Title: Downloading Soft Fonts in PCL XL 2.0 **Authors:**	**Word for Windows File: Sfnt-XL20.doc** **Word for Windows Version: 8.0** **Figure Files: none**

Revision History

Rev	Revision Description	Date	Author
0.0	Draft	09/08/98	
1.0	Initial Distribution	10/07/98	

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

This page intentionally left blank

1. Introduction

How to download soft fonts (temporary fonts downloaded within a job) is one of the most frequently asked questions about PCL XL. Although **Appendix L** of the *PCL XL Feature Reference Protocol Class 2.0* provides a description of the font formats and download procedure, it lacks concrete examples for downloading Bitmap and TrueType soft fonts. This document will explain the details of the font downloading procedure for both bitmap and TrueType fonts.

This document assumes that the reader is familiar with PCL XL, its syntax and basic job structure. The reader should also be familiar with the PCL XL assembly language and jetasm assembler.

This document also assumes that the reader is familiar with the concept of fonts and font technologies, particularly bitmap and TrueType. The reader should also be familiar with the TrueType font file structure and terminology. (See Microsoft documentation: *TrueType 1.0 Font Files* at http://www.microsoft.com/typography/tt/tt.htm, and *Far East TrueType Font Files* for more information.)

2. Organization

Downloading a font consists of two basic procedures:

1. Downloading a Font Header
2. Downloading Characters

Section 3 will concentrate on the specifics of downloading a font header. Section 4 will concentrate on the specifics of downloading characters.

3. Downloading Font Headers

This section will describe the process of downloading a font header.

Font headers supply PCL XL and LaserJet font rendering technologies with information that is global to the font being downloaded (i.e. not specific to any one character).

3.1 Operators

The following operators are used to download a PCL XL font header:

BeginFontHeader

This operator signals the beginning of the font header download procedure. It provides information about the font that is used to prepare PCL XL's font database: **FontFormat**, **FontName**. This operator ***must*** be followed by a ReadFontHeader operator.

ReadFontHeader

This operator allows *arbitrary chunks* of the font header data to be downloaded. This operator ***must*** be followed by either another ReadFontHeader operator or an EndFontHeader operator.

Note: All font data must be sent to PCL XL 2.0 ***Most-Significant-Bit-First***.

Note: This operator has been misunderstood by some early users because it can only be used to download a maximum of 64k bytes of data at a time. If the data to be downloaded is larger than 64k bytes, simply split the data into smaller chunks and use multiple ReadFontHeader operators in succession.

EndFontHeader

This operator signals the end of the font header download procedure. Although it has no attributes, this operator provides easy fault recovery and fits well into the structure of the PCL XL language.

3.2 The PCL XL 2.0 Font Header Download Procedure

For both bitmap and TrueType soft fonts, the following general font header download procedure is recommended:

1. Decide the **Font Name**, **Mapping** and **Number of Characters** to be used for the font to be downloaded.
2. Begin the font header download sequence by executing the BeginFontHeader operator, supplying the **Font Name** and Font**Format**(0) attributes.
3. Construct a PCL XL **Format 0 Font Header,** filling in the **Format**(0), **Orientation, Mapping**, **Font Scaling Technology**, **Variety**(0), and **Number of Characters** fields.
4. Send the PCL XL **Format 0 Font Header** by executing a ReadFontHeader operator**.**
5. Construct and send *all* **Required Font Data Segments** (except the **NULL** segment) by executing additional ReadFontHeader operators.
6. Construct and send *any desired* **Optional Font Data Segments** by executing additional ReadFontHeader operators.
7. Construct and send the **NULL Font Data Segment** with one final ReadFontHeader operator.
8. Execute the EndFontHeader operator.

When this procedure is completed successfully, PCL XL will add the downloaded information to its font database. Any graphics state operator or drawing operator may now be executed. The font may even be selected with the **SetFont** operator. However, no characters from this font will print with the **Text** or **TextPath** operators until the characters have been downloaded.

