only preview and 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. For this reason, `fsType` values must be represented in formats that did not originally support them. The `FSType` entry (note different capitalization) uses the same values as the `fsType` entry, except that `FSType` is found in font types other than TrueType and OpenType. `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, Distiller uses the one in `FontInfo`. For Type 42 fonts, if `FSType` is not present in either dictionary, the `fsType` in an included OS/2 table is used. (See the discussion following the table on page 7.)

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` setting* that is used in the tables in "Specific Guidelines" on page 4.

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`], no embedding allowed (this setting explicitly allows for bitmap embedding, but Adobe applications do not currently support bitmap embedding),
3. Else if `fsType/FSType` bit 3 is `1` [`xxxx xx0x xxxx 1xxx`], editable embedding is allowed,
4. Else if `fsType/FSType` bit 2 is `1` [`xxxx xx0x xxxx 01xx`], preview and 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. Because Adobe products do not provide for installable embedding, the next least restrictive bit interpretation is used—editable embedding.

1 Embedded Font Operations Using the Acrobat SDK

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 and printing, as well as for 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 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, 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 must check whether the returned font is a system font. If it is, full editing can be allowed. If it is not, 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, and then checks progressively for greater and greater restrictions.

Plug-ins should subset fonts when they are used for preview and 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 1.1

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

Note: Do not use `PDEFontGetAttrs` to check the protection field bit in the `PDEFontAttrs` structure. The protection field is valid only 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 embedding bits for preview and print, or editing.
// Based on the font embedding bits, decide whether to embed or subset
// the font.

if ((pdeFontAttrs.protection & kPDEFontNoEditableEmbedding) == 0) {
// Editing OK. Embed the entire font.
fontCreateFlags = kPDEFontCreateEmbedded;
}
else if ((pdeFontAttrs.protection & kPDEFontNoEmbedding) == 0) {
// Preview and 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 1.2

The following example shows how to find a font that may be either installed on the system or embedded in a document, and then investigate the embedding information in the font. The key APIs demonstrated are `PDEFontGetSysFont` and `PDSysFontGetAttr`. To determine whether the font is installed on the system (as opposed to embedded in the document), use the `PDFindSysFont` method as shown in Example 4.1 on page 4.

Note: Do not use `PDEFontGetAttrs` to check the protection field bit in the `PDEFontAttrs` structure. The protection field is valid only 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 and print embedding OK.
```

```
//Editing is NOT OK.
else
/* Embedding not allowed... */
}
```

Glossary

C

CEF (Compact Embedded Font)

A font format used by SVG.

CFF (Compact Font Format)

A format 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)*.

CID (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.

CJK

An acronym used to identify the Chinese, Japanese, and Korean languages (the main Asian multi-byte languages).

E

EPS (Encapsulated PostScript)

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.

F

fsType/FSType

A reference to the `fsType/FSType` entry in a TrueType, OpenType, CFF, or CID-keyed font. See "Using fsType/FSType" on page 4 for details.

O

OCFs (Original Composite Fonts)

An older type of CJK PostScript font, pre-dating CID-keyed Type 1 fonts.

OpenType

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.

OrigFontType

A reference 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.

S

SVG (Scalable Vector Graphics)

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.

T

TrueType

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

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.

W

WasEmbedded

A refererence 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 "General Guidelines" on page 4.

X

XUID

A reference 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 "Specific Guidelines" on page 4) 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.

Index