Distiller® Font Management

Technical Note #5408

Distiller Version 4.05

ADOBE SYSTEMS INCORPORATED
Corporate Headquarters
345 Park Avenue
San Jose, CA 95110-2704
(408) 536-6000

Adobe Developer Technologies *September 14, 2000*

Date/Rev #	Comments
21 Aug. 2000	Initial release.
14 Sept. 2000	Moved description of indirect TrueType font support into a separate chapter. Added clarification about role of MakeCID. Also made minor clarifications and corrections throughout document.

Contents

1 About This Document

This document specifies the types of fonts Distiller 4.05 supports and describes Distiller considerations for finding, embedding, and subsetting those fonts. This document also describes the tasks Distiller performs to support fonts derived from TrueType fonts.

This document is intended for more experienced Distiller users who are also familiar with font technology.

Conventions

This document uses the following typographic conventions:

- **Serif bold** is used for dialog titles, such as **File** > **Preferences**.
- **San serif bold** is used for PostScript operators, such as **findfont,** or PostScript and PDF key words, such as **FontName**.
- `Courier` is used for file pathnames, such as `Acrobat 4.0\Resource\Fonts` and for Distiller messages.
- *Italicized serif* is used for terms that are being defined, for document titles, for Boolean values, and for emphasis.
- Blue text indicates a hypertext link within the document.
- Blue-underlined text indicates a hypertext link to another document or an action involving another application, such as a browser.

If You're Just Getting Started ...

If you are just getting started with Distiller, consider reading Distiller-related material in the *Adobe® Acrobat® 4.0 Guide*, available through the Acrobat **Help** pull-down window.

If you are just getting started using Asian fonts with Acrobat products, consider reading the document *Using Asian Fonts in Adobe Portable Document Format*. That 6-page document describes how to ensure portability of PDF files using Asian fonts and how to set up a system to support Asian fonts. *Using Asian Fonts in Adobe Portable Document Format* is available from http://www.adobe.com/products/acrobat/pdfs/acrasiafont.pdf.

If you're trying to find an answer to a simple Acrobat font question, you may find the 14-page document *Adobe 4.0 Frequently Asked Questions* helpful. That document is available from http://www.adobe.com/products/acrobat/pdfs/acr4faq.pdf.

If You're Trouble-Shooting ...

This document presents explanatory information, rather than trouble-shooting help. If you need help solving a specific problem, please search for applicable reports available from the Adobe web site, using the site's search engine http://www.adobe.com/support/database.html. In particular, you may find the following support database reports helpful:

322902 *How Acrobat 4.x Viewers Handle Fonts and Font Information*

315903 *Error "Subset is embedded..." or "...embedded font is not installed..." Using Touchup Tool in Acrobat*

321395 *TrueType Fonts Appear Incorrectly in PDF Created in Windows NT with AdobePS 5.0 or 5.1*

If You Want to Report Documentation Problems ...

You can help make this document better by sending your feedback and corrections to **doc_problems@adobe.com**.

Additional Resources

This document contains a glossary, which starts on page 41.

This document references the Web sites and documents listed in the following sections.

PostScript and Host Fonts

Information about PostScript fonts and host fonts is available from the following sources:

- Information about various font types, available from the Adobe Developers Association web site at http://partners.adobe.com/asn/developer/typeforum/main.html.
- Technical Note #5040, *Supporting Downloadable PostScript Language Fonts*, available from http://partners.adobe.com/asn/developer/technotes.html.
- Section 5.7 of the *PostScript Language Reference, Third Edition*, available from http://partners.adobe.com/asn/developer/technotes.html.
- *Type 1 Font Specification*, available from http://partners.adobe.com/asn/developer/technotes.html.
- OpenType Specification, available from http://partners.adobe.com/asn/developer/typeforum/opentype.html.
- OpenType specification for the OS/2 table, available from http://partners.adobe.com/asn/developer/opentype/os2.htm.

Fonts in PDF Files

Information about how PDF fonts are represented in PDF files is available from the following sources:

- Section 7.7 of the *Portable Document Reference Manual, Version 1.3*, available from http://partners.adobe.com/asn/developer/technotes.html.
- Technical Note #5004, *Adobe Font Metrics File Format Specification*, available from http://partners.adobe.com/asn/developer/technotes.html.
- *Using Asian Fonts in Adobe Portable Document Format*, available from http://www.adobe.com/products/acrobat/pdfs/acrasiafont.pdf.
- Technical Note #5641, *Enabling PDF Font Embedding for CID-Keyed Fonts*, available at http://partners.adobe.com/asn/developer/technotes.html.

2 Overview of Distiller Font Management

The overall goal in Distiller's handling of fonts is to ensure that PDF files contain enough information to allow Acrobat and Acrobat Reader® (henceforth collectively called *Acrobat*) to produce fonts that closely match the characteristics of the fonts in the original PostScript file and that those files be viewable on all platforms and be as small as possible.

To achieve that goal, Distiller (by default) produces PDF files that contain descriptions of the fonts referenced in the PDF file and that omit the (potentially large) fonts themselves. At the user's request or if Distiller deems it necessary, Distiller may also include fonts in PDF files, in which case those fonts are considered *embedded* in the PDF file.

The font descriptions Distiller creates in the PDF files it produces helps to ensure Acrobat can display and print glyphs associated with a font, even if that font is not embedded in the PDF file.

NOTE: A *glyph* is specific rendering of a character. For example, the glyphs A, **A**, and *A* are renderings of the abstract "A" character. Historically, the terms *character* and *glyph* have often been used interchangeably in computer typography, but advances in this area have made the distinction more meaningful.

PDF font description information includes the following:

- *Font name*. The PostScript **FontName** for the font.
- *Font metric data*. Quantitative descriptions of the glyphs drawn by the font. Such descriptions include the width of each glyph, the maximum height above the baseline reached by the glyphs, and the maximum depth below the baseline reached by the glyphs.
- *Font description flags*. These flags describe attributes of the font, such as whether it is a serif font and whether it is a fixed-width font, such as Courier.

When Acrobat opens a PDF file that contains an embedded font, it uses that embedded font to draw glyphs. However, when Acrobat opens a PDF file that does NOT contain an embedded font, it uses the description of the font to do one of the following:

- To find a font by the same name on the host system.
- To create a substitute font that matches most of the characteristics of the original font (called *font substitution*).

To produce PDF files that are as small as possible, Distiller subsets fonts. More specifically, when embedding a font, Distiller (by default) includes only the information required to draw the characters used in the PDF file. That task is called *font subsetting*.

The following sections introduce the Distiller tasks of font embedding and font subsetting and the Acrobat task of font substitution.

Font Embedding (a Distiller Task)

Distiller can embed fonts in PDF files (provided license restrictions are satisfied); however, such files can be much larger than equivalent files without embedded fonts.

When producing PDF files intended for use on other systems[1] (systems other than the one on which they originated), you may wish to direct Distiller to embed fonts, for the following reasons:

- *When the exact font is required (for visual or design reasons)*. Certain efforts, such as high-end printing, require Acrobat to use the exact font specified in the original PostScript file. For such situations, you should direct Distiller to embed the font in the PDF file. When Acrobat displays or prints a PDF file containing an embedded font, it uses the desired font, rather than another font on the host system or a substitute font.
- *When using non-Roman fonts or non-standard fonts (to ensure portability)*. When displaying or printing a PDF file, standard installations (English only) of Acrobat cannot create substitute fonts for non-Roman fonts, even for common Asian fonts; and, in general, Acrobat may be unable to create substitute fonts for unusual Roman fonts because it may not have enough information about those fonts. To ensure portability, you should embed non-Roman or non-standard Roman fonts in the PDF files that use them.

In addition to the user (you) directing Distiller to embed a font, Distiller may, on its own, decide to embed a font. Unless over-ridden by user directions, Distiller may embed fonts for the following reasons:

- *When Acrobat will be unable to create a substitute font*. Acrobat cannot create a substitute fonts for any font that uses special encoding or that contains characters that are not part of the Adobe Standard Latin Character set. (The Adobe Standard Latin Character Set is defined in Section E.5 of the *PostScript Language Reference, Third Edition*.) (This document uses the adjectives *Roman* and *Latin* interchangeably.)
- *When font substitution is likely to be unacceptable*. Font substitution is likely to be unacceptable for fonts that represent non-standard characters or special-purpose characters, such as icons, logos, and all-caps fonts. Such fonts may use standard encoding and Standard Latin Character Set characters; however, the glyphs they produce do not resemble the conceptual characters represented in that character set. For example, some companies customize fonts to include corporate logo glyphs in positions usually assigned to alphabetic characters.

When Acrobat opens a PDF file containing embedded fonts, it does not install those fonts on the system; rather, it uses those fonts to display or print only the file containing the embedded font. In other words, embedded fonts can neither be used to view or print other PDF files nor can they be used by other applications.

1. PDF files that are viewed only on the originating system need not have fonts embedded because all fonts referenced by the file are available to Acrobat.

Font Subsetting (a Distiller Task)

When embedding a font, Distiller (by default) subsets fonts — that is, it includes only the information required to draw glyphs for the characters used in the PDF file being produced. Font subsetting supports the goal of producing the smallest PDF files possible, yet still embed a version of the font that allows glyphs to be drawn as intended.