3.3 The PCL XL 2.0 Format 0 Font Header

The first piece of downloaded font data that must be supplied to PCL XL 2.0 is a **Format 0 Font Header**:

Byte	**15(MSB) 8**	**7 (LSB)0**	**Byte**
0	Format = 0	Orientation	1
2	Mapping		3
4	Font Scaling Technology	Variety = 0	5
6	Number Of Characters		7

3.3.1 Definitions of Fields

The fields in the PCL XL 2.0 Format 0 Font Header are defined as follows:

Format *(UBYTE)*: Download format flag. (Must be 0 for PCL XL 2.0)

Orientation *(UBYTE):* Specifies the orientation of characters in a bitmap font. The following values are supported by PCL XL 2.0:

- 0 Portrait
- 1 Landscape
- 2 Reverse Portrait
- 3 Reverse landscape

For TrueType fonts, this field *must* be set to 0.

Font Scaling Technology *(UBYTE):* This field specifies the font technology to be used to scale the characters. The following values are supported by PCL XL 2.0:

- 1 TrueType
- 254 Bitmap

PCL XL 2.0 does not support Intellifont rendering in the printer. Intellifont characters should be rendered on the host and downloaded as bitmap characters.

Variety *(UBYTE):* This field must be 0 for PCL XL 2.0.

Mapping *(UINT16):* Specifies the native mapping for the font in the PCL-style symbolset format. The following are *some* of the values supported by PCL XL 2.0 (See **Appendix O** of the *PCL XL Feature Reference Protocol Class 2.0* for a more complete list):

- 590 Unicode
- 629 Win 3.1 Latin 1
- 619 Win3.1J-DBCS
- 579 GB2312-1980
- 596 Big5
- 616 KS C5601-1987

In most cases, the value that is used in this field and the **SymbolSet** attribute that will be used later to select the font with the SetFont operator must be the same.

Number of Characters *(UINT16):* The number of characters that will be downloaded for this font.

3.3.2 Bitmap Example in PCL XL Assembly

```
uint16 8 FontHeaderLength
ReadFontHeader
dataLength 8 hex_raw*[
      00    // Format Must Be 0
      00    // Orientation = 0 = Portrait
      01 15 // Mapping = 277 = Roman 8
      fe    // Font Scaling Technology Must Be 254 for Bitmap
      00    // Variety Must Be 0
      04 00 // Number Of Characters = 1024
]
```

3.3.3 TrueType Example in PCL XL Assembly

```
uint16 8 FontHeaderLength
ReadFontHeader
dataLength 8 hex_raw*[
      00    // Format Must Be 0
      00    // Orientation Must Be 0 = Portrait
      01 15 // Mapping = 277 = Roman 8
      01    // Font Scaling Technology Must Be 1 for TrueType
      00    // Variety Must Be 0
      02 20 // Number Of Characters = 640
]
```

3.4 Segmented Font Data

Segmented font data must immediately follow the PCL XL 2.0 Format 0 Font Header. Each segment contains three parts: **Segment Identifier (UINT16)**, **Segment Size (UINT32)** and **Segment Data.**

The Segmented Font Data portion of the downloaded soft font header will be arranged in the following manner:

Byte	**15(MSB) 8**	**7 (LSB)0**	**Byte**
0	1st Segment ID		1
2	1st Segment Size		3
4			5
6	1st Segment Data		7
...			...
6 + 1st seg size	2nd Segment ID		
…	2nd Segment Size		…
…	2nd Segment Data		…
…	More... ...Segments		…
	NULL Segment ID = 0xFFFF		
	NULL Segment Size = 0		

3.4.1 Supported Segments for Bitmap Fonts

PCL XL 2.0 supports the following segments for downloaded bitmap soft fonts:

- Required:
  - Bitmap Resolution Segment
- Optional:
  - Vendor Information Segment
- Required Terminal (must be sent last)
  - NULL Segment

3.4.2 Supported Segments for TrueType Fonts

PCL XL 2.0 supports the following segments for downloaded TrueType soft fonts:

- Required:
  - Global TrueType Segment
- Optional:
  - Galley Character Segment
  - Vertical Exclude Segment
  - Vertical Rotation Segment
  - Vertical Transformation Segment
  - Vendor Information Segment
- Required Terminal (must be sent last)
  - NULL Segment

3.5 Font Data Segments

This section will present uses for and detailed descriptions of all of the segments listed in section 3.4 in alphabetical order.

