Title: xlttlib User's Guide **Authors:**	**Word for Windows File: xlttlib.doc** **Word for Windows Version: 8.0** **Figure Files: none**

Revision History

Rev	Revision Description	Date	Author
0.0	Draft	10/15/98	
1.0	Initial Distribution	10/23/98	

Note: This document is for HP Personnel only. Please do not copy and distribute without notification to the original author.

HEWLETT-PACKARD COMPANY MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE SOFTWARE OR TECHNICAL INFORMATION. HEWLETT-PACKARD COMPANY DOES NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE OR TECHNICAL INFORMATION IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS, OR OTHERWISE. YOU ASSUME THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE OR TECHNICAL INFORMATION. The exclusion of implied warranties is not permitted by some jurisdictions. The above exclusion may not apply to you.

IN NO EVENT WILL HEWLETT-PACKARD COMPANY BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION AND THE LIKE) ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE OR TECHNICAL INFORMATION EVEN IF HEWLETT-PACKARD HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to you. Hewlett-Packard liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort including negligence, product liability or otherwise), will be limited to US $50.

Table Of Contents

1. INTRODUCTION 5

2. SCOPE 5

3. DESIGN PRINCIPLES 6

4. TRUETYPE REQUIREMENTS 7

5. USING XLTTLIB 8

5.1 INITIALIZING THE LIBRARY 8
5.2 REGISTERING FONT(S) 8
5.3 USING FONT(S) 9
5.3.1 Downloading the entire font 9
5.3.2 Downloading a font header 10
5.3.3 Downloading sets of characters 10
5.3.4 Calculating the linearly-scaled widths of characters at a given font size 11
5.4 UNREGISTERING FONT(S) 11
5.5 SHUTTING DOWN THE LIBRARY 12

6. ERRORS 12

This page intentionally left blank

1. Introduction

xlttlib is a library of TrueType handling routines that can be used by independent software developers and HP internal test engineers who want to create applications that generate PCL XL data streams. It is meant to be a companion to jetlib, the PCL XL protocol library, also available from HP.

xlttlib understands the specifics of TrueType font files, including DBCS font files, and how to download them as soft fonts to a PCL XL printer.

This document assumes that the reader has at least cursory knowledge of the format of a TrueType font. (See Microsoft documentation: *TrueType 1.0 Font Files* at http://www.microsoft.com/typography/tt/tt.htm, and *Far East TrueType Font Files* also available from Microsoft.)

2. Scope

xlttlib will:

- ✓ calculate the linearly-scaled widths of characters at a given font size for the client
- ✓ download TrueType fonts to a PCL XL printer
- ✓ track TrueType glyphs as they are downloaded to avoid downloading the same sub-glyph more than once for a stroke-based font

xlttlib will not:

- render bitmaps of TrueType characters
- manage the selection of fonts or the setting of font attributes (angle, scale, shear, boldvalue, vertical rotation, vertical substitution, etc.)
- track font headers as they are downloaded to avoid downloading the same font header more than once
- track characters as they are downloaded to avoid downloading the same character more than once
- print characters
- delete fonts from the printer

3. Design Principles

- ✓ **Use a standard programming language**: xlttlib is a set of functions written in the C programming language. C was chosen because most commercial programs are written in C or C++. The routines could easily be combined into a C++ class.
- ✓ **Be portable**: xlttlib is written to be as portable as possible. No assumptions are made about the memory model (endian-ness) or I/O. Callbacks are used to allow the client to implement memory and I/O functionality in any manner that best fits the application. Memory allocations are limited to sizes of less than 64 kilobytes. The library has been compiled and tested successfully on both the Intel and Motorolla platforms.
- ✓ **Be simple**: TrueType handling is complex, but the use of xlttlib should be as simple as possible. xlttlib exports only 9 functions, each with an intuitive purpose. The library consists of one header file (xlttlib.h) and one implementation file (xlttlib.c)
- ✓ **Handle Asian fonts**: xlttlib handles Shift-JIS (Japanese), GB (Simplified Chinese), Big5 (Traditional Chinese), and KSC (Korean) fonts. Vertical rotation and vertical substitution support is included. Default (galley) characters are downloaded automatically for all fonts, and the Japanese 1-byte galley character is also handled appropriately for Shift-JIS-based fonts.
- ✓ **Handle Unicode fonts**: All latin TrueType fonts are Unicode based. In addition, many Asian TrueType fonts are Unicode based. xlttlib can read and download Unicode-based fonts. (Notes: 1. xlttlib cannot distinguish Japnese Unicode-based fonts from other Unicode-based fonts, so the 1-byte Galley character will not be downloaded in this case. 2. Vertical-rotation with Unicode-based fonts has not been explicitly specified by the Unicode Consortium, so xlttlib uses a common-sense approach.)
- ✓ **Handle symbol fonts**: Symbol fonts can be used with xlttlib.
- ✓ **Handle multiple fonts**: xlttlib can manage up to 254 fonts at a time.