Distiller always subsets non-Type 1 fonts and Type 1 fonts that represent non-Roman characters; however, for Type 1 Roman fonts, you can direct Distiller to subset embedded fonts, as specified in the Distiller user interface.

You should avoid subsetting Type 1 fonts in the following situations:

- *PDF file might be edited*. The Acrobat touch-up tool allows users to modify text, including inserting new characters and replacing existing characters with other characters. If a user modifies text to include characters that were not contained in the embedded and subsetted font, Acrobat would be unable to display or print those new characters.
- *PDF files might be merged*. Acrobat allows multiple PDF files to be merged into one. That capability is used to insert pages from one PDF file into another PDF file (**Document > Insert Pages…**) or to replace pages from one PDF file with pages from another (**Document > Replace Pages…**).

  When Acrobat merges two PDF files, each containing a subsetted version of the same font, it produces a new PDF file that retains both subsetted fonts. The net size of the two subsetted fonts may be larger than the full font would have been.

Font Substitution (an Acrobat Task)

Acrobat tries to create a substitute font if the font referenced in the PDF file is not embedded and if Acrobat cannot find the font on the host system. Acrobat may be unable to find a font if the font is not installed on the host system or activated by ATM or if the user has instructed Acrobat not to use the fonts on the host system.

Acrobat creates a substitute font by applying the font descriptions in a PDF file to another font installed on the host system. As described in the following sections, the font Acrobat uses for font substitution and how it applies the information in the font descriptions depends on whether the original font represents Roman characters or Asian ones. Acrobat can create substitute fonts only for Roman and CJK fonts.

Acrobat generates substitute fonts for display on screen or for printing; it never embeds substitute fonts in PDF files. Further, substitute fonts can only be used by Acrobat — they cannot be used in other applications.

Roman Fonts

When creating a substitute font for a Roman font, Acrobat applies the information in the font description to a multiple master typeface to obtain another font that (usually) closely

resembles the font referenced in the PDF file. Acrobat uses the AdobeSerMM multiple master typeface to substitute for serif fonts, and AdobeSanMM, for sans-serif fonts.

Acrobat manipulates a substitution font to exactly match the character widths of the original font and to closely match the weight, angle, and general proportions of the original font. The result is a substitute font that produces glyphs that preserve the line and page breaks from the original document and that preserves the overall look and feel of the original document.

Acrobat can produce Roman substitute fonts only for fonts that use the Adobe Standard Latin Character Set.

Asian Fonts

When creating a substitute font for an Asian font, Acrobat uses one of the CJK fonts installed with Acrobat. Such fonts are bundled with the Acrobat product or can be downloaded for free with the Asian font packs. Unlike the fonts used to create Roman substitute fonts, the CJK fonts are not multiple master fonts.

Using the font description from the PDF file, Acrobat chooses the substitute font that most closely matches the characteristics of the original font. Then, Acrobat creates a new instance of the font in which it modifies information that cause the glyphs in the font to be stretched or shrunken to more closely resemble the original font.

3 Supported Fonts

This chapter describes the fonts supported by Distiller. The first section, "Host Font Conversion," defines important terminology used throughout this document and the relationship between host fonts and PostScript language font programs, henceforth called PostScript fonts. The next section, "Supported Fonts," describes the fonts supported by Distiller. The next chapter, "Indirect Support for TrueType Fonts," contains related information.

"Host Font Conversion" is presented first because your understanding of font conversion is important. Distiller font management is described relative to PostScript fonts, not host fonts. More specifically, if you want to know how Distiller handles a certain host font, you must search for descriptions of the PostScript font into which the host font is converted.

Host Font Conversion

In general, Distiller supports only PostScript fonts. Such fonts may be embedded in PostScript files and/or may reside on the Distiller's host system.

Host fonts are any fonts installed on a system. Such fonts can be installed at various times: with the operating system, with font packages, and with applications. Because host fonts are usually compressed, they occupy far less space than their PostScript font counterparts. Distiller can process host fonts that are also PostScript fonts (for example, Type 1 / PFA).

*PostScript fo*nts are fonts that can be directly interpreted by Distiller and by PostScript printers. They are the format PostScript drivers produce either for printing to a PostScript printer or for saving in a PostScript file. In general, PostScript fonts are embedded in PostScript files; however, some types of PostScript fonts also exist as host fonts (usually on UNIX systems).

NOTE: The Windows PostScript driver user interface (UI) for downloading host fonts to a PostScript printer may have left you with the impression that the driver is capable of downloading host fonts, as is. What is left unsaid is that the PostScript driver converts a non-PostScript host font into a PostScript font before downloading it to a printer.

Font conversion is the process of converting a font from one representation to another. It is done only for fonts that are being embedded in a PostScript file or a PDF file. As illustrated in Figure 3.1, fonts may go through several conversions.

FIGURE 3.1 Font conversions

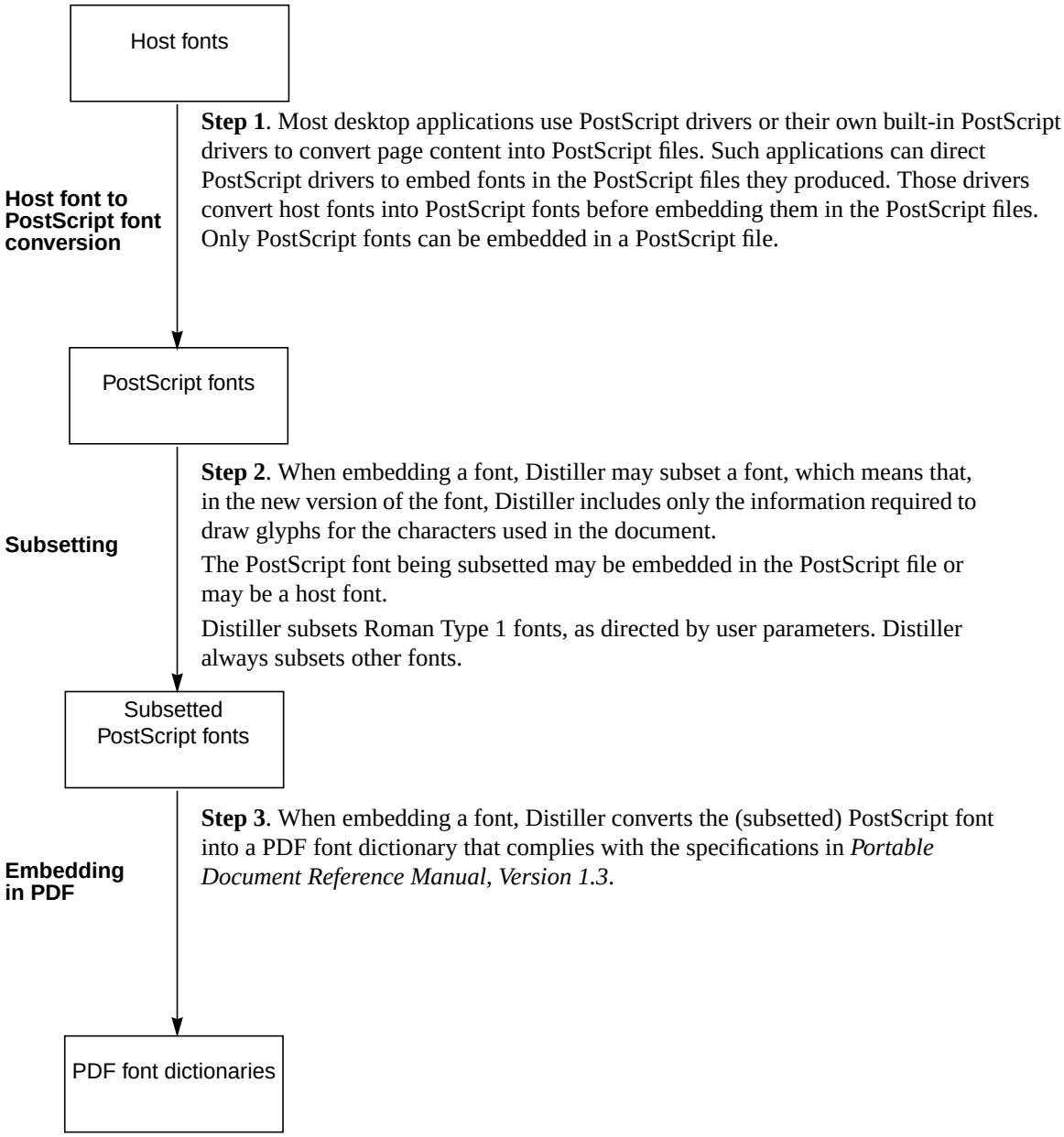
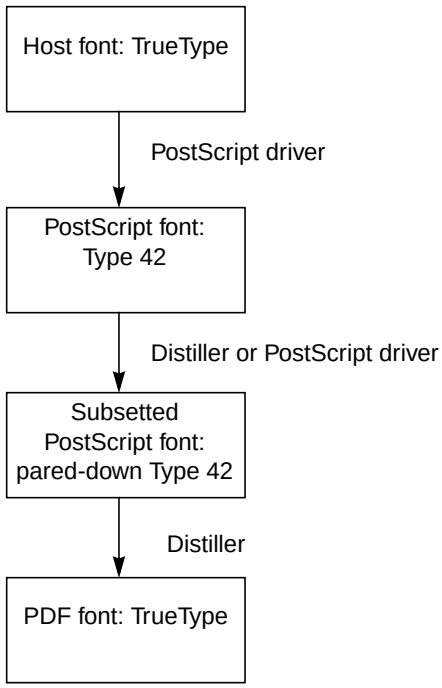


Figure 3.2 provides an example of font conversion for a TrueType font when user parameters specify the font is to be embedded (as a Type 42/CIDFontType2 font) and subsetted in the PDF file produced.