3.5.1 The Bitmap Resolution Segment (Bitmap Fonts Only)

The BR Segment tells PCL XL 2.0 the rendered resolution of the characters to be downloaded.

3.5.1.1 Format

The BR Segment has the following format:

Byte	15(MSB) 8	7 (LSB)0	Byte
0	"B"	"R"	1
2	Segment Size = 4		3
4			5
6	X Resolution		7
8	Y Resolution		9

Supported values for the resolution fields in PCL XL 2.0 are:

- 300
- 600
- 1200

Note: For PCL XL 2.0, the X Resolution and Y Resolution values must be equal.

3.5.1.2 BR Segment PCL XL Assembly Example

In the following example, PCL XL is told that all characters to be downloaded for this font will be rendered at 600 x 600 dpi.

```
uint16 10 FontHeaderLength
ReadFontHeader
dataLength 10 hex_raw* [
      42 52       // Segment ID Must Be "BR"
      00 00 00 04 // Segment Size Must Be 4
      02 58       // X Resolution = 600
      02 58       // Y Resolution = 600
]
```

3.5.2 The Galley Character Segment (TrueType Fonts Only)

By default, PCL XL 2.0 will print a blank space when a character that does not exist within the current font is requested.

The GC Segment may be used to tell PCL XL 2.0 which galley (substitute) character(s) to print when a character that does not exist within the current font is requested. One galley character is designated as a default. In addition, multiple galley characters can be defined, one per specified region. (Specifying a galley character of 0xFFFF causes the TrueType glyph 0 to be printed if it exists.)

Note: The galley characters must be downloaded before they can be printed. If a galley character does not exist, a blank space will be printed in its place.

3.5.2.1 Format

The GC Segment has the following format:

Byte	15(MSB) 8	7 (LSB)0	Byte
0	"G"	"C"	1
2	Segment Size = (6*N)+6		3
4			5
6	Format = 0		7
8	Default Galley Character		9
10	Number Of Regions = N		11
12	Region 1 Upper Left Character Code		13
14	Region 1 Lower Right Character Code		15
16	Region 1 Galley Character		17
…	More… …Regions		…
(6*N)+6	Region N Upper Left Character Code		(6*N)+7
(6*N)+8	Region N Lower Right Character Code		(6*N)+9
(6*N)+10	Region N Galley Character		(6*N)+11

3.5.2.2 GC Segment PCL XL Assembly Example

In the following example, TrueType glyph 0 is designated as the default galley character and character 0x00A5 is designated as the galley character for the range [0..255]. If a non-existent character in the range [0..255] is requested, character 0x00A5 will be printed. If any other non-existent character is requested, TrueType glyph 0 will be printed. (This example follows the Shift-JIS rules for galley characters as implemented by Microsoft Windows.)

```
uint16 18 FontHeaderLength
ReadFontHeader
dataLength 18 hex_raw* [
      47 43       // Segment ID Must Be “GC”
      00 00 00 0c // Segment Size = 12
      00 00       // Format Must Be 0
      ff ff       // Use glyph 0 as the default
      00 01       // Number of Regions = 1
      00 00       // Range 1 Upper Left = 0
      00 ff       // Range 1 Lower Right = 255
      00 a5       // Substitute character 0x00A5 for non-existent characters
                  // in Range 1
]
```

3.5.3 The Global TrueType Segment (TrueType Fonts Only)

The GT Segment is by far the most complicated font data segment. Essentially, it is a container for tables that are found within the TrueType font file of the font being downloaded.

The following tables are recognized by PCL XL 2.0:

- Required
  - head
  - maxp
  - gdir (Empty table. This is a placeholder for the table that the printer will allocate to store downloaded characters.)
- Optional (Send them if they exist in the TrueType font file)
  - cvt
  - fpgm
  - prep
- Special Cases
  - hhea (Required if using Character Class 0)
  - hmtx (Required if using Character Class 0)
  - vhea (Required if doing vertical-rotated writing, using Character Class 0 or 1, and the table exists in the TrueType font file)
  - vmtx (Required if doing vertical-rotated writing, using Character Class 0 or 1, and the table exists in the TrueType font file)

  (See section 4 for more information about character classes.)

