The Compact Font Format Specification

Adobe Developer Support

Technical Note #5176

Version 1.0

16 March 2000

Contents

The Compact Font Format Specification

1 Introduction

This document describes the specification of a font format that is suitable for compactly representing one or more Type 1 or CID-keyed fonts. Unlike previous Type 1 and CID-keyed font formats, CFF allows multiple fonts to be stored together in a unit called a *FontSet*. Principal space savings are a result of using a compact binary representation for most of the information, sharing of common data between fonts, and defaulting frequently occurring data.

The CFF format is designed to be used in conjunction with Type 2 charstrings for the character description procedures (see Adobe Technical Note #5177: "The Type 2 Charstring Format").

The design supports the embedding of PostScript® language code which permits additional flexibility and extensibility of the format when used in printer environments (see Appendix E).

2 Data Layout

Conceptually the binary data is organized as a number of separate data structures. The overall layout within the binary data is shown Table 1. Since some of these data structures are reached via offsets the ordering could be changed, although the first five occupy fixed locations.

Table 1 CFF Data Layout

Entry	Comments
Header	–
Name INDEX	–
Top DICT INDEX	–
String INDEX	–
Global Subr INDEX	–
Encodings	–
Charsets	–
FDSelect	CIDFonts only
CharStrings INDEX	per-font
Font DICT INDEX	per-font, CIDFonts only
Private DICT	per-font
Local Subr INDEX	per-font or per-Private DICT for CIDFonts
Copyright and Trademark Notices	–

Appendix D shows an annotated example of a CFF font.

3 Data Types

This section describes data representation and types used by the format.

All multi-byte numeric data and offset fields are stored in big-endian byte order (high byte low offset) and do *not* honor any alignment restrictions. This leads to a format that is free from padding bytes.

Data objects are often specified by byte offsets that are relative to some reference point within the CFF data. These offsets are 1 to 4 bytes in length. This document uses the convention of enclosing the reference point in parentheses and uses a reference point of (0) to indicate an offset relative to the start of the CFF data and (self) to indicate an offset relative to the data structure containing the offset.

The CFF format data types are shown in Table 2.

Table 2 CFF Data Types

Name	Range	Description
Card8	0 – 255	1-byte unsigned number
Card16	0 – 65535	2-byte unsigned number
Offset	varies	1, 2, 3, or 4 byte offset (specified by OffSize field)
OffSize	1– 4	1-byte unsigned number specifies the size of an Offset field or fields
SID	0 – 64999	2-byte string identifier

This document describes data structures by listing field types, names, and descriptions. Data structures may be given a type name and subsequently described. Arrays of objects are indicated by the usual square bracket convention enclosing the array length.

The majority of CFF data is contained by either of two data structures called DICT and INDEX which are described in subsequent sections.

4 DICT Data

Font dictionary data comprising key-value pairs is represented in a compact tokenized format that is similar to that used to represent Type 1 charstrings. Dictionary keys are encoded as 1- or 2-byte operators and dictionary values are encoded as variable-size numeric operands that represent either integer or real values. An operator is preceded by the operand(s) that specify its value. A DICT is simply a sequence of operand(s)/operator bytes concatenated together.

A number of integer operand types of varying sizes are defined and are encoded as shown in Table 3 (first byte of operand is b0,

second is b1, and so on).

Table 3 Operand Encoding

Size	b0 range	Value range	Value calculation
1	32 – 246	−107 – +107	b0−139
2	247 – 250	+108 – +1131	(b0−247)∗256+b1+108
2	251 – 254	−1131 – −108	−(b0−251)∗256−b1−108
3	28	−32768 – +32767	b1<<8|b2
5	29	$-(2^{31})$ – $+(2^{31}-1)$	b1<<24|b2<<16|b3<<8|b4

Note 1　*The 1- , 2-, and 3- byte integer formats are identical to those used by Type 2 charstrings.*

Examples of the integer formats are shown in Table 4.

Table 4 Integer Format Examples

Value	Encoding
0	8b
100	ef
−100	27
1000	fa 7c
−1000	fe 7c
10000	1c 27 10
−10000	1c d8 f0
100000	1d 00 01 86 a0
−100000	1d ff fe 79 60

A real number operand is provided in addition to integer operands. This operand begins with a byte value of 30 followed by a variable-length sequence of bytes. Each byte is composed of two 4-bit nibbles as defined in Table 5. The first nibble of a pair is stored in the most significant 4 bits of a byte and the second nibble of a pair is stored in the least significant 4 bits of

a byte.

Table 5 Nibble Definitions

Nibble	Represents
0 – 9	0 – 9
a	. (decimal point)
b	E
c	E–
d	<reserved>
e	– (minus)
f	end of number

A real number is terminated by one (or two) 0xf nibbles so that it is always padded to a full byte. Thus, the value −2.25 is encoded by the byte sequence (1e e2 a2 5f) and the value 0.140541E−3 by the sequence (1e 0a 14 05 41 c3 ff).

Operators and operands may be distinguished by inspection of their first byte: 0–21 specify operators and 28, 29, 30, and 32–254 specify operands (numbers). Byte values 22–27, 31, and 255 are reserved. An operator may be preceded by up to a maximum of 48 operands.