4. TrueType Requirements

xlttlib can recognize and process TrueType font files with the following attributes:

- ✓ The TrueType font file *must* conform to the following Microsoft specifications:
  - *TrueType 1.0 Font Files*
  - *Far East TrueType Font Files*
- ✓ The TrueType font file *must* contain valid data for all of the following tables:
  - cmap
  - glyf
  - head
  - hhea
  - hmtx
  - loca
  - maxp
  - OS/2
- ✓ The TrueType font file *may* contain valid data for any of the following tables:
  - cvt
  - fpgm
  - mort (used to identify vertical substitute glyphs)
  - prep
  - vhea (vertical metrics header)
  - vmtx (vertical metrics)
- ✓ The TrueType font's cmap table *must* contain at least one Platform ID 3 (Microsoft) subtable.
- ✓ The first Platform ID 3 (Microsoft) subtable in the TrueType font's cmap table *must* have one of the following specific IDs:
  - 0 Symbol
  - 1 Unicode
  - 2 Japanese (Shift-JIS)
  - 3 Simplified Chinese (GB)
  - 4 Traditional Chinese (Big5)
  - 5 Korean (KSC, Wansung)
- ✓ The first Platform ID 3 subtable in the TrueType font's cmap table *must* be format 4 (segmented mapping to delta values).
- ✓ The first Platform ID 3 subtable in the TrueType font's cmap table *must not* map to values greater than the maxp table's numGlyphs – 1.

Note: xlttlib does not directly process TrueType collection files (TTCs).

5. Using xlttlib

Using xlttlib consists of the following main procedures:

1. **Initializing the library**: The initialization process passes memory management routines to the library and lets the library initialize global variables.
2. **Registering font(s)**: Registering a font passes file access routines for the font file to the library, allows the library to do a quick verification of the font file, and lets the library cache some of the font file information in its font database. The library will return a font handle to the client for future reference and will tell the client what mapping must be used with the font.
3. **Using font(s)**: The client program can have the library download the font in its entirety or in pieces. The client can also have the library calculate character widths.
4. **Unregistering font(s)**: When the client is done using a registered font, it should always unregister the font to allow the library to free memory used to manage the font.
5. **Shutting down the library**: The shutdown procedure allows the library to clean up after itself.

5.1 Initializing the library

Before the library can be initialized, the client program must implement two memory management routines (**memAlloc** and **memFree**) and place references to them in the following structure:

```
typedef struct xlttlib_MemAccessTag
{
  HP_pUByte (*memAlloc)(HP_UInt16 byteCount);
  void      (*memFree) (HP_pUByte dataPtr);
}xlttlib_MemAccessType;
```

Initialization is a simple process, passing this structure to the library through the following interface:

```
boolean xlttlib_Initialize(xlttlib_MemAccessType memAccess);
```

5.2 Registering font(s)

For each font to be used, the client program must implement two file access routines (**seek** and **read**) and place references to them in the following structure:

```
typedef struct xlttlib_FontAccessTag
{
  boolean (*seek)(void *clientDataPtr, HP_UInt32 offsetFromTop);
  boolean (*read)(void *clientDataPtr, HP_UInt16 byteCount,
    HP_pUByte buffer);
  void    *clientDataPtr;
}xlttlib_FontAccessType;
```

- ✓ The implementation of the **seek** routine *must* set the current location to **offsetFromTop** bytes from the top of the font file.
- ✓ The implementation of the **read** routine *must* read from the current location.

- ✓ The **clientDataPtr** *may* reference any data that the client might need to be passed to the **seek** and **read** routines at run-time.

The client must also declare a structure of the following type to receive information about the registered font from the library:

```
typedef struct xlttlib_FontInfoTag
{
  xlttlib_FontHandleType fontHandle;
  xlttlib_MappingType    mapping;
}xlttlib_FontInfoType;
```

To register a font, the client passes these structures through the following interface:

```
boolean xlttlib_RegisterFont(xlttlib_FontAccessType  fontAccess,
                             xlttlib_FontInfoType   *fontInfoPtr);
```

Upon return, the **fontHandle** in the **fontInfo** structure will be used by the client to refer to the font in future calls to the library, while the **mapping** value will inform the client what type of font this is. The **mapping** value will be one one of the following, determined by the font's cmap table:

```
xlttlib_ESymbolMapping
xlttlib_EUnicodeMapping
xlttlib_EJapaneseMapping
xlttlib_ESimpChineseMapping
xlttlib_ETradChineseMapping
xlttlib_EKoreanWansungMapping
```

5.3 Using font(s)

A client can use the library to perform several useful actions on a regestered font:

1. Download the entire font
2. Download a font header
3. Download sets of characters
4. Calculate the linearly-scaled widths of characters at a given font size

5.3.1 Downloading the entire font

In certain cases, it makes sense to download an entire font – the font header and all of the characters in the font. The library exports a function specifically for this purpose:

```
boolean xlttlib_DownloadEntireFont(HP_StreamHandleType    pStream,
                                   xlttlib_FontHandleType fontHandle,
                                   HP_pUByte              fontName);
```

- ✓ The client *must* refer to a registered font by supplying the **fontHandle**.
- ✓ The client *must* supply a jetlib stream pointer (**pStream**) and a null-terminated **fontName** string.