3.5.3.1 Format

- Like all font data segments, the GT Segment begins with a Segment Identifier and a Segment Size:

Byte	15(MSB) 8	7 (LSB)0	Byte
0	"G"	"T"	1
2	Segment Size = X		3
4			5

- The segment size is followed by a **TrueType Soft Font Directory Header**:

Byte	15(MSB) 8	7 (LSB)0	Byte
0	SFNT Version = 0x00010000		1
2			3
4	Number Of Tables = N		5
6	Search Range		7
8	Entry Selector		9
10	Range Shift		11

Where:

- Search Range = (maximum power of 2 <= N) * 16
- Entry Selector = Log_2(maximum power of 2 <= N)
- Range Shift = (N * 16) - Search Range

- The next piece of data to send is the **Table Directory**:

Byte	15(MSB) 8	7 (LSB)0	Byte
0	1st Table Tag		1
2			3
4	1st Table CheckSum		5
6			7
8	1st Table Offset		9
10			11
12	1st Table Size		13
14			15
…	More Directory… …Entries		…
(N*16)-16	Nth Table Tag		(N*16)-15
(N*16)-14			(N*16)-13
(N*16)-12	Nth Table Checksum		(N*16)-11
(N*16)-10			(N*16)-9
(N*16)-8	Nth Table Offset		(N*16)-7
(N*16)-6			(N*16)-5
(N*16)-4	Nth Table Size		(N*16)-3
(N*16)-2			(N*16)-1

Note: The entries in the Table Directory must be arranged in ascending order by tag:

Note: The checksum values are ignored by PCL XL 2.0.

Note: The offset values are measured from the top of the TrueType Soft Font Directory Header to the start of the table data in the PCL XL embedded data stream.

Note: Each table size must be an even multiple of 4 bytes.

- The contents of the tables indicated by the table directory, padded so that each table's size is an even multiple of 4 bytes, must immediately follow the table directory at the respective offsets specified in the table directory. For brevity, no examples are given here.

3.5.3.2 GT Segment PCL XL Assembly Examples

- The following assembly code downloads a GT Segment Identifier and Segment Size:

```
uint16 6 FontHeaderLength
ReadFontHeader
dataLengthByte 6
hex_raw* [
      47 54          // Segment ID Must Be “GT”
      00 00 00 f4    // Segment Size = 244
]
```

- The following assembly code downloads a TrueType Soft Font Directory Header:

```
uint16 12 FontHeaderLength
ReadFontHeader
dataLengthByte 12
hex_raw* [
      00 01 00 00    // SFNT Version Must Be 0x00010000
      00 06          // Number Of Tables = 6
      00 40          // Search Range = 64
      00 02          // Entry Selector = 2
      00 20          // Range Shift = 32
]
```

- The following assembly example shows a Table Directory being downloaded:

```
uint16 96 FontHeaderLength
ReadFontHeader
dataLengthByte 96
hex_raw* [
      63 76 74 20 // cvt tag
      00 00 00 00 // cvt checksum = 0 (Ignored)
      00 00 00 6c // cvt offset = 108
      00 00 00 08 // cvt length = 8
      66 70 67 6d // fpgm tag
      00 00 00 00 // fpgm checksum = 0 (Ignored)
      00 00 00 74 // fpgm offset = 116
      00 00 00 14 // fpgm length = 20
      67 64 69 72 // gdir tag
      00 00 00 00 // gdir checksum = 0 (Ignored)
      00 00 00 00 // gdir offset Must Be 0
      00 00 00 00 // gdir length Must Be 0
      68 65 61 64 // head tag
      00 00 00 00 // head checksum = 0 (Ignored)
      00 00 00 88 // head offset = 136
      00 00 00 38 // head length = 56
      6d 61 78 70 // maxp tag
      00 00 00 00 // maxp checksum = 0 (Ignored)
      00 00 00 c0 // maxp offset = 192
      00 00 00 20 // maxp length = 32
      70 72 65 70 // prep tag
      00 00 00 00 // prep checksum = 0 (Ignored)
      00 00 00 e0 // prep offset = 224
      00 00 00 14 // prep length = 20
]
```

3.5.4 The NULL Segment

The NULL Segment tells PCL XL 2.0 that the font header download is complete. It must be followed immediately by an EndFontHeader operator.