An operator may have one or more operands of the types shown in Table 6.

Table 6 Operand Types

Type	Description
number	Integer or real number
boolean	Integer type with the values 0 (false) and 1 (true)
SID	String id (see section 10)
array	One or more numbers
delta	A number or a delta-encoded array of numbers (see below)

The length of array or delta types is determined by counting the operands preceding the operator. The second and subsequent numbers in a delta are encoded as the difference between

successive values. For example, an array a0, a1, ..., an would be encoded as: a0 (a1–a0) (a2–a1) ..., (an–a(n–1)).

Two-byte operators have an initial escape byte of 12.

Further compaction of dictionary data is achieved by establishing default values for various DICT keys. For those keys that have a default value the absence of the corresponding operator in a DICT implies a key should take its default value.

(A list of DICT operators for the Top and Private DICTs may be found in sections 9 and 15, respectively.)

5 INDEX Data

An INDEX is an array of variable-sized objects. It comprises a *header*, an *offset array*, and *object data*. The offset array specifies offsets within the object data. An object is retrieved by indexing the offset array and fetching the object at the specified offset. The object's length can be determined by subtracting its offset from the next offset in the offset array. An additional offset is added at the end of the offset array so the length of the last object may be determined. The INDEX format is shown in Table 7.

Table 7 INDEX Format

Type	Name	Description
Card16	count	Number of objects stored in INDEX
OffSize	offSize	Offset array element size
Offset	offset [count+1]	Offset array (from byte preceding object data)
Card8	data [<varies>]	Object data

Offsets in the offset array are relative to the byte that precedes the object data. Therefore the first element of the offset array is always 1. (This ensures that every object has a corresponding offset which is always nonzero and permits the efficient implementation of dynamic object loading.)

An empty INDEX is represented by a count field with a 0 value

and no additional fields. Thus, the total size of an empty INDEX is 2 bytes.

Note 2   *An INDEX may be skipped by jumping to the offset specified by the last element of the offset array.*

6 Header

The binary data begins with a header having the format shown in Table 8.

Table 8 Header Format

Type	Name	Description
Card8	major	Format major version (starting at 1)
Card8	minor	Format minor version (starting at 0)
Card8	hdrSize	Header size (bytes)
OffSize	offSize	Absolute offset (0) size

Implementations reading font set files must include code to check version numbers so that if and when the format and therefore the version number changes, older implementations will reject newer versions gracefully. If the major version number is understood by an implementation it can safely proceed with reading the font. The minor version number indicates extensions to the format that are undetectable by implementations that do not support them although they will be unable to take advantage of these extensions.

The hdrSize field must be used when locating the Name INDEX. It is provided so that future versions of the format may introduce additional data between the offSize field and the Name INDEX in a manner that is compatible with older implementations.

The offSize field specifies the size of all offsets (0) relative to the start of CFF data.

7 Name INDEX

This contains the PostScript language names (FontName or CIDFontName) of all the fonts in the FontSet stored in an INDEX structure. The font names are sorted, thereby permitting a

binary search to be performed when locating a specific font within a FontSet. The sort order is based on character codes treated as 8-bit unsigned integers. A given font name precedes another font name having the first name as its prefix. There must be at least one entry in this INDEX, i.e. the FontSet must contain at least one font.

For compatibility with client software, such as PostScript interpreters and Acrobat®, font names should be no longer than 127 characters and should not contain any of the following ASCII characters: [,], (,), {, }, <, >, /, %, null (NUL), space, tab, carriage return, line feed, form feed. It is recommended that font names be restricted to the printable ASCII subset, codes 33 through 126. Adobe Type Manager® (ATM®) software imposes a further restriction on the font name length of 63 characters.

Note 3   *For compatibility with earlier PostScript interpreters, see Technical Note #5088, "Font Naming Issues."*

A font may be deleted from a FontSet without removing its data by setting the first byte of its name in the Name INDEX to 0 (NUL). This kind of deletion offers a simple way to handle font upgrades without rebuilding entire fontsets. Binary search software must detect deletions and restart the search at the previous or next name in the INDEX to ensure that all appropriate names are matched.

8 Top DICT INDEX

This contains the top-level DICTs of all the fonts in the FontSet stored in an INDEX structure. Objects contained within this INDEX correspond to those in the Name INDEX in both order and number. Each object is a DICT structure that corresponds to the top-level dictionary of a PostScript font.

A font is identified by an entry in the Name INDEX and its data is accessed via the corresponding Top DICT.

9 Top DICT Data

The names of the Top DICT operators shown in Table 9 are, where possible, the same as the corresponding Type 1 dict key. Operators that have no corresponding Type 1 dict key are noted in the table below along with a default value, if any. (Several

operators have been derived from FontInfo dict keys but have been grouped together with the Top DICT operators for simplicity. The keys from the FontInfo dict are indicated in the *Default,notes* column of Table 9.)