5.3.2 Downloading a font header

In typical cases, only a subset of a font's characters are needed to print a document. In these cases, it is appropriate to download only the font header, to be followed later by downloading sets of characters.

The library exports a function to download just the font header:

```
boolean xlttlib_DownloadHeader(HP_StreamHandleType    pStream,
                               xlttlib_FontHandleType fontHandle,
                               HP_pUByte              fontName);
```

- ✓ The client *must* refer to a registered font by supplying the **fontHandle**.
- ✓ The client *must* supply a jetlib stream pointer (**pStream**) and a null-terminated **fontName** string.

Note: In addition to downloading the font header, this function will also download galley characters.

5.3.3 Downloading sets of characters

In typical cases, only a subset of a font's characters are needed to print a document. In these cases, it is appropriate to download only the font header, to be followed later by downloading sets of characters.

The library exports a function to download sets of characters:

```
boolean xlttlib_DownloadCharacters(HP_StreamHandleType    pStream,
                                   xlttlib_FontHandleType fontHandle,
                                   HP_pUByte              fontName,
                                   HP_UInt16              startCode,
                                   HP_UInt16              endCode);
```

- ✓ The client *must* refer to a registered font by supplying the **fontHandle**.
- ✓ The client *must* supply a jetlib stream pointer (**pStream**) and a null-terminated **fontName** string.
- ✓ **startCode** and **endCode** *may* be the same, but **endCode** *must never* be less than **startCode**.

5.3.4 Calculating the linearly-scaled widths of characters at a given font size

PCL-XL expects an application to send widths for each character to be printed. The following function can be used to get scaled character widths for a range of characters from the specified font at a specified charSize (user units):

```
boolean xlttlib_GetLinearScaledCharWidths(
    xlttlib_FontHandleType  fontHandle,
    HP_Real32               charSize,
    boolean                 verticalRotation,
    boolean                 verticalSubstitution,
    HP_UInt16               startCode,
    HP_UInt16               endCode,
    HP_pSInt16              widthsPtr);
```

- ✓ The client *must* refer to a registered font by supplying the **fontHandle**.
- ✓ **startCode** and **endCode** *may* be the same, but **endCode** *must not* be less than **startCode**.

Note: Widths will be linearly-scaled. That is, if the font size increases by a factor of 2 so will each character's width. The **widthsPtr** must be allocated before this function is called and must be at least large enough to store ((endCode-startCode)+1) entries.

Note: For proportional DBCS fonts, the character width may be different if the font is vertical-rotated, so be sure to set the **verticalRotated** flag whenever the characters will be printed in a vertical rotated fashion. Also, vertical substitute glyphs may have different metrics from their horizontal counterparts, so be sure to set the **verticalSubstitution** flag if the characters will be printed in a vertical substituted fashion.

5.4 Unregistering font(s)

When a client has finished using a font, it should allow the library to free any memory associated with the font by unregistering it with the following function:

```
boolean xlttlib_UnregisterFont(xlttlib_FontHandleType fontHandle);
```

- ✓ The client *must* refer to a registered font by supplying the **fontHandle**.

Note: Calling this function *will not* delete the font from the printer. To delete the font from the printer, the client must call the jetlib function, HP_RemoveFont_1.

5.5 Shutting down the library

When a client has finished using the library, it should allow the library to do any final clean-up by calling the following function:

```
boolean xlttlib_ShutDown(void);
```

Note: Calling this function will effectively unregister all registered fonts.

Note: Calling this function *will not* delete registered fonts from the printer. To delete a font from the printer, the client must call the jetlib function, HP_RemoveFont_1.

6. Errors

If any of the library's functions returns failure (FALSE), the following routine can be called to determine the cause:

```
extern xlttlib_ErrorType xlttlib_GetError(void);
```

It will return one of the following error codes:

Error Code	Possible Cause(s)
`xlttlib_EUnknown`	Unknown
`xlttlib_EInvalidCallSequence`	An attempt to use the library before it is initialized
`xlttlib_EInvalidFontAccess`	The seek or read function pointer in the fontAccess structure is NULL
`xlttlib_EOutOfFontHandles`	Attempt to register more than 254 fonts
`xlttlib_EInvalidCharRange`	endCode is less than startCode
`xlttlib_ESeekError`	A call to fontAccess.seek failed
`xlttlib_EMemAllocError`	A call to memAccess.memAlloc failed
`xlttlib_EInvalidMemAccess`	The memAlloc or memFree function pointer in the memAccess structure is NULL
`xlttlib_EInvalidFontInfoPointer`	The fontInfoPtr parameter is NULL
`xlttlib_EInvalidFont`	The font does not conform to the TrueType requirements
`xlttlib_EInvalidFontHandle`	The specified fontHandle does not refer to a registered font
`xlttlib_EInvalidWidthsPtr`	The widthsPtr parameter is NULL
`xlttlib_EReadError`	A call to fontAccess.read failed