3.5.4.1 Format

The NULL Segment has the following format:

Byte	15(MSB) 8	7 (LSB)0	Byte
0	0xFFFF		1
2	Segment Size = 0		3
4			5

3.5.4.2 NULL Segment PCL XL Assembly Example

```
uint16 6 FontHeaderLength
ReadFontHeader
dataLength 6 hex_raw* [
      ff ff       // Segment ID Must Be 0xFFFF
      00 00 00 00 // Segment Size Must Be 0
]

EndFontHeader     // Must Follow Immediately
```

3.5.5 The Vertical Exclude Segment (Vertical TrueType Fonts Only)

The VE Segment identifies the ranges of characters that *should **not** be rotated* when performing vertical-rotated writing.

If the segment is not present, only the characters with advance widths equal to UnitsPerEm will be rotated. (See the SetCharAttributes and SetCharSubMode operators for more information on vertical-rotated writing.)

3.5.5.1 Format

The VE Segment has the following format:

Byte	15(MSB) 8	7 (LSB)0	Byte
0	"V"	"E"	1
2	Segment Size = (4*N)+2		3
4			5
6	Format = 0	NumRanges = N	7
8	Range 1 FirstCode		9
10	Range 1 LastCode		11
…			
(4*N)+4	Range N FirstCode		(4*N)+5
(4*N)+6	Range N LastCode		(2*N)+7

3.5.5.2 VE Segment PCL XL Assembly Example

In the following example, one range of characters [0..255] is excluded from vertical rotation. When this font is used, if **eVerticalRotated WritingMode** attribute is set by the **SetCharAttributes** operator, none of the characters in the range [0..255] will be rotated, but all other characters will be rotated 90 degrees counter-clockwise.

```
uint16 12 FontHeaderLength
ReadFontHeader
dataLength 12 hex_raw* [
      56 45       // Segment ID must be "VE"
      00 00 00 06 // Segment Size = 6
      00          // Format Must Be 0
      01          // NumRanges = 1
      00 00       // Range 1 First Code = 0
      00 ff       // Range 1 Last Code = 255
]
```

3.5.6 The Vendor Information Segment

The VI Segment is an all-purpose segment that can be used by ISV's to store any extraneous font information that may be important to them (copyright, etc.).

The contents of the segment will be ignored by PCL XL 2.0.

3.5.6.1 Format

The VI Segment has the following format:

Byte	15(MSB) 8	7 (LSB)0	Byte
0	"V"	"I"	1
2	Segment Size = N		3
4			5
6	Put any data…		7
…			…
N+4	…here		N+5

3.5.6.2 VI Segment PCL XL Assembly Example

In the following example, the string "Hello" is included in the VI segment. This string will be ignored by PCL XL 2.0.

```
uint16 11 FontHeaderLength
ReadFontHeader
dataLength 11 hex_raw* [
      56 49             // Segment ID must be "VI"
      00 00 00 05       // Segment Size = 5
      48 65 6c 6c 6f    // The string "Hello" (will be ignored)
]
```

3.5.7 The Vertical Rotation Segment (Vertical TrueType Fonts Only)

The VR Segment provides PCL XL with the sTypoDescender value from the TrueType font file's OS/2 table. This value is used by the printer for fine-tuning vertical rotation.