FIGURE 3.2 Font conversion example: TrueType font that is embedded and subsetted

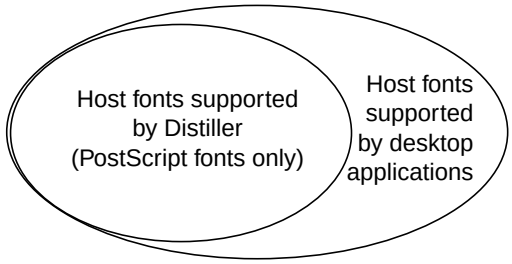


Supported Fonts

Distiller supports most types of PostScript fonts that exist as fonts embedded in PostScript files, as described in "Embedded PostScript Fonts Supported by Distiller." Distiller also supports some types of PostScript fonts that exist as host fonts, as described in "Host PostScript Fonts Supported by Distiller." The term *supported* means that Distiller is able to find the font, produce a PDF file that contains a font description for the font, and embed the font in a PDF file.

As illustrated in Figure 3.3, Distiller supports a subset of the host fonts supported by desktop applications. Distiller uses an internal mechanism to find fonts, rather than using system utilities. The use of that mechanism prevents Distiller from finding host fonts that are not also PostScript fonts. (See also the Q&A discussion, "No Support for TrueType Fonts But UI Lists Them" on page 37.)

FIGURE 3.3 Comparing Distiller and desktop application host-font support



Embedded PostScript Fonts Supported by Distiller

For all platforms (Windows, Macintosh, and UNIX®) Table 3.1 describes the PostScript fonts supported by Distiller, provided those fonts are embedded in a PostScript file. If Distiller encounters an embedded PostScript font that is NOT supported, it produces a PostScript error, such as `undefined` or `invalidfont`.

NOTE: You can use Adobe Type Manager Delux to determine the font type for a particular font. Select the font in question, and then select **Window > New Sample Window**. The font type appears as the entry for the **Technology** field.

NOTE: You can obtain more information about each font type from the Adobe Developers Association web site at http://partners.adobe.com/asn/developer/typeforum/main.html.

TABLE 3.1 PostScript fonts supported by Distiller as fonts embedded in a PostScript file

PostScript Font	Comments
Type 1 / PFA	ASCII-encoded Type1 font
Type 3	PostScript outline or bitmap fonts. Type 3 outline fonts can be installed on a host system. Type 3 bitmap fonts are created only by PostScript drivers on Windows systems and only in the following situations: ● Printer memory is limited ● User has set printer driver preferences to allow use of bitmap fonts. Type 3 bitmap fonts are not recommended. They produce poor quality glyphs and require more time to render than their outline counterparts.
Type 42	Roman-character TrueType font in a PostScript wrapper
CIDFontType0	CID-keyed font with Type 1 charstrings.
CIDFontType2	CID-keyed font with TrueType charstrings. PostScript drivers create such fonts from host double-byte TrueType fonts.

Host PostScript Fonts Supported by Distiller

Table 3.2 and Table 3.3 describe the host fonts supported by Distiller for Windows and Distiller for Macintosh, respectively. Distiller host font support requires that Adobe Type Manager® (ATM®) version 4.0 or later be installed on the host system.

Differences arise between the host fonts supported on each platform largely because of the different font types supported by each platform. Table 3.2 and Table 3.3 also describe the PostScript font types into which host fonts are converted.

Table 3.2 or Table 3.3 omit several PostScript fonts that appear in Table 3.1 because not all PostScript fonts exist as host fonts. For example, the font type CIDFontType2 exists only as an embedded font; it never exists as a host font.

TABLE 3.2 Windows host fonts supported by Distiller

Font type / file type	Description	Retail Font Product	Converted into PostScript font
Type 1 / PFB	PFB file containing a PostScript font program that includes segment headers followed by binary font data	Adobe Type Library Type on Call Adobe FontFolio™	Type 1 / PFA
Type 3			No conversion required.
CIDFontType0	CID-keyed fonts not included in sfnt resources.	Type on Call, 1.0J Morisawa ViewFonts	No conversion required.

TABLE 3.3 Macintosh host fonts supported by Distiller

Font type	Description	Retail Font Product	Converted into PostScript font
Type 1 / LWFN	Type 1 font in a compressed binary format.	Adobe Type Library Type on Call Adobe FontFolio	Type 1
CIDFontType0	CID-keyed fonts not included in sfnt resources. (Such fonts are also called *naked CID fonts*.)	Type on Call, 1.0J Morisawa ViewFonts	No conversion required.
CID-keyed with an sfnt wrapper	CID-keyed font in a Macintosh sfnt resource wrapper that includes additional sfnt layout tables.	Adobe Type Library, 2.0J	CIDFontType0 (without extra layout tables)

4 Indirect Support for TrueType Fonts

Distiller does not directly support TrueType fonts, in that it cannot find them or embed them; however, it does provide indirect support that allows it to produce PDF files that represent fonts that are TrueType fonts or are derived from TrueType fonts. Those capabilities are described in "Determining Glyph Placement for Unembedded TrueType Fonts" and "Retrieving Omitted Information for Type 42 and CIDFontType2 Fonts" on page 20.

Determining Glyph Placement for Unembedded TrueType Fonts

Although Distiller does not support CJK TrueType fonts, Acrobat does. However, Distiller has a work-around that allows it to produce PDF files that describe CJK TrueType fonts adequately to allow Acrobat to view or print those fonts. This work-around involves Distiller's use of WidthOnly fonts.

Distiller does not support TrueType fonts because it cannot find them, so it cannot use them to determine where on the page to place glyphs. However, Distiller has access to a set of *WidthOnly fonts* that reflect the horizontal spacing characteristics of any Roman characters[1] in the original font. Distiller uses the WidthOnly fonts to determine glyph placement for CJK TrueType fonts.

Acrobat can only display or print fonts not embedded in the PDF file if the specified font is available to Acrobat. The desired font may be available to Acrobat if the appropriate language kit is installed with Acrobat, if Acrobat is running in the native language, or if the original font has been installed and activated on the Acrobat host (and the user has allowed Acrobat to use host fonts).

Distiller is installed with a set of WidthOnly fonts, one for each Adobe CJK TrueType font in the Adobe Type Library and one for each Macintosh and Windows CJK system font. Further, the Mac Distiller utility MakeCID can produce WidthOnly fonts for any font not represented in the existing WidthOnly fonts. See the *Adobe Acrobat 4.0 Guide* for instructions on using MakeCID.

The WidthOnly fonts produced by MakeCID on Macintosh systems may be ported to Windows systems, as described in *Using Asian Fonts in Adobe Portable Document Format*.

WidthOnly fonts cannot be embedded.

1. CJK TrueType fonts may be composed of mono-spaced Asian glyph descriptions and proportional Roman glyph descriptions. The WidthOnly fonts describe the horizontal spacing of only Roman glyphs, they do not describe the horizontal spacing of of the Asian glyphs.

Retrieving Omitted Information for Type 42 and CIDFontType2 Fonts

Distiller implements additional processing for PostScript fonts derived from TrueType fonts (Type 42 and CIDFontType2). Such processing is required to address modifications or omissions that can occur when older PostScript drivers convert TrueType fonts into their PostScript font counterparts.

As described in Figure 3.1 on page 14, desktop applications can direct PostScript drivers to include the PostScript font equivalents of TrueType fonts in the PostScript files they produce. Such PostScript fonts are always embedded in the PostScript files that use them.

PostScript drivers convert TrueType fonts into Type 42 and CIDFontType2 fonts. Table 4.1 and Table 4.2 describe the TrueType fonts supported by PostScript drivers.

TABLE 4.1 TrueType fonts supported by Windows PostScript drivers

Font type / file type	Description	Retail Font Product	Converted into PostScript font
TrueType	TrueType font **NOTE:** PostScript drivers convert Roman TrueType fonts into Type 42, and Chinese-Japanese-Korean (CJK) TrueType fonts into CIDFontType2.	Adobe WebType™ OS bundle 3rd party	Type 42 and CIDFontType2
TrueType Collection (CID-keyed)	TrueType Collection font containing individual TrueType fonts. **NOTE:** No Roman TrueType Collections are currently produced.	OS bundle 3rd party	CIDFontType2

TABLE 4.2 TrueType fonts supported by Macintosh PostScript drivers

Font type / file type	Description	Retail Font Product	Converted into PostScript font
TrueType (Roman)	TrueType font.	Adobe WebType OS bundle 3rd party	Type 42

Before a PostScript driver embeds a host font in a PostScript file, it converts the host font into its PostScript font equivalent. During that conversion, some older drivers produce fonts that may modify or omit information contained in the original font, such as the precise font name, license restrictions, and Unicode information. If Distiller cannot find the information it needs

in the PostScript font, it tries to obtain it from the original TrueType font. If Distiller is unable to find the original TrueType font, it may be unable to successfully process a PostScript file.