Table 9 Top DICT Operator Entries

Name	Value	Operand(s)	Default, notes
version	0	SID	–, FontInfo
Notice	1	SID	–, FontInfo
Copyright	12 0	SID	–, FontInfo
FullName	2	SID	–, FontInfo
FamilyName	3	SID	–, FontInfo
Weight	4	SID	–, FontInfo
isFixedPitch	12 1	boolean	0 (false), FontInfo
ItalicAngle	12 2	number	0, FontInfo
UnderlinePosition	12 3	number	–100, FontInfo
UnderlineThickness	12 4	number	50, FontInfo
PaintType	12 5	number	0
CharstringType	12 6	number	2
FontMatrix	12 7	array	0.001 0 0 0.001 0 0
UniqueID	13	number	–
FontBBox	5	array	0 0 0 0
StrokeWidth	12 8	number	0
XUID	14	array	–
charset	15	number	0, charset offset (0)
Encoding	16	number	0, encoding offset (0)
CharStrings	17	number	–, CharStrings offset (0)
Private	18	number number	–, Private DICT size and offset (0)
SyntheticBase	12 20	number	–, synthetic base font index
PostScript	12 21	SID	–, embedded PostScript language code

Table 9 Top DICT Operator Entries (continued)

Name	Value	Operand(s)	Default, notes
BaseFontName	12 22	SID	–, (added as needed by Adobe-based technology)
BaseFontBlend	12 23	delta	–, (added as needed by Adobe-based technology)

The embedded **PostScript** operator provides an escape mechanism that may be used to address extensibility or compatibility issues in a printer font (see Appendix F).

The separation of dictionary data into top-level and Private dictionaries reflects Type 1 usage where Top DICT data is parsed at findfont time and used to construct a valid Type 1 font dictionary. The **Private** operator value specifies a size and an offset that is followed at font rendering time in order to construct the data structures associated with Private DICT data.

The operators **BaseFontName** and **BaseFontBlend** are added to multiple master font instances embedded in PDF documents. **BaseFontName** is the **FontName** of the original multiple master font and **BaseFontBlend** is the User Design Vector used to compute the instance.

CIDFonts require the additional Top DICT operators shown in Table 10.

Table 10 CIDFont Operator Extensions

Name	Value	Operand(s)	Default, notes
ROS	12 30	SID SID number	–, Registry Ordering Supplement
CIDFontVersion	12 31	number	0
CIDFontRevision	12 32	number	0
CIDFontType	12 33	number	0
CIDCount	12 34	number	8720
UIDBase	12 35	number	–
FDArray	12 36	number	–, Font DICT (FD) INDEX offset (0)

Table 10 CIDFont Operator Extensions

Name	Value	Operand(s)	Default, notes
FDSelect	12 37	number	–, FDSelect offset (0)
FontName	12 38	SID	–, FD FontName

The **ROS** operator combines the Registry, Ordering, and Supplement keys together. CIDFonts are described fully in section 18.

The Top DICT begins with the **SyntheticBase** and **ROS** operators for synthetic and CIDFonts, respectively. Regular Type 1 fonts begin with some other operator. (This permits the determination of the kind of font without parsing the entire Top DICT.)

10 String INDEX

All the strings, with the exception of the FontName and CIDFontName strings which appear in the Name INDEX, used by different fonts within the FontSet are collected together into an INDEX structure and are referenced by a 2-byte unsigned number called a string identifier or SID. Only unique strings are stored in the table thereby removing duplication across fonts. Further space saving is obtained by allocating commonly occurring strings to predefined SIDs. These strings, known as the standard strings, describe all the names used in the ISOAdobe and Expert character sets along with a few other strings common to Type 1 fonts. A complete list of standard strings is given in Appendix A.

The client program will contain an array of standard strings with nStdStrings elements. Thus, the standard strings take SIDs in the range 0 to (nStdStrings–1). The first string in the String INDEX corresponds to the SID whose value is equal to nStdStrings, the first non-standard string, and so on. When the client needs to determine the string that corresponds to a particular SID it performs the following: test if SID is in standard range then fetch from internal table, otherwise, fetch string from the String INDEX using a value of (SID–nStdStrings) as the index.

An SID is defined as a 2-byte unsigned number but only takes values in the range 0–64999, inclusive. SID values 65000 and above are available for implementation use.

A FontSet with zero non-standard strings is represented by an empty INDEX.

11 Glyph Organization

The glyphs within a font constitute a charset and are accessed via an encoding. An encoding is an array of codes associated with some or all glyphs in a font and a charset is an array of "names" for all glyphs in the font. (In CFF these names are actually SIDs or CIDs, which must be unique.)

In order to understand how charsets, encodings, and glyphs are related in CFF it is useful to think of them as 3 "parallel" arrays that are indexed in unison. Thus, it is possible to name and encode the glyph at the given glyph index (GID) by using the GID to index the charset and encoding arrays, respectively. By definition the first glyph (GID 0) is ".notdef" and must be present in all fonts. Since this is always the case, it is not necessary to represent either the encoding (unencoded) or name (.notdef) for GID 0. Consequently, taking advantage of this optimization, the encoding and charset arrays always begin with GID 1.