3.5.7.1 Format

The VR Segment has the following format:

Byte	15(MSB) 8	7 (LSB)0	Byte
0	"V"	"R"	1
2	Segment Size = 4		3
4			5
6	Format = 0		7
8	sTypoDescender		9

3.5.7.2 VR Segment PCL XL Assembly Example

The following example tells PCL XL 2.0 that this TrueType font's OS/2 table contains the value –204 for sTypoDescender.

```
uint16 10 FontHeaderLength
ReadFontHeader
dataLength 10 hex_raw* [
      56 49       // Segment ID Must Be "VR"
      00 00 00 02 // Segment Size Must Be 4
      00 00       // Format Must Be 0
      ff 34       // sTypoDescender = -204
]
```

3.5.8 The Vertical Transformation Segment (Vertical TrueType Fonts with Substitutes Only)

The VT Segment serves the same purpose as the TrueType 'mort' table. It provides a mechanism for the substitution of glyphs for use in vertical writing. Therefore, it should only be downloaded for TrueType fonts that: 1) will be used for vertical-rotated writing; and 2) enumerate vertical substitute glyphs through either a **mort** table or a **GSUB** table.

(See the SetCharSubMode and SetCharAttributes operators for more information on vertical-rotated writing.)

3.5.8.1 Format

The VT Segment has the following format:

Byte	15(MSB) 8	7 (LSB)0	Byte
0	"V"	"T"	1
2	Segment Size = (4*N)+4		3
4			5
6	Horizontal Glyph ID 1		7
8	Vertical Substitute Glyph ID 1		9
…	More… …Substitutes		…
(4*N)+2	Horizontal Glyph ID N		(4*N)+3
(4*N)+4	Vertical Substitute Glyph ID N		(4*N)+5
(4*N)+6	End Of Table Marker 1 = 0xFFFF		(4*N)+7
(4*N)+8	End Of Table Marker 2 = 0xFFFF		(4*N)+9

3.5.8.2 VT Segment PCL XL Assembly Example

In the following example, two substitutes are defined for vertical writing. When the font is used, if the **eVerticalSubstitution CharSubMode** is set using the **SetCharSubMode** operator, glyph 11 will be printed in place of glyph 10 and glyph 13 will be printed in place of glyph 12.

```
uint16 18 FontHeaderLength
ReadFontHeader
dataLength 18 hex_raw* [
      56 54       // Segment ID must be “VT”
      00 00 00 0c // Segment Size = 12
      00 0a       // Horiz Glyph ID 1 = 10
      00 0b       // Verti Glyph ID 1 = 11
      00 0c       // Horiz Glyph ID 2 = 12
      00 0d       // Verti Glyph ID 2 = 13
      ff ff       // Sentinel 1 = 0xFFFF
      ff ff       // Sentinel 2 = 0xFFFF
]
```

4.0 Downloading Characters

This section will describe the process of downloading characters.

4.1 Operators

The following operators are used to download characters in PCL XL:

BeginChar

This operator signals the beginning of the character downloading procedure. It indicates the font that the characters are intended for with the **FontName** attribute. This operator ***must*** be followed by a ReadChar operator.

ReadChar

This operator is used to download the data for exactly one bitmap character or TrueType glyph. The **CharCode** attribute tells PCL XL which character to associate the data with, while **CharDataSize** tells PCL XL how much data to read from the embedded data stream. This operator ***must*** be followed by either another ReadChar operator or an EndChar operator.

Note: All font data must be sent to PCL XL 2.0 ***Most-Significant-Bit-First***.

EndChar

This operator signals the end of the character download procedure. Although it has no attributes, this operator provides easy fault recovery and fits well into the structure of the PCL XL language.

4.2 Character Downloading Procedure

For both bitmap and TrueType soft fonts, the following general character download procedure is recommended:

1. Be sure that the font header has already been downloaded.
2. Before printing a string of text with this font, decide *which characters* must be downloaded and recall the *name* of the font for which they must be downloaded.
3. Begin the character downloading process by executing the BeginChar operator, supplying the **FontName** attribute.
4. For each character to be downloaded, send the character data by executing a ReadChar operator, supplying the **CharCode** and **CharDataSize** attributes. (TrueType only: If the TrueType character is a composite of glyph pieces, send the data for each glyph piece by executing a ReadChar operator.)
5. Execute the EndChar operator to signal the end of the character downloading procedure.
6. You may now print the string with the Text or TextPath operators.
7. For each string to be printed with this font, repeat steps 2 through 6.