NOTE: To avoid problems that might arise from Distiller being unable to find an original TrueType font, you should always run Distiller on the same computer on which the PostScript file was created, even if that TrueType font was embedded in the PostScript file as a Type 42 or CIDFontType 2 font.

Distiller may be unable to find the original TrueType font in the following situations:

- PostScript file (containing embedded Type 42 or CIDFontType2 fonts) is produced on one system and then processed by Distiller on another that lacks the original TrueType font.
- Directory containing the TrueType font is changed or updated after the PostScript file was created. Such changes/updates can happen in the following situations:
  - On a system running Windows 95, you create a PostScript file that contains references to TrueType fonts. Later, you upgrade your system to Windows 98 and then use Distiller to process that PostScript file. Distiller cannot find the original Roman TrueType fonts because the operating system upgrade changed them.
  - You create a PostScript file that contains references to TrueType fonts. Later, you enable multi-lingual support through the system **Control Panel** window and then use Distiller to process that PostScript file. Distiller cannot find the original TrueType fonts because they were changed when you enabled multi-lingual support.

The following sections describe the data that older PostScript drivers may omit or change when converting TrueType fonts and describe how Distiller addresses such omissions.

PostScript Font Name

In the Type 42 or CIDFontType2 fonts they create, older PostScript drivers may modify the font name from that contained in the original TrueType font. More specifically, Type 42 and CIDFontType2 fonts contain two name-related fields — **OrigFontName** and **FontName** — that reflect but may not match the PostScript font name. An example of font name modification appears in Figure 4.1.

FIGURE 4.1 Font name modification in TrueType fonts

Distiller always attempts to obtain[2] the font name from the original TrueType font. If successful, Distiller changes in the PDF file font references and font dictionaries to use the original font name.

Distiller attempts to use the font name of the original TrueType font for the following reasons:

- *Consistent with Distiller UI*. User specifies host font names in the **AlwaysEmbed** and **NeverEmbed** lists. Such names rare the same as the original TrueType font name.
- *Helps Acrobat use the right font*. If the font is not embedded, the original font name allows Acrobat to find the corresponding TrueType font, if that font is installed on the Acrobat host system.

If Distiller is unable to obtain the original TrueType font name, it instead uses the font name in the Type 42 or CIDFontType2 font.

License (Embedding) Restrictions (fsType)

In the Type 42 or CIDFontType2 fonts they create, older PostScript drivers may not retain the license restrictions contained in the original TrueType font.

TrueType fonts (as well as CID-keyed fonts) may contain license restrictions represented in an integer called **fsType**, which is an optional entry in the **OS/2** table. **fsType** specifies license restrictions for the font, including whether the font may be embedded. The **OS/2** (and **fsType)** specification is described at http://partners.adobe.com/asn/developer/opentype/os2.htm.

If **fsType** is missing in a Type 42 or CIDFontType2 font, Distiller attempts to find **fsType** in the original TrueType font. If Distiller cannot find the original TrueType font, it assumes license restrictions are satisfied. "Satisfying License Restrictions" on page 32 describes how Distiller interprets **fsType** information when determining whether to embed a font.

Unicode Information

In the CIDFontType2 fonts they create, older PostScript drivers may not retain the Unicode cmap table contained in the original TrueType font. A *Unicode cmap* maps from Unicode character codes to glyph descriptions and allows text strings that use the font to be converted to Unicode values for export to other applications or plug-ins.

If the ToUnicode cmap table is missing from a CIDFontType2 font, Distiller attempts to find the missing information in the original TrueType font. If Distiller cannot find the original TrueType font, it omits the Unicode cmap from the font.

---

2. Distiller finds the original TrueType font using information unrelated to the font name. More specifically, Distiller matches the Type 42 or CIDFontType2 font with the original TrueType font by comparing the values of checksums in four tables in the **sftnts** header. Those tables are **head**, **hhea**, **hmtx**, **maxp**.

5 Font Management Tasks

This chapter describes the font management tasks performed by Distiller, which include the following:

Searching for PostScript Fonts

This section describes why Distiller needs to find PostScript fonts, where it searches for them, what happens when multiple identically-named PostScript fonts exist on the system, and what Distiller does when it cannot find a PostScript font.

Even though Distiller does not print any PostScript files, it still needs to find the PostScript fonts used in those files, for the following reasons:

- *For glyph placement*. The font contains information about horizontal spacing (*character width*), which Distiller uses to determine where on the page to place glyphs. Information on horizontal spacing is required whether or not fonts are embedded. (The term *character width* evolved before the terms *glyph* and *character* were differentiated; hence, we must tolerate a commonly used term that is, in a literal sense, inaccurate.)
- *For font embedding*. At the user's direction or for other reasons, Distiller may embed a font in the PDF file. Further, before embedding some types of fonts, Distiller checks information in individual fonts to determine whether the font's manufacturer allows the font to be embedded.
- *To support Acrobat font substitution*. Distiller may use information in fonts to populate entries in the PDF dictionaries that describe fonts. Acrobat may use such entries for font substitution.

This section describes only Distiller's search for PostScript fonts. It does not describe how Distiller searches for TrueType fonts, as described in "Retrieving Omitted Information for Type 42 and CIDFontType2 Fonts" on page 20.

Note to Reader About Focus on PostScript Fonts

The remainder of this document describes Distiller behavior relative to PostScript fonts, which may be embedded in PostScript files and may exist on the host system. Because of the focus on PostScript fonts, any use of the term *fonts* without further differentiation is intended to

mean *PostScript fonts*. Other types of fonts, such as TrueType, are described in a way that differentiates them from PostScript fonts.

If you are unsure about the relationship between host fonts and their PostScript font counterparts, please read "Host Font Conversion" on page 13. Also, Table 3.2 and Table 3.3 provide a mapping from host fonts into PostScript fonts.

Parameters That Affect Font Searching

Distiller determines whether the font was embedded in the PostScript file, and then searches in the directories specified in the **Font Locations** window. The **Font Locations** window can be accessed by selecting **Settings** > **Fonts**. Users can change the list, by adding directories, removing directories, or changing the order of directories. Table 5.1 shows the default directories in the **Font Locations** window.

TABLE 5.1 Default font locations

Platform	Default font locations
Windows	`Acrobat 4.0\Resource\Fonts` ATM font folders (for example c:\psfonts)
Macintosh	`Acrobat 4.0:Resource:Font` `System Folder:Fonts`
UNIX	Directory specified in the environment variable `PSRESOURCEPATH`

Rules for Searching

When a system contains several identically-named fonts, the order in which Distiller searches for fonts determines which of those fonts Distiller selects. For example, if you have three versions of Helvetica®, one embedded in the PostScript file and the other residing in other folders, Distiller selects the first one it finds, which in this example is the one embedded in the PostScript file.

NOTE: Despite their promising names, the font entries ***UniqueID*** and ***XUID*** cannot be used to verify Distiller has selected the right font. Those entries are contained in the fonts themselves, not in the font references. The PostScript interpreter uses ***UniqueID*** and ***XUID*** in font caching, as described in the PostScript Language Reference, Version 3.0.

The following subsections describe for each platform and font type, where Distiller looks for fonts and what happens when multiple identically-named fonts are available.

Type 1, Type 3, and Type 42 (Usually Roman) Fonts on Windows or Macintosh

Roman PostScript fonts include Type 1, Type 3, and Type 42 fonts. Those fonts may also be used for other character sets, although such use is unconventional.

As described in the following steps, Distiller searches for Roman fonts, stopping when it finds the desired font:

1. Examines fonts embedded in the PostScript file, comparing **FontName** in the font reference with **FontName** in the individual embedded fonts.
2. Searches the directories specified in the **Font Locations** window, comparing **FontName** in the font reference with **FontName** in the individual fonts.
3. Same as Step 2, but compares **FontName** in the font reference with the font file names.
4. Distiller determines whether the font is represented in the ATM font substitution database. If it is not, Distiller replaces references to the font with Courier. No references to the missing font are included in the PDF file. As a result, Acrobat uses Courier instead of the missing font, even when Acrobat is running on a system that has the missing font.

CID-Keyed PostScript Fonts and CMap Files on Windows

CID-keyed PostScript fonts include the font types CIDFontType0 and CIDFontType2.

CMap files are used with CID-keyed fonts to define the mapping from individual character codes to a font number and a character selector. (A CMap serves the same function for a CID-keyed font as an encoding table does for a Type 1 font.)