12 Encodings

Encoding data is located via the offset operand to the **Encoding** operator in the Top DICT. Only one **Encoding** operator can be specified per font except for CIDFonts which specify no encoding. A glyph's encoding is specified by a 1-byte code that permits values in the range 0–255.

Each encoding is described by a format-type identifier byte followed by format-specific data. Two formats are currently defined as specified in Tables 11 and 12.

Table 11 Format 0

Type	Name	Description
Card8	format	=0

Table 11 Format 0

Type	Name	Description
Card8	nCodes	Number of encoded glyphs
Card8	code [nCodes]	Code array

Each element of the code array represents the encoding for the corresponding glyph. This format should be used when the codes are in a fairly random order.

Table 12 Format 1

Type	Name	Description
Card8	format	=1
Card8	nRanges	Number of code ranges
struct	Range1 [nRanges]	Range1 array (see Table 13)

The format of a Range1 is described in Table 13.

Table 13 Range1 Format (Encoding)

Type	Name	Description
Card8	first	First code in range
Card8	nLeft	Codes left in range (excluding first)

Each Range1 describes a group of sequential codes. For example, the codes 51 52 53 54 55 could be represented by the Range1: 51 4, and a perfectly ordered encoding of 256 codes can be described with the Range1: 0 255.

This format is particularly suited to encodings that are well ordered.

A few fonts have multiply-encoded glyphs which are not supported directly by any of the above formats. This situation is indicated by setting the high-order bit in the format byte and supplementing the encoding, regardless of format type, as

shown in Table 14.

Table 14 Supplemental Encoding Data

Type	Name	Description
Card8	nSups	Number of supplementary mappings
struct	Supplement [nSups]	Supplementary encoding array (see Table 15 below)

The format of a Supplement is specified in Table 15.

Table 15 Supplement Format

Type	Name	Description
Card8	code	Encoding
SID	glyph	Name

Each Supplement describes a single code-to-glyph mapping which provides another encoding for a glyph that has already been mentioned in the main encoding table.

Sorting glyphs by encoding and then SID for unencoded glyphs (remembering that .notdef must be first) typically yields very small font encodings. Still more optimization is possible by observing that many fonts adopt one of two common encodings. In these cases the operand to the **Encoding** operator in the Top DICT specifies a predefined encoding id, in place of an offset, as defined in Table 16.

Table 16 Encoding ID

Id	Name
0	Standard Encoding
1	Expert Encoding

If the font uses Standard Encoding the **Encoding** operator can be omitted from the Top DICT since its default value is 0. Details of predefined encodings can be found in Appendix B.

It is not necessary for a font to contain all the glyphs specified by a predefined encoding in order to be able to use it. The only requirement is that every glyph in the font has an identical

encoding to those in the predefined encoding (including unencoded glyphs).

Two or more fonts may share the same encoding by setting the offset operand of the **Encoding** operator to the same value in each font.

By definition, glyphs that are not specified by custom or predefined encodings are unencoded.

Note 4   *Predefined encodings may be applied to a variety of fonts regardless of charset whereas custom encodings may only be applied to fonts with specific charsets. Consequently, predefined encodings are specified as code to SID mappings and custom encodings are specified as code to GID mappings.*

13 Charsets

Charset data is located via the offset operand to the **charset** operator in the Top DICT. Each charset is described by a format-type identifier byte followed by format-specific data. Three formats are currently defined as shown in Tables 17, 18, and 20.

Table 17 Format 0

Type	Name	Description
Card8	format	=0
SID	glyph [nGlyphs–1]	Glyph name array

Each element of the glyph array represents the name of the corresponding glyph. This format should be used when the SIDs are in a fairly random order. The number of glyphs (nGlyphs) is the value of the count field in the **CharStrings** INDEX. (There is one less element in the glyph name array than nGlyphs because the .notdef glyph name is omitted.)

Table 18 Format 1

Type	Name	Description
Card8	format	=1
struct	Range1 [<varies>]	Range1 array (see Table 19)

The Range1 format is shown in Table 19.

Table 19 Range1 Format (Charset)

Type	Name	Description
SID	first	First glyph in range
Card8	nLeft	Glyphs left in range (excluding first)

Each Range1 describes a group of sequential SIDs. The number of ranges is not explicitly specified in the font. Instead, software utilizing this data simply processes ranges until all glyphs in the font are covered. This format is particularly suited to charsets that are well ordered.

Table 20 Format 2

Type	Name	Description
Card8	format	=2
struct	Range2 [<varies>]	Range2 array (see Table 21)

The Range2 format is shown in Table 21.

Table 21 Range2 Format

Type	Name	Description
SID	first	First glyph in range
Card16	nLeft	Glyphs left in range (excluding first)

Format 2 differs from format 1 only in the size of the nLeft field in each range. This format is most suitable for fonts with a large well-ordered charset— for example, for Asian CIDFonts.

Careful attention to the allocation order of SIDs typically yields very small font charsets. Still more optimization is possible by observing that many fonts adopt one of 3 common charsets. In these cases the operand to the **charset** operator in the Top DICT specifies a predefined charset id, in place of an offset, as shown

in Table 22.