4.3 Downloading Bitmap Characters: Format 0

4.3.1 Format 0

PCL XL 2.0's downloaded bitmap characters have the following format:

Byte	15(MSB)	8	7	(LSB)0	Byte
0	Format = 0		Class = 0		1
2	Left Offset				3
4	Top Offset				5
6	Character Width = W				7
8	Character Height = H				9
10	Bitmap Character Data				11
...					…
((W*H)/8)+8	…Goes Here				((W*H)/8)+9

4.3.2 Bitmap Character Downloading PCL XL Assembly Example

The following PCL XL assembly illustrates how to download a bitmap character:

```
uint16 76 CharCode
uint32 306 CharDataSize
ReadChar
dataLength 306
hex_raw* [
      00    // Format Must Be 0
      00    // Class Must Be 0
      00 0d // Left Offset = 13
      00 4a // Top Offset = 74
      00 20 // Character Width = 32
      00 4a // Character Height = 74
      00 00 f8 00 00 01 fc 00 00 03 fe 00 00 07 ff 00
      00 07 ff 00 00 07 ff 00 00 07 ff 00 00 07 ff 00
      00 03 fe 00 00 01 fc 00 00 00 f8 00 00 00 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      ff ff f8 00 ff ff f8 00 ff ff f8 00 ff ff f8 00
      ff ff f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00 00 00 f8 00 00 00 f8 00
      00 00 f8 00 00 00 f8 00
]
```

4.4 Downloading TrueType Glyphs: Format 1

4.4.1 Format 1

The PCL XL 2.0 TrueType character format, **Format 1**, has three class variations. Each class is intended for use in a specific circumstance, to enhance memory usage and performance. Therefore, each class should be used based on these specific circumstances.

4.4.1.1 Class 0

Class 0 should be used when a majority of the glyphs in the font will be downloaded. It assumes that the hmtx table has already been downloaded as part of the GT segment in the font header. (For vertical writing, it also assumes that any existing vmtx table has already been downloaded as part of the GT segment in the font header.)

Class 0 TrueType characters have the following format:

Byte	15(MSB)	8	7	(LSB)0	Byte
0	Format = 1		Class = 0		1
2	Character Data Size = X				3
4	TrueType Glyph ID				5
6	TrueType Glyph Data…				7
...					…
X				…Goes Here	X+1

PCL XL Assembly Example:

```
uint16 39 CharCode
uint32 106 CharDataSize
ReadChar
dataLength 106
hex_raw* [
      01    // Format Must Be 1
      00    // Class = 0
      00 68 // Character Data Size = 104
      00 09 // TrueType Glyph ID = 9
      00 01 00 bc 03 d8 01 c6 06 1f 00 03 00 47 b0 15
      2b 58 b1 01 00 b8 01 00 40 09 01 08 01 04 00 1e
      ff 00 01 b8 00 00 b2 00 0f 04 2b b1 01 02 b0 00
      40 09 01 0c 01 05 01 1e ff 00 01 b8 00 00 b2 02
      10 04 2b 00 b3 01 54 03 39 3f ec 01 b2 02 5d 00
      10 ec 31 30 59 13 03 21 03 f8 3c 01 0a 3c 03 d8
      02 47 fd b9
]
```

4.4.1.2 Class 1

Class 1 should be used when a minority of the glyphs in the font will be downloaded. Using this class eliminates the need to download the hmtx table in the font header. (For vertical writing, it assumes that any existing vmtx table has already been downloaded as part of the GT segment in the font header.)