On Windows systems, Distiller searches for CID-keyed fonts or CMap files in the following locations, stopping when it finds the desired font:

1. Examines fonts embedded in the PostScript file, comparing **FontName** in the font reference with **FontName** in the individual embedded fonts.
2. Searches ATM folders, comparing **FontName** in the font reference with **FontName** in the individual fonts. (ATM is the only mechanism for installing CID-keyed fonts on Windows systems. As a result, Distiller for Windows searches only the ATM folders for CID-keyed fonts and CMap files.)
3. Same as Step 2, but compares **FontName** in the font reference with the font file names.
4. Searches the default folder `Distiller\Data\PSDisk\Resource` for WidthOnly fonts. *WidthOnly fonts* contain information similar to the font metric data used with Roman fonts. Distiller uses such fonts to determine horizontal spacing of the characters; it cannot embed WidthOnly fonts.

   The Macintosh version of Distiller uses the MakeCID utility to produce WidthOnly fonts for any Asian fonts installed on the system.

ATM prevents multiple identically-named CID-keyed fonts from being installed on a system. Because ATM is the only mechanism for installing CID-keyed fonts on Windows systems, identically-named CID-keyed fonts in the ATM folders is not an issue.

CID-keyed PostScript Fonts on Macintosh

CID-keyed PostScript fonts include the font types CIDFontType0 and CIDFontType2.

On Macintosh systems, Distiller searches for CID-keyed fonts in the following locations, stopping when it finds the desired font:

1. Examines fonts embedded in the PostScript file, comparing **FontName** in the font reference with **FontName** in the individual embedded fonts.
2. Searches through font folders in the **Font Locations** window, comparing **FontName** in the font reference with **FontName** in the individual fonts. Distiller searches the directories, following the order specified in the **Font Locations** window.
3. Same as Step 2, but compares **FontName** in the font reference with the font file names. Distiller searches the directories, following the order specified in the **Font Locations** window.
4. Distiller searches the WidthOnly folder, `Distiller:Data:psdisk:Resource:CIDFont`. Distiller uses the WidthOnly folder to determine horizontal spacing of the characters; it does not embed WidthOnly fonts.

CMap Files on Macintosh

On Macintosh systems, Distiller searches for CMap files in the following locations, stopping when it finds the desired file:

1. Examines fonts embedded in the PostScript file. (Embedded fonts may contain CMaps.)
2. Searches the folder `Distiller:Data:psdisk:Resource:CMap`.
3. Searches the font folders described for Macintosh in Table 5.1.

Obtaining Font Metric Data

This section explains how Distiller obtains font metric data and other information that Acrobat uses for font substitution. The meaning of *other information* is defined later in this section.

Font metric data is a set of numbers that provide quantitative descriptions of the glyphs drawn by the font. Such descriptions include glyph-specific information, which is the width of each glyph drawn by the font (character width), and general information, such as the maximum height above the baseline reached by the glyphs (**Ascent**), the maximum depth below the baseline reached by the glyphs (**Descent**), and the font bounding box (**FontBBox**). **FontBBox** is the smallest rectangle enclosing the shape that would result if all glyphs drawn by the font were placed with their origins coincident, and then painted. Font metrics are described in the *PostScript Language Reference, Version 3*.

For Roman fonts, other information that Acrobat uses for font substitution includes whether the font is a serif font and what character set it uses. For CJK fonts, other information includes the style of the glyphs in the font (which is described as a Panose number), the language of the font, and the subset of characters. Henceforth, the term *font metric data* includes the non-

quantitative information that describes a font. Such usage is not entirely accurate; however, it eliminates some wordiness.

The following sections describes how Distiller obtains font metric data for Roman fonts and for CJK fonts.

For Roman Fonts

For Roman fonts that use standard encoding and that represent characters from the Adobe Standard Latin Character Set, Distiller obtains font metric data from the fonts themselves and/or from the Adobe Type Manager (ATM) database. The following paragraphs explain whether Distiller obtains font metric data from both, one, or the other sources.

Distiller cannot obtain font metric data for Roman fonts that use non-standard encoding or represent fonts not in the Standard Latin Character Set; rather, Distiller always embeds such fonts because Acrobat cannot create equivalent substitute fonts. (See also "Embedding Fonts" on page 28.)

The *ATM font substitution database* is a file that contains descriptions for commonly-used Type 1 Roman fonts. The Windows versions of the database contains entries for 1,990 fonts, and the Macintosh version, 1,831. The ATM font substitution database lacks descriptions for only an estimated 300 Adobe Type 1 Roman fonts developed since 1994.

The ATM font substitution database is installed automatically with Distiller.

If Distiller finds the font and finds an entry for the font in the ATM font substitution database, it obtains character width from the font and other data from the ATM font substitution database. Character width is a vector that describes the width of a specific glyph.

If Distiller cannot find the font in the ATM font substitution database, it obtains all font metric data from the font. (See also "Selection of Sources for Font Metric Data" on page 38.) If Distiller cannot find the font, it uses Courier as a substitute font.

For CJK Fonts

For CJK fonts, Distiller obtains font metric data from the font or, if not found, from the WidthOnly font. Then, Distiller tries to obtain any missing font metric data from the J font database, which is internal to Distiller (as opposed to being a separate file). (As described earlier, the term *font metric data* includes the non-quantitative information that describes a font.)

The J font database, which is relatively small, contains information for approximately 610 Adobe Japanese PostScript and TrueType fonts developed before 1997. (It does not contain information on any CJK fonts developed since 1997.) Entries in the J font database contain the following information: PostScript **FontName**, OS name, Panose number, **Ascent**, **CapHeight**, **xHeight**, **Descent**, **StemH**, and **StemV**.

Distiller uses PostScript **FontName** to find an entry in the CJK font database. If Distiller cannot find a database entry for a font, it uses default values for any font metric data not represented in the font.

The information in the J font database is not very important because Acrobat requires that CJK fonts referenced from a PDF file either be embedded in that file or reside on the system as a scalable host font (from one of the Asian font packs).

Embedding Fonts

This section describes when and why Distiller embeds fonts and what happens when Distiller is unable to embed a font that should be embedded.

Parameters That Affect Font Embedding

Distiller is controlled through Distiller parameters that can be provided by the job options file selected by the user or by the PostScript file being converted. For each PostScript file it converts, Distiller resets the Distiller parameters with the currently selected job options file. As a result, Distiller parameters in the PostScript file being converted may change Distiller parameters set by the job options file.

On Windows and Macintosh systems, users can create new job options files with modified Distiller parameters, using the Distiller UI. (Distiller on UNIX does not provide a UI; rather, its run-time parameters may specify the name of a file containing a short PostScript job that defines Distiller parameters.)

Table 5.2 reviews the Distiller parameters that affect font embedding, which are summarized from Technical Note #5151, *Acrobat Distiller Parameters*. Table 5.3 reviews font parameters that can affect embedding.

TABLE 5.2 Distiller parameters that affect font embedding

Parameter Name	Description
AlwaysEmbed	An array of font names that the Distiller must always attempt to embed. (See Note following table.) You can modify the font names in the **AlwaysEmbed** list, by selecting **Settings > Job Options… > Fonts** dialog in the Distiller UI. Select the type or location of font from the **Embedding** window and then move that font to the **Always Embed** window. You can also augment the **AlwaysEmbed** list with fonts not installed on the system, by selecting the **Settings > Job Options… > Fonts > Add Name…** dialog. Enter the name of the font that you want embedded and select the **Always Embed list** radio button. That UI feature allows you to direct Distiller to embed fonts that are embedded in the PostScript file but is selectable only if **Embed All Fonts** has not been selected.
CannotEmbedFontPolicy	Specifies how the Distiller should respond if a font cannot be found or embedded. Possible values are: ● OK, ignore if a font cannot be found or embedded ● Warning, warn and continue if a font cannot be found or embedded ● Error, terminate the current job if a font cannot be found or embedded The Distiller UI allows users to modify the value of **CannotEmbedFontPolicy** by selecting **Settings > Job Options… > Fonts > When Embedding Fails**.
EmbedAllFonts	If *TRUE*, the Distiller embeds all fonts except those in the **NeverEmbed** list. (See Note following table.) You can modify the value of **EmbedAllFonts** by selecting **Settings > Job Options… > Fonts > Embed All Fonts** in the Distiller UI.

TABLE 5.2 Distiller parameters that affect font embedding *(Continued)*

Parameter Name	Description
NeverEmbed	An array of font names that the Distiller should not embed. (See Note following table.) You can modify the font names in the **NeverEmbed** list, by selecting **Settings** > **Job Options…** > **Fonts** dialog in the Distiller UI. Select the type or location of font from the **Embedding** window and then move that font to the **Never Embed** window. You can also augment the **NeverEmbed** list with fonts not present on the system, by selecting the **Settings** > **Job Options** > **Fonts** > **Add Name…** dialog. Enter the name of the font that you don't want embedded and select the **Never Embed list** radio button. That UI feature allows you to direct Distiller to NOT embed fonts that are embedded in the PostScript file but is selectable only if **Embed All Fonts** has not been selected. **NOTE:** The limitation described in the previous sentence is a Distiller bug. You should be able to specify a font using the **Settings** > **Job Options** > **Fonts** > **Add Name…** dialog when **Embed All Fonts** has been selected.
CompatibilityLevel	Specifies the version of PDF that Distiller should produce. Possible values are 1.2 or 1.3. Those versions correspond to versions of the *Portable Language Format Reference*. You can modify **CompatibilityLevel** using the **Settings** > **Job Options…** > **General** > **Compatibility** dialog in the Distiller UI. Selecting **Acrobat 3.0** specifies version 1.2, and selecting **Acrobat 4.0** specifies version 1.3.