Table 22 Charset ID

Id	Name
0	ISOAdobe
1	Expert
2	ExpertSubset

If the font has an ISOAdobe charset, the **charset** operator can be omitted from the Top DICT since its default value is 0. Details of predefined charsets can be found in Appendix C. A font may use a predefined charset if it exactly matches in the first nGlyphs. CID fonts must not use predefined charsets.

Two or more fonts may share the same charset by setting the offset operand of the **charset** operator to the same value in each font.

14 CharStrings INDEX

This contains the charstrings of all the glyphs in a font stored in an INDEX structure. Charstring objects contained within this INDEX are accessed by GID. The first charstring (GID 0) must be the .notdef glyph. The number of glyphs available in a font may be determined from the count field in the INDEX.

The format of the charstring data, and therefore the method of interpretation, is specified by the **CharstringType** operator in the Top DICT. The **CharstringType** operator has a default value of 2 indicating the Type 2 charstring format which was designed in conjunction with CFF. Type 1 charstrings are documented in the "Adobe Type 1 Font Format" published by Addison-Wesley. Type 2 charstrings are described in Adobe Technical Note #5177: "Type 2 Charstring Format." Other charstring types may also be supported by this method.

15 Private DICT Data

The names of the Private DICT operators shown in Table 23 are, where possible, the same as the corresponding Type 1 dict keys. Operators that have no corresponding Type 1 dict key are

indicated with a note in Table 23.

Table 23 Private DICT Operators

Name	Value	Operand(s)	Default, notes
BlueValues	6	delta	–
OtherBlues	7	delta	–
FamilyBlues	8	delta	–
FamilyOtherBlues	9	delta	–
BlueScale	12 9	number	0.039625
BlueShift	12 10	number	7
BlueFuzz	12 11	number	1
StdHW	10	number	–
StdVW	11	number	–
StemSnapH	12 12	delta	–
StemSnapV	12 13	delta	–
ForceBold	12 14	boolean	false
LanguageGroup	12 17	number	0
ExpansionFactor	12 18	number	0.06
initialRandomSeed	12 19	number	0
Subrs	19	number	–, Offset (self) to local subrs
defaultWidthX	20	number	0, see below
nominalWidthX	21	number	0, see below

The local subrs offset is relative to the beginning of the Private DICT data.

The **defaultWidthX** and **nominalWidthX** operators supply width values for glyphs. If a glyph width equals the **defaultWidthX** value it can be omitted from the charstring, otherwise the glyph width is computed by adding the charstring width to **nominalWidthX** value. If **nominalWidthX** is carefully chosen the bulk of the widths in the charstrings can be reduced from 2-byte to single-byte numbers thereby saving space.

The **OtherBlues** and **FamilyOtherBlues** operators must occur

after the **BlueValues** and **FamilyBlues** operators, respectively.

A Private DICT is required, but may be specified as having a length of 0 if there are no non-default values to be stored.

16 Local / Global Subrs INDEXes

Both Type 1 and Type 2 charstrings support the notion of subroutines or subrs. A subr is typically a sequence of charstring bytes representing a sub-program that occurs in more than one place in a font's charstring data. This subr may be stored once but referenced many times from within one or more charstrings by the use of the **callsubr** operator whose operand is the number of the subr to be called. The subrs are local to a particular font and cannot be shared between fonts. Type 2 charstrings also permit global subrs which function in the same way but are called by the **callgsubr** operator and may be shared across fonts.

Local subrs are stored in an INDEX structure which is located via the offset operand of the **Subrs** operator in the Private DICT. A font without local subrs has no **Subrs** operator in the Private DICT.

Global subrs are stored in an INDEX structure which follows the String INDEX. A FontSet without any global subrs is represented by an empty Global Subrs INDEX.

Subr numbers in Type 2 charstrings are skewed by a number called the "subr number bias" which is calculated from the count of the subroutines in either the local or global subr INDEXes. The bias is calculated as follows:

```
Card16 bias;
Card16 nSubrs = subrINDEX.count;

if (CharstringType == 1)
     bias = 0;
else if (nSubrs < 1240)
     bias = 107;
else if (nSubrs < 33900)
     bias = 1131;
else
     bias = 32768;
```

For correct subr selection the calculated bias must be added to

the subr number operand before accessing the appropriate subr INDEX. This technique allows subr numbers to be specified using negative as well as positive numbers thereby fully utilizing the available number ranges and thus saving space. (The above calculation obviates the need for an explicit bias to be stored in the font which is currently the case for Type 1 fonts.) Tables 24, 25, and 26 show the relationship between subr indices, numbers, number sizes and range counts for the different biasing schemes (column headings are described following Table 26).