Class 1 TrueType characters have the following format:

Byte	15(MSB)	8	7	(LSB)0	Byte
0	Format = 1		Class = 1		1
2	Character Data Size = X				3
4	Left Side Bearing (from the hmtx table)				5
6	Advance Width (from the hmtx table)				7
8	TrueType Glyph ID				9
10	TrueType Glyph Data…				11
...					…
X				…Goes Here	X+1

PCL XL Assembly Example:

```
uint16 39 CharCode
uint32 110 CharDataSize
dataLength 110
hex_raw* [
      01    // Format Must Be 1
      01    // Class = 1
      00 6c // Character Data Size = 108
      00 bc // Left Side Bearing = 188
      02 82 // Advance Width = 642
      00 09 // TrueType Glyph ID = 9
      00 01 00 bc 03 d8 01 c6 06 1f 00 03 00 47 b0 15
      2b 58 b1 01 00 b8 01 00 40 09 01 08 01 04 00 1e
      ff 00 01 b8 00 00 b2 00 0f 04 2b b1 01 02 b0 00
      40 09 01 0c 01 05 01 1e ff 00 01 b8 00 00 b2 02
      10 04 2b 00 b3 01 54 03 39 3f ec 01 b2 02 5d 00
      10 ec 31 30 59 13 03 21 03 f8 3c 01 0a 3c 03 d8
      02 47 fd b9
]
```

4.4.1.3 Class 2

Class 2 should be used when: 1) a minority of the glyphs in the font will be downloaded; 2) the font will be used for vertical-rotated writing; **and** 3) a vmtx table exists in the TrueType font file. Using this class eliminates the need to download the hmtx and vmtx tables as part of the GT segment in the font header.

Class 2 TrueType characters have the following format:

Byte	15(MSB)	8	7	(LSB)0	Byte
0	Format = 1		Class = 2		1
2	Character Data Size = X				3
4	Left Side Bearing (from the hmtx table)				5
6	Advance Width (from the hmtx table)				7
8	Top Side Bearing (from the vmtx table)				9
10	TrueType Glyph ID				11
12	TrueType Glyph Data…				13
...					…
X	…Goes Here				X+1

PCL XL Assembly Example:

```
uint16 33 CharCode
uint32 98 CharDataSize
ReadChar
dataLength 98
hex_raw* [
      01    // Format Must Be 1
      02    // Class = 2
      00 60 // Character Data Size = 96
      00 32 // Left Side Bearing = 50
      00 80 // Advance Width = 128
      00 2D // Top Side Bearing = 45
      02 01 // TrueType Glyph ID = 513
      00 02 00 32 00 04 00 4E 00 AF 00 03 00 07 00 31
      B1 00 03 B8 01 00 B3 01 02 07 06 41 0A 01 10 00
      04 00 05 01 04 00 00 01 13 00 03 00 04 01 10 00
      05 2F ED D4 FD 00 3F 3C FD 3C DE 3C 3F 3C 31 30
      01 37 07 23 27 17 23 35 33 4E 06 10 06 19 16 16
      AF 71 71 AB 16 00
]
```

4.4.2 Downloading Special TrueType Glyphs

"Special" TrueType glyphs are those that are not directly associated with a character code. The kinds of special glyphs include:

- the glyph used to represent the default galley character (usually TrueType glyph 0) if specified by a GC segment
- vertical substitute glyphs indicated by a VT segment
- glyph pieces (glyphs referenced by composite characters).

When downloading special glyphs, use one of the formats mentioned in section 4.4.1 depending on the circumstances, but specify the value **0xFFFF** for the **CharCode** attribute to the ReadChar operator. This "special" **CharCode** value tells PCL XL 2.0 that it is a "special" glyph.

Note: Special glyphs are not counted in the **Number Of Characters** field of the **PCL XL 2.0 Format 0 Font Header**.

PCL XL Assembly Example:

```
uint16 65535 CharCode   // CharCode Must Be 0xFFFF
uint32 48 CharDataSize
ReadChar
dataLength 48
hex_raw* [
      01    // Format Must Be 1
      02    // Class = 2 (could be any class, as appropriate)
      00 2E // Character Data Size = 46
      00 64 // LeftSide Bearing = 100
      01 00 // Advance Width = 256
      00 41 // Top Side Bearing = 65
      00 00 // TrueType Glyph ID = 0 (TrueType Default Glyph)
      00 01 00 64 00 64 00 9B 00 9B 00 03 00 0C B3 02
      01 00 01 2F DD 00 2F DD 31 30 37 23 35 33 9B 37
      37 64 37 00
]
```