NOTE: Distiller resolves ambiguity in the parameters that specify font embedding, by giving the **NeverEmbed** priority over both the **EmbedAllFonts** flag and the **AlwaysEmbed** list. For example, if a font is in the **NeverEmbed** list, it will not be embedded, even if the **EmbedAllFonts** flag is *TRUE* or if the font is in the **AlwaysEmbed** list.

TABLE 5.3 Font parameters that affect font embedding

Parameter Name	Description
fsType	A set of flags that indicates whether the font can be embedded and whether subsetting is allowed. **fsType** is generally required with TrueType CID-keyed fonts and CIDFontType0 fonts. **fsType** is optional with TrueType Roman fonts. **fsType** is further described in http://partners.adobe.com/asn/developer/opentype/os2.htm
Encoding	An array that associates character names with character codes. Character codes, in turn, map to glyphs. Character encoding can be changed without altering anything else in the font program.
Character Set	A conceptual collection of characters, such as the Adobe Standard Latin Character Set, described in Appendix E of the *PostScript Language Reference, Third Edition*.

Rules for Embedding

The following sections describe the rules Distiller uses to determine whether to try to embed a font of a particular type.

NOTE: Fonts embedded in the PostScript file are embedded in the PDF file only if embedding rules dictate that those font should be embedded. In other words, Distiller does not embed a font in the PDF file simply because that font is embedded in the PostScript file.

In general, if Distiller cannot embed a font that should be embedded, it responds as specified in the Distiller parameter **CannotEmbedFontPolicy**; however, some fonts must be embedded to produce a readable PDF file, regardless of Distiller parameters. Distiller terminates a job with error when it cannot embed a font that must be embedded. The following sections identifies those fonts that must be embedded.

Type 1 Fonts

Distiller embeds a Type1 font if it is NOT in the **NeverEmbed** list, the font was found, and at least ONE of the following conditions is true:

- **EmbedAllFonts** flag is *true*.
- Font is in the **AlwaysEmbed** list
- PostScript file uses the font for characters that are not included in the Standard Latin Character Set.
- Font contains many glyph definitions (**CharStrings** dict length > 229). Such fonts are usually used for non-standard glyphs.
- Font contains few glyph definitions (**CharStrings** dict length < 115). Such fonts are used for logos or special glyph sets, such as an all-capital letter font.

Type 3 Fonts

Type 3 fonts are always embedded. Type 3 fonts are used to represent non-standard characters for which Acrobat is unable to create substitute fonts.

CIDFontType0 Fonts

Distiller embeds a CIDFontType0 font if ALL of the following conditions are true:

- Font was found (rather than being a WidthOnly font) and its license restrictions are satisfied ("Satisfying License Restrictions" on page 32).
- Distiller is producing PDF 1.3. (CIDFontType0 fonts cannot be embedded in earlier versions of PDF.)
- Font is NOT in the **NeverEmbedList**.
- ONE of the following conditions is true:
  - **EmbedAllFonts** flag is *true*
  - Font is on the **AlwaysEmbed** list

– Font contains an unknown Registry-Ordering-Supplement (ROS). Distiller knows about the following ROS identifiers:

Adobe-Japan1-*
Adobe-Korea1-*
Adobe-GB1-*
Adobe-CNS1-*

where the "*" indicates any integer (as in a wildcard).

When installed for use with the appropriate Language Kit, Acrobat has access to all CIDFontType0 fonts for which Distiller has ROS numbers. Adobe provides Language Kits for the following languages: Japanese, Chinese (2 versions), and Korean.

Type 42/CIDFontType2

If the font is found (rather than being a WidthOnly font) and if license restrictions are satisfied ("Satisfying License Restrictions" on page 32), Distiller embeds a Type 42/CIDFontType2 font if at least ONE of the following conditions is satisfied:

- Font uses a private encoding, regardless of **NeverEmbed**. If such a font cannot be embedded and if **CompatibilityLevel** is defined as 1.2, Distiller terminates the job with an error.
- Font is derived from a CIDFontType2 or Type 42 font and its sibling fonts are embedded, regardless of **NeverEmbed**. (Distiller may split a single TrueType font into multiple Type 42 and CIDFontType2 fonts. If Distiller embeds one of the splintered fonts, it embeds all related fonts.)
- PostScript file uses the font for characters that are NOT included in the Adobe Standard Latin Character Set, regardless of **NeverEmbed**.
- PostScript file uses the font for only characters ONLY in the Standard Latin Character Set. Distiller embeds the font if it is NOT on the **NeverEmbed** list and either **EmbedAll** is *true* or font is in the **AlwaysEmbed** list.

Satisfying License Restrictions

This section describes how Distiller supports font license restrictions specified in the TrueType and CIDFontType0 font formats. Such restrictions are specified in **fsType** integers. "Finding a Font's fsType Integer" explains how Distiller finds an **fsType** integer for a font and what it does if it cannot find one. "Evaluating a Font's fsType Integer" explains how Distiller evaluates the information in an **fsType** integer. "How to Determine Whether a Font Can Be Embedded" on page 34 may also be helpful.

NOTE: As described earlier, Distiller does not assume license restrictions are satisfied for fonts embedded in the PostScript file being processed. Rather, Distiller complies with license restrictions, as described earlier in this section.

Finding a Font's fsType Integer

Distiller finds and responds to a font's **fsType,** as described below:

- For Type 42 or CIDFontType2 fonts, **fsType** is optional. If **fsType** is present in the font, Distiller complies with its specifications, as described in "Evaluating a Font's fsType

Integer," which follows. If the **fsType** integer is not present, Distiller attempts to find it in the original font. If Distiller cannot find the original font or if the original font does not contain an **fsType** integer, Distiller assumes it CAN embed the font.

- For CIDFontType0 fonts, **fsType** integers are required. If **fsType** is present in the font, Distiller complies with its specifications, as described in "Evaluating a Font's fsType Integer," which follows. If **fsType** is not present in the font, Distiller assumes it CANNOT embed the font.

Evaluating a Font's fsType Integer

Distiller evaluates license restrictions represented by flags in **fsType** integer. The meanings of those flags is described in the OpenType specification, available at http://partners.adobe.com/asn/developer/opentype/os2.htm. Distiller evaluates the flags in the order described below, stopping when it comes to a conclusion. As described in the OpenType specification, if multiple embedding flags are set, the least restrictive license takes primacy.

1. If **fsType** is 0, Distiller concludes that license restrictions are satisfied.
2. If editable embedding is allowed (bit-mask 0x0008), Distiller concludes that license restrictions are satisfied. (*Editable embedding* means that the font may be embedded and temporarily loaded on other systems. Although not relevant to Acrobat font use, the definition in the OpenType specification goes on to say that documents containing Editable fonts may be opened for reading and writing.)
3. If preview and print embedding is allowed (bit-mask 0x0004), Distiller concludes that license restrictions are satisfied. (*Preview and print embedding* means that the font may be embedded and temporarily loaded on a remote system. Although not relevant to Acrobat font use, the definition in the OpenType specification goes on to say that documents containing Preview and Print fonts must be opened "read-only;" no edits may be applied to the document.)
4. If embedding is restricted (bit-mask 0x0002), Distiller concludes that embedding is not allowed.
5. If subsetting is not allowed (bit-mask 0x0100) or only bitmap embedding is allowed (bit-mask 0x0200), Distiller concludes that it cannot meet the license restrictions required to embed the font.

   Distiller cannot meet a license restriction that prohibits subsetting because Distiller subsets all fonts except Type 1. Type 1 fonts do not contain **fsType** integers. Neither can Distiller meet a license restriction that requires a font to be bitmap form because it does not have the capacity to convert outline fonts to that form. (You can direct a PostScript driver to convert CIDFontType0 fonts to Type 3 bitmap fonts, but, during that process, the **fsType** integer is lost.)
6. (See following note.) For Type 42 or CIDFontType2 fonts, if the reserved bit is set (bit-mask 0x0001), Distiller concludes that the font cannot be embedded, which is consistent with the Microsoft Web Embedding Fonts Utility (WEFT) interpretation of that same bit. However, that interpretation causes some problems because, despite the specification's definition of bit position-0 (bit-mask 0x0001) as being reserved, some TrueType font

applications mistakenly set the bit, believing it affects embedding restriction as in bit position-1 (bit-mask 0x0002).

NOTE: The check for the reserved bit described in Step 6. exists only in Distiller version 4.0. That check did not exist in prior Distiller versions and was removed in Distiller version 4.05.

How to Determine Whether a Font Can Be Embedded

You can use Distiller or PDFWriter to determine whether a font can be embedded or you can use a font tool to determine the value of a font's **fsType**.

The Distiller UI window **Settings** > **Job Options…** > **Fonts** > **Embedding** > **TrueType** shows only TrueType host fonts that can be embedded.

The PDFWriter font window shows both embeddable and un-embeddable host fonts, using red to differentiate the un-embeddable fonts.

Both Microsoft and Apple provide tools that allow you to see the value of a font's **fsType** integer. The Microsoft tool, called Font Properties Extension, is available from http://www.microsoft.com/typography/free.htm. The Apple tool, called FontSummarizer, is available from http://www.fonts.apple.com/Tools.