Table 24 nSubrs < 1240, bias = 107

Ordered index			Reorder index			Biased number			Size	Count
0	–	214	0	–	214	–107	–	+107	1	215
215	–	1238	215	–	1238	+108	–	+1131	2	1024
								Total:		1239

Table 25 nSubrs < 33900, bias = 1131

Ordered index			Reorder index			Biased number			Size	Count
0	–	214	1024	–	1238	–107	–	+107	1	215
215	–	1238	0	–	1023	–1131	–	–108	2	1024
1239	–	2262	1239	–	2262	+108	–	+1131	2	1024
2263	–	33898	2263	–	33898	+1132	–	+32767	3	31636
								Total:		33899

Table 26 nSubrs >= 33900, bias = 32768

Ordered index			Reorder index			Biased number			Size	Count
0	–	214	32661	–	32875	–107	–	+107	1	215
215	–	1238	31637	–	32660	–1131	–	–108	2	1024
1239	–	2262	32876	–	33899	+108	–	+1131	2	1024
2263	–	33899	0	–	31636	–32768	–	–1132	3	31637
33900	–	65535	33900	–	65535	+1132	–	+32767	3	31636
								Total:		65536

Where the column headings are interpreted as follows:

Ordered index	subr index ordered by most frequent subr first.
Reorder index	ordered index reordered for bias (this is the index of subrs in the local/global Subr INDEX structures).
Biased number	reorder index with bias subtracted (callsubr/callgsubr operand).
Size	biased number size (bytes).
Count	number of subrs in range.

17 Synthetic Fonts

A synthetic font is a modification of another font by means of a different transformation matrix or encoding. Obliqued, expanded, and condensed fonts are examples of fonts that may be constructed as synthetic fonts.

Synthetic fonts have a name and a Top DICT that refers to a base font. The Top DICT may contain the following operators: **FullName**, **ItalicAngle**, **FontMatrix**, **SyntheticBase**, and **Encoding**.

The **SyntheticBase** operator is required and specifies the zero-based index of the font that is to be used as the base font. The **FontMatrix** and/or Encoding is applied to this font in order to algorithmically create a new font. The other operator values override those given in the base font. The Top DICT must begin with a **SyntheticBase** operator. The base font shall not be a CID-keyed font or a synthetic font.

18 CID-keyed Fonts

The representation of a CIDFont is designed to be separable from its encoding. In keeping with this strategy, the CFF representation does not include any encoding information which instead resides in a CMap file. If a need arises for a more compact representation of the CMap file, CFF can be extended to accommodate it.

A CFF CIDFont has the **CIDFontName** in the Name INDEX and a corresponding Top DICT. The Top DICT begins with **ROS** operator which specifies the Registry-Ordering-Supplement for

the font. This will indicate to a CFF parser that special CID processing should be applied to this font. Specifically:

- The **FDArray** operator is expected to be present, with a single argument specifying an offset to the Font DICT INDEX. Each Font DICT in this array specifies information unique to a particular group of glyphs in the font. The mapping of glyphs to Font DICTs is specified by the FDSelect structure described below. Each Font DICT will specify a corresponding Private DICT with the **Private** DICT operator.

- The charset data, although in the same format as non-CIDFonts, will represent CIDs rather than SIDs, i.e. charstrings are "named" by CIDs in a CIDFont. In a complete CIDFont the charset table will specify an identity mapping (where GID equals CID for all glyphs) as a single range beginning at CID 1 (CID 0, the .notdef glyph, is omitted) that covers all the glyphs in the font. Subset CIDFonts will generally need to use a more complex charset table representing a non-identity mapping (where CID doesn't equal GID).

- The Top DICT will include an **FDSelect** operator specifying an offset to a charset-like data structure (see next section) which contains a, possibly range-encoded, list of indexes, from which a single index may be derived for each glyph. The index identifies the Font DICT, and therefore the Private DICT, to be used when rasterizing a glyph.

- The encoding data is omitted (see above); no **Encoding** operator will be present and the default StandardEncoding should not be applied.

There are no predefined charsets for CID fonts.

19 FDSelect

The FDSelect associates an FD (Font DICT) with a glyph by specifying an FD index for that glyph. The FD index is used to access one of the Font DICTs stored in the Font DICT INDEX.

FDSelect data is located via the offset operand to the **FDSelect** operator in the Top DICT. FDSelect data specifies a format-type identifier byte followed by format-specific data. Two formats

are currently defined, as shown in Tables 27 and 28.

Table 27 Format 0

Type	Name	Description
Card8	format	=0
Card8	fds [nGlyphs]	FD selector array

Each element of the fd array (fds) represents the FD index of the corresponding glyph. This format should be used when the FD indexes are in a fairly random order. The number of glyphs (nGlyphs) is the value of the count field in the CharStrings INDEX. (This format is identical to charset format 0 except that the .notdef glyph is included in this case.)

Table 28 Format 3

Type	Name	Description
Card8	format	=3
Card16	nRanges	Number of ranges
struct	Range3 [nRanges]	Range3 array (see Table 29)
Card16	sentinel	Sentinel GID (see below)

The format of a Range3 is shown in Table 29.

Table 29 Range3 Format

Type	Name	Description
Card16	first	First glyph index in range
Card8	fd	FD index for all glyphs in range

Each Range3 describes a group of sequential GIDs that have the same FD index. Each range includes GIDs from the 'first' GID up to, but not including, the 'first' GID of the next range element. Thus, elements of the Range3 array are ordered by increasing 'first' GIDs. The first range must have a 'first' GID of 0. A sentinel GID follows the last range element and serves to delimit the last range in the array. (The sentinel GID is set equal to the number of glyphs in the font. That is, its value is 1 greater than the last GID in the font.) This format is particularly suited to FD indexes that are well ordered (the usual case).