The web site http://partners.adobe.com/asn/developer/opentype/os2.htm explains how to interpret the values of a font's **fsType** integer.

Responding When a Font Cannot Be Embedded

If Distiller is unable to embed a font that appears in the **AlwaysEmbed** list, Distiller responds as directed in the Distiller parameter **CannotEmbedFontPolicy** (Table 5.2). If that parameter allows Distiller to complete the job (**CannotEmbedFontPolicy** is OK or Warning), Distiller produces a PDF file that provides information about the un-embeddable font, including, if possible, information that allows Acrobat to create a font substitution.

Tips for Predicting Acrobat Host-Font Use When Original Font Not Embedded

You might want to believe that Acrobat uses the fonts on its host system in place of fonts not embedded in the PDF file, but that is not always the case. Rather, for each category of font (Roman, CJK TrueType, and CID), several conditions must be satisfied before Acrobat uses host fonts. The conditions for each category of fonts are described in the following sections.

Roman Fonts

Acrobat uses the host font if the following conditions are satisfied:

- Acrobat is configured to use local fonts.
- Font is in the Acrobat search path.
- Font is not a base font or in the Arial® family. (Acrobat searches the system for the font and its alias and uses the first of either font it finds. For example, ArialMT is the alias font for

Arial. If Acrobat is processing a font reference for Arial and finds ArialMT before it finds Arial, it uses ArialMT to display and print the font.)

- No other similarly-named fonts exist on the host system.
- For TrueType fonts, Distiller had access to the original font, so it could update the **FontName** field.

NOTE: Remember that, if Distiller cannot find a Roman font, it replaces references to that font with references to Courier. Even if the unfound font is present on the Acrobat system, Acrobat uses Courier, as directed in the font references. (See "Type 1, Type 3, and Type 42 (Usually Roman) Fonts on Windows or Macintosh" on page 25).

CJK TrueType Fonts

Acrobat uses the host font if the following conditions are satisfied:

- Acrobat is configured to use local fonts.
- Font is in the Acrobat search path.
- Font is descended from a TrueType font and uses Glyph ID font encoding. (Glyph ID font encoding is conventional in TrueType fonts. Unlike MacRoman or WinAnsi encoding used in Type 1 fonts, Glyph ID font encoding makes use of additional tables to map from a character code to the glyph definition that produces the associated glyph.)
- Distiller had access to the original font, so it could update the **FontName** field. (This condition is important only if an older PostScript driver was used to produce the Type 42 or CIDFontType 2 font.)

CID Fonts

Acrobat never uses CID fonts installed on the system without first modifying them. More specifically, if Acrobat finds a CID font with the same name as the referenced font, it creates a substitute font, by combining the font installed on the system with the Acrobat subset font. Acrobat creates such substitute fonts because it cannot be certain that the font requested in the PDF file has the same supplement number as the font installed on the system.

Subsetting Fonts

When embedding a font, Distiller (by default) subsets fonts — that is, it includes only the information required to draw glyphs for the characters used in the PDF file being produced. Font subsetting supports the goal of producing the smallest PDF files possible, yet still embed a version of the font that allows characters to be drawn as intended.

Distiller may subset Type 1 Roman fonts embedded in PDF files. (Distiller always subsets other types of embedded fonts and Type 1 fonts that represent non-Roman characters.)

Table 5.4 describes Distiller parameters that influence font subsetting.

TABLE 5.4 Distiller parameters that affect font subsetting

Parameter Name	Description
MaxSubsetPct	A threshold beyond which Distiller embeds the entire font, rather than a subset of the font. Distiller interprets **MaxSubsetPct** as a percentage of the font's glyph definitions used in the document. In other words, Distiller embeds the entire font if the PostScript file uses more than the percentage of glyphs described by **MaxSubsetPct**. (*A glyph definition* is the set of PostScript instructions that, when processed by an interpreter, produce a glyph. Each character represented by the font encoding vector is associated with a unique glyph definition.) For example, assume a PostScript file uses a font that contains 200 glyph definitions and **MaxSubsetPct** is 75%. If the file uses 100 of the 200 glyph definitions, the percent use is only 50% (even if the file uses individual glyph definitions multiple times). In that case, Distiller subsets the font before embedding it. If the file uses 151 glyph definitions of the 200 (more than 75%), the font is not subsetted; rather, Distiller embeds the entire font. **MaxSubsetPct** pertains only to Type 1 Roman fonts, with the following exceptions: Distiller always subsets multiple master fonts (instances of multiple master typefaces). The Distiller UI allows users to modify the value of **MaxSubsetPct** using the % field associated with **Settings > Job Options… > Fonts > Subset embedded fonts below:**
SubsetFonts	If *TRUE*, the Distiller embeds only the information required to draw glyphs for the characters used in the document, rather than embedding the entire font. Pertains only to Type 1 Roman fonts. The Distiller UI allows users to define **SubsetFonts** by selecting **Settings > Job Options… > Fonts > Subset embedded fonts below:**

6 Frequently Asked Questions

This chapter addresses frequently asked questions about how Distiller handles fonts.

Importance of Embedding TrueType Fonts in PostScript Files

Question. How can I get Distiller to work correctly with TrueType fonts.

Answer. Distiller does not directly support TrueType fonts, but it does support their PostScript font counterparts: Type 42 and CIDFontType2. Using PostScript drivers, desktop applications can convert some types of TrueType fonts into Type 42 or CIDFontType2 fonts. Such fonts are embedded in the PostScript files produced by the applications.

You should do the following:

- On Windows systems, set the Distiller printer to download TrueType fonts as Type 42 fonts.
- Direct your desktop application to include all TrueType fonts in the PostScript file it produces. That way, Distiller will be able to use the font spacing for the TrueType font. If a TrueType font is not embedded in the PostScript file, Distiller uses a similarly-named non-TrueType font to determine spacing. If Distiller cannot find a similarly-named font, it replaces references to the TrueType font with references to Courier.
- Set Distiller parameters to embed the TrueType fonts to ensure that Acrobat uses the correct TrueType font to view or print your PDF file. Otherwise, Acrobat may NOT use the exact TrueType font you want, even on a system that has the desired TrueType font. (This situation is true with ANY font not embedded in a PDF file.)
- Distill the PostScript file on the same system used to create the file. That way, Distiller will be able to access the original TrueType fonts to obtain information not included in their PostScript font counterparts. (See "Retrieving Omitted Information for Type 42 and CIDFontType2 Fonts" on page 20.)

No Support for TrueType Fonts But UI Lists Them

Question. Why can Distiller list TrueType fonts in the **Settings > Job Options > Fonts > Embedding > TrueType** window but not be able to find TrueType fonts referenced from a PostScript file? (See previous Q&A.)

Answer. Distiller uses a different mechanism to find fonts for the UI than for processing font references in PostScript files. It uses system utilities to find TrueType fonts for display in the **Settings > Job Options > Fonts** window but uses other internal utilities to find font references in PostScript files.

Selection of Sources for Font Metric Data

Question. I've developed a customized version of Palatino®. The glyphs drawn by that font differ from the glyphs drawn by conventional Palatino, and so the font metric data in that font differs from that in standard Palatino.

If I embed my customized font, Acrobat displays the font without a problem. If I don't embed the font, the substitute font Acrobat uses in place of my customized font looks like standard Palatino, rather than like my customized version. How can I get Acrobat to produce a substitute font that looks like my customized one?

Answer. The easiest, most reliable way of dealing with a customized font is to always embed it, as you did in your first instance. However, if you wish to avoid embedding your customized font (you wish to have Acrobat create a substitute font for your customized font), then the following options exist:

- *Distinctively-named font*. If your font has a distinctive name (unlike other commonly-used fonts), Distiller will be unable to find font metrics for the font in the ATM font substitution database, and so will obtain font metrics from your custom font.
- *Font name same as base-14 font name*. If your font has the same name as a base-14 font or as Arial or ArialMT, Acrobat will use an alias font, rather than trying to create a substitute font. The customized features of your font will not be honored.

  An alias font is a font installed by Acrobat that closely resembles the original font. Acrobat chooses an alias font based on the FontName in the font reference. For example, ArialMT and Helvetica are alias pairs, which means that Acrobat uses them interchangeably. More specifically, if Acrobat is looking for Helvetica but finds ArialMT first, it uses ArialMT in place of Helvetica.
- *Otherwise*. If your font has the same name as another commonly-used font but is neither a base-14 font nor Arial, then you can force Distiller to obtain font metrics from your font file, rather than from the ATM font substitution database. Acrobat will use the customized font metrics to create a substitute font.

  You force Distiller to obtain all font metric data from the font by renaming the font database file (so that Distiller cannot find it). However, as long as the database's name is changed, you must make sure that Distiller can find all fonts used in the PostScript files it processes. Otherwise, Distiller will use a replacement font (by default, Courier) in place of the referenced font.

  On Windows, the ATM font substitution database is in the file `distsadb.dos`, and, on Mac, in `superatm.db`.

See also "Font Substitution (an Acrobat Task)" on page 11 and "Obtaining Font Metric Data" on page 26.

Conversions from MacRoman Encoding to WinAnsi

Question. I noticed that Distiller running on Macintosh converts MacRoman encoding to WinAnsi. Why does it do that?

Answer. WinAnsi encoding defines more characters than MacRoman encoding. Distiller takes advantage of this difference in the following way: If Distiller encounters a character that requires special encoding and if that character comes from a font that uses MacRoman encoding, Distiller determines whether the specially-encoded character can be represented using WinAnsi. If it can be, Distiller uses WinAnsi encoding for the font.

CID-Keyed Fonts and the Embedding Window

Question. I have several CID-keyed fonts on my system that don't appear in the embedding window **Settings** > **Job Options** > **Fonts** > **Embedding**. Why not?

Answer. The fonts you're talking about are license-restricted fonts. Distiller doesn't list in the embedding window fonts that cannot be embedded. Doing so would be misleading.

Glossary

This section defines terms used throughout this document. Many of the following definitions were taken from the *PostScript Language Reference, Third Edition*.

base-14 fonts

Fonts that are guaranteed to be available to Acrobat. Base-14 fonts include:

- Courier (Roman, bold, italic, and bold italic)
- Arial MT (Roman, bold, oblique, and bold oblique)
- Times New Roman® PS MT (Roman, bold, italic, and bold italic)
- ITC Symbol®
- ITC ZapfDingbats®

Earlier version of Distiller and Acrobat used a slightly different set of base-14 fonts. That set included the different faces of Courier, Helvetica (which has since been replaced by Arial MT), Times® (which has since been replaced by Times New Roman), ITC Symbol, and ITC ZapfDingbats.

CMap file

Character Mapping (CMap) files are used with CIDFontType0 fonts to define the mapping from individual character codes to a CID number. A CMap serves the same function for a CIDFontType0 font as an encoding does for a Type 1 font or as a cmap table in a TrueType font.

A single CMap file may be used to provide encoding information for multiple CIDFontType0 fonts, which is more efficient than having the same encoding in multiple font files.

character

An abstract symbol, such as a letter, a ligature (such as “fi”), a number, or a special character (such as “>”). Such an abstract symbol has a representative shape.

See also glyph.

character identifier (CID)

An integer used to access glyph definitions. More specifically, the identifier number for a character in a CID-keyed font. Characters are accessed using this number, instead of using a character name, as with Type 1 Roman fonts.

character set

Character set has the following context-dependent meanings:

- A set of characters whose glyph definitions are conventionally present in a given class of font. An example would be the Adobe Standard Latin character set.
- A formal system for identifying characters and grouping them into a character collection. For CID-keyed fonts (defined below), the CIDs are character identifiers and the Registry-Ordering-Supplement identifies the character collection.

CID-keyed font

CID-keyed fonts provide a convenient and efficient method for defining multi-byte character encodings, base fonts with a large number of glyphs, and composite fonts that use these base fonts and character encodings. Additionally, they provide straightforward methods for creating a rearranged font, which selects glyph definitions from one or more existing fonts by means of a revised encoding. These capabilities provide great flexibility for representing text in writing systems for languages with a large number of characters, such as Chinese, Japanese, and Korean. (*PostScript Language Reference, Third Edition*, pp. 364)

A CID-keyed font is the combination of a CMap with one or more CIDFonts, base fonts, or composite fonts containing glyph definitions.

double-byte font

A font that uses two-bytes to reference each character. Using two bytes allows fonts to reference more characters than they can using a single byte.

font

A program that contains glyph definitions, which are procedures for drawing letters and symbols whose shapes share certain stylistic properties.

font encoding

A method of mapping from character codes to glyph definitions. Fonts use a flexible encoding scheme by which character codes select glyph definitions. The association between character codes and glyph definitions is not part of the glyph definitions themselves; rather, it is described by a separate encoding vector.

font metric data

Quantitative descriptions about the font, such as **Ascent**, **Descent**, and **FontBBox**. **Ascent** is the maximum height above the baseline reached by characters in the font; **Descent** is the maximum depth below the baseline reached by characters in this font; and **FontBBox** is the font's bounding box, which is the smallest rectangle enclosing the shape that would result if all characters in the font were placed with their origins coincident, and then painted. Some font metric data is optional and some required.

This document uses the term *font metric data* to also include other information Acrobat uses to create font substitutions, such as whether the font is serif (Roman fonts only) and what style the font uses (CJK only). Such usage is not entirely accurate; however, it eliminates some wordiness.

Font metrics are described in more detail in *Adobe Font Metrics File Format Specification*.

font reference

An element of a PostScript or PDF file that references the font that should be used to draw the indicated characters. A font reference includes only the font name.

glyph

A specific rendering of a character — the final presentation form of a character. A glyph is the result of an interpreter processing the glyph definition associated with a character. For example, the glyphs A, **A**, and *A* are renderings of the abstract “A” character.

glyph definition

Instructions in a font that, when processed by an interpreter, produce a glyph. Instructions may be written using various conventions, including PostScript, PDF, or TrueType.

Glyph definitions are organized into *fonts*. A font contains glyph definitions for a particular character set; for example, the Helvetica and Times-Roman fonts contain glyph definitions for a set of standard Latin characters. There is one glyph definitions for every character that can be drawn by a font.

See also character and glyph.

host font

Host fonts are any fonts installed on a host system. Such fonts can be installed at various times: with the operating system, with font packages, and with applications. Because host fonts are usually compressed, they occupy far less space than their PostScript font counterparts.

multiple master font

An individual Type 1 font derived from a multiple master typeface. A multiple master font is a specific style variation from a family of related styles. For example, Myriad® Semibold Condensed and Myriad Light Semi Extended are fonts that are derived from a single multiple master typeface.

multiple master typeface

A large family of fonts within a single Type 1 font program. The fonts grouped in a multiple master typeface have related styles. Each multiple master typeface contains all the information necessary to generate hundreds of multiple master fonts, each having specific style variations.

Panose number

A ten-digit number that describes the characteristics of a font. Each digit in the number describes a font characteristic, such as family type, serif style, and weight. Acrobat uses Panose numbers to select suitable fonts for substitution, primarily for CJK fonts.

Panose numbers are described in **http://www.fonts.com/hp/panose/greybook/frame.htm**.

PostScript font

Short for *downloadable PostScript language font program*, a PostScript font uses a a format that can be processed by the PostScript Interpreter.

Registry, Ordering, and Supplement (ROS) values

Entries in a CIDFont and CMap that identify the character collection to be used for a CID-keyed font. For example, Adobe sells a number of Japanese fonts with the ROS value “Adobe-Japan1-2,” where “Adobe” is the Registry; “Japan1” is the ordering; and “2” is the Supplement.

ROS value does not uniquely identify a font; rather, it identifies the CID character collection from which the CID numbers are taken. For example, Ryumin-Light with ROS number “Adobe-Japan1-4” has more characters than Ryumin-Light with ROS value “Adobe-Japan1-2.” (Font name and font version uniquely identify a font.)

The following documents may help your understanding of ROS values:

- *Technical Note #5014, Adobe CMap and CID Font Files Specification, Version 1.0* specifies the ROS number format. That document is available from http://partners.adobe.com/asn/developer/PDFS/TN/5014.CJK_CID.pdf
- *Technical Note #5094, Adobe CJKV Character Collections and CMaps for CID-Keyed Fonts* explains which national and international character sets and encodings are supported by Adobe’s Character Collections. That document is available from http://partners.adobe.com/asn/developer/PDFS/TN/5094.CJK_CID.pdf.

Roman characters

Characters that appear in the Adobe Standard Latin Character Set. (This document uses the terms *Roman* and *Latin* interchangeably.)

Standard Latin Character Set

Adobe standard Latin characters are described in Section E.5 of the *PostScript Language Reference, Third Edition*. (This document uses the terms *Roman* and *Latin* interchangeably.)

substitute font

A font used in place of another font that is not available to Acrobat. A substitute font is derived from another font and closely matches the characteristics of the font it replaces.

Acrobat uses font information in the PDF file to select another font that matches the characteristics of the original font and then to create a new version of that font that it modifies to more closely resemble the original font. For standard Roman fonts, Acrobat selects either the AdobeSerifMM and AdobeSansMM multiple master fonts. For standard Asian fonts, Acrobat selects one of the fonts from the Asian font pack installed with Acrobat.

ToUnicode cmap

A ToUnicode cmap is a table that maps from character codes to Unicode values and allows text strings that use the font to be converted to Unicode values for export to other applications or plug-ins.

The ToUnicode cmap is optional for a PDF file but is essential for non-standard fonts. Without a ToUnicode CMap, text in a PDF document that references non-standard fonts may not be searchable or indexable.

WidthOnly fonts

WidthOnly fonts contain information similar to the font metric data used with Roman fonts. They do not contain glyph descriptors.

Distiller uses WidthOnly fonts for character placement; it cannot embed them in the PDF files it produces.

Acrobat and Distiller are installed with WidthOnly CID-keyed fonts and compatible fonts for all CJK fonts in the Adobe Type Library and for all Macintosh and Windows CJK system fonts. The Macintosh version of Distiller uses the MakeCID utility to produce WidthOnly fonts for any CJK fonts for which WidthOnly fonts don’t already exist.