Appendix A
Standard Strings

(SID / name)

0	.notdef	21	four	42	I
1	space	22	five	43	J
2	exclam	23	six	44	K
3	quotedbl	24	seven	45	L
4	numbersign	25	eight	46	M
5	dollar	26	nine	47	N
6	percent	27	colon	48	O
7	ampersand	28	semicolon	49	P
8	quoteright	29	less	50	Q
9	parenleft	30	equal	51	R
10	parenright	31	greater	52	S
11	asterisk	32	question	53	T
12	plus	33	at	54	U
13	comma	34	A	55	V
14	hyphen	35	B	56	W
15	period	36	C	57	X
16	slash	37	D	58	Y
17	zero	38	E	59	Z
18	one	39	F	60	bracketleft
19	two	40	G	61	backslash
20	three	41	H	62	bracketright

63	asciicircum	92	braceleft	121	ellipsis
64	underscore	93	bar	122	perthousand
65	quoteleft	94	braceright	123	questiondown
66	a	95	asciitilde	124	grave
67	b	96	exclamdown	125	acute
68	c	97	cent	126	circumflex
69	d	98	sterling	127	tilde
70	e	99	fraction	128	macron
71	f	100	yen	129	breve
72	g	101	florin	130	dotaccent
73	h	102	section	131	dieresis
74	i	103	currency	132	ring
75	j	104	quotesingle	133	cedilla
76	k	105	quotedblleft	134	hungarumlaut
77	l	106	guillemotleft	135	ogonek
78	m	107	guilsinglleft	136	caron
79	n	108	guilsinglright	137	emdash
80	o	109	fi	138	AE
81	p	110	fl	139	ordfeminine
82	q	111	endash	140	Lslash
83	r	112	dagger	141	Oslash
84	s	113	daggerdbl	142	OE
85	t	114	periodcentered	143	ordmasculine
86	u	115	paragraph	144	ae
87	v	116	bullet	145	dotlessi
88	w	117	quotesinglbase	146	lslash
89	x	118	quotedblbase	147	oslash
90	y	119	quotedblright	148	oe
91	z	120	guillemotright	149	germandbls

150	onesuperior	179	Ecircumflex	208	ecircumflex
151	logicalnot	180	Edieresis	209	edieresis
152	mu	181	Egrave	210	egrave
153	trademark	182	Iacute	211	iacute
154	Eth	183	Icircumflex	212	icircumflex
155	onehalf	184	Idieresis	213	idieresis
156	plusminus	185	Igrave	214	igrave
157	Thorn	186	Ntilde	215	ntilde
158	onequarter	187	Oacute	216	oacute
159	divide	188	Ocircumflex	217	ocircumflex
160	brokenbar	189	Odieresis	218	odieresis
161	degree	190	Ograve	219	ograve
162	thorn	191	Otilde	220	otilde
163	threequarters	192	Scaron	221	scaron
164	twosuperior	193	Uacute	222	uacute
165	registered	194	Ucircumflex	223	ucircumflex
166	minus	195	Udieresis	224	udieresis
167	eth	196	Ugrave	225	ugrave
168	multiply	197	Yacute	226	yacute
169	threesuperior	198	Ydieresis	227	ydieresis
170	copyright	199	Zcaron	228	zcaron
171	Aacute	200	aacute	229	exclamsmall
172	Acircumflex	201	acircumflex	230	Hungarumlautsmall
173	Adieresis	202	adieresis	231	dollaroldstyle
174	Agrave	203	agrave	232	dollarsuperior
175	Aring	204	aring	233	ampersandsmall
176	Atilde	205	atilde	234	Acutesmall
177	Ccedilla	206	ccedilla	235	parenleftsuperior
178	Eacute	207	eacute	236	parenrightsuperior

237	twodotenleader	266	ff	295	Vsmall
238	onedotenleader	267	ffi	296	Wsmall
239	zerooldstyle	268	ffl	297	Xsmall
240	oneoldstyle	269	parenleftinferior	298	Ysmall
241	twooldstyle	270	parenrightinferior	299	Zsmall
242	threeoldstyle	271	Circumflexsmall	300	colonmonetary
243	fouroldstyle	272	hyphensuperior	301	onefitted
244	fiveoldstyle	273	Gravesmall	302	rupiah
245	sixoldstyle	274	Asmall	303	Tildesmall
246	sevenoldstyle	275	Bsmall	304	exclamdownsmall
247	eightoldstyle	276	Csmall	305	centoldstyle
248	nineoldstyle	277	Dsmall	306	Lslashsmall
249	commasuperior	278	Esmall	307	Scaronsmall
250	threequartersemdash	279	Fsmall	308	Zcaronsmall
251	periodsuperior	280	Gsmall	309	Dieresissmall
252	questionsmall	281	Hsmall	310	Brevesmall
253	asuperior	282	Ismall	311	Caronsmall
254	bsuperior	283	Jsmall	312	Dotaccentsmall
255	centsuperior	284	Ksmall	313	Macronsmall
256	dsuperior	285	Lsmall	314	figuredash
257	esuperior	286	Msmall	315	hypheninferior
258	isuperior	287	Nsmall	316	Ogoneksmall
259	lsuperior	288	Osmall	317	Ringsmall
260	msuperior	289	Psmall	318	Cedillasmall
261	nsuperior	290	Qsmall	319	questiondownsmall
262	osuperior	291	Rsmall	320	oneeighth
263	rsuperior	292	Ssmall	321	threeeighths
264	ssuperior	293	Tsmall	322	fiveeighths
265	tsuperior	294	Usmall	323	seveneighths

324	onethird	353	AEsmall	382	001.003
325	twothirds	354	Ccedillasmall	383	Black
326	zerosuperior	355	Egravesmall	384	Bold
327	foursuperior	356	Eacutesmall	385	Book
328	fivesuperior	357	Ecircumflexsmall	386	Light
329	sixsuperior	358	Edieresissmall	387	Medium
330	sevensuperior	359	Igravesmall	388	Regular
331	eightsuperior	360	Iacutesmall	389	Roman
332	ninesuperior	361	Icircumflexsmall	390	Semibold
333	zeroinferior	362	Idieresissmall		
334	oneinferior	363	Ethsmall		
335	twoinferior	364	Ntildesmall		
336	threeinferior	365	Ogravesmall		
337	fourinferior	366	Oacutesmall		
338	fiveinferior	367	Ocircumflexsmall		
339	sixinferior	368	Otildesmall		
340	seveninferior	369	Odieresissmall		
341	eightinferior	370	OEsmall		
342	nineinferior	371	Oslashsmall		
343	centinferior	372	Ugravesmall		
344	dollarinferior	373	Uacutesmall		
345	periodinferior	374	Ucircumflexsmall		
346	commainferior	375	Udieresissmall		
347	Agravesmall	376	Yacutesmall		
348	Aacutesmall	377	Thornsmall		
349	Acircumflexsmall	378	Ydieresissmall		
350	Atildesmall	379	001.000		
351	Adieresissmall	380	001.001		
352	Aringsmall	381	001.002		

Appendix B
Predefined Encodings

Standard Encoding (code / SID / name)

code	SID	name		code	SID	name		code	SID	name
0	0	.notdef		21	0	.notdef		42	11	asterisk
1	0	.notdef		22	0	.notdef		43	12	plus
2	0	.notdef		23	0	.notdef		44	13	comma
3	0	.notdef		24	0	.notdef		45	14	hyphen
4	0	.notdef		25	0	.notdef		46	15	period
5	0	.notdef		26	0	.notdef		47	16	slash
6	0	.notdef		27	0	.notdef		48	17	zero
7	0	.notdef		28	0	.notdef		49	18	one
8	0	.notdef		29	0	.notdef		50	19	two
9	0	.notdef		30	0	.notdef		51	20	three
10	0	.notdef		31	0	.notdef		52	21	four
11	0	.notdef		32	1	space		53	22	five
12	0	.notdef		33	2	exclam		54	23	six
13	0	.notdef		34	3	quotedbl		55	24	seven
14	0	.notdef		35	4	numbersign		56	25	eight
15	0	.notdef		36	5	dollar		57	26	nine
16	0	.notdef		37	6	percent		58	27	colon
17	0	.notdef		38	7	ampersand		59	28	semicolon
18	0	.notdef		39	8	quoteright		60	29	less
19	0	.notdef		40	9	parenleft		61	30	equal
20	0	.notdef		41	10	parenright		62	31	greater

63	32	question		92	61	backslash		121	90	y
64	33	at		93	62	bracketright		122	91	z
65	34	A		94	63	asciicircum		123	92	braceleft
66	35	B		95	64	underscore		124	93	bar
67	36	C		96	65	quoteleft		125	94	braceright
68	37	D		97	66	a		126	95	asciitilde
69	38	E		98	67	b		127	0	.notdef
70	39	F		99	68	c		128	0	.notdef
71	40	G		100	69	d		129	0	.notdef
72	41	H		101	70	e		130	0	.notdef
73	42	I		102	71	f		131	0	.notdef
74	43	J		103	72	g		132	0	.notdef
75	44	K		104	73	h		133	0	.notdef
76	45	L		105	74	i		134	0	.notdef
77	46	M		106	75	j		135	0	.notdef
78	47	N		107	76	k		136	0	.notdef
79	48	O		108	77	l		137	0	.notdef
80	49	P		109	78	m		138	0	.notdef
81	50	Q		110	79	n		139	0	.notdef
82	51	R		111	80	o		140	0	.notdef
83	52	S		112	81	p		141	0	.notdef
84	53	T		113	82	q		142	0	.notdef
85	54	U		114	83	r		143	0	.notdef
86	55	V		115	84	s		144	0	.notdef
87	56	W		116	85	t		145	0	.notdef
88	57	X		117	86	u		146	0	.notdef
89	58	Y		118	87	v		147	0	.notdef
90	59	Z		119	88	w		148	0	.notdef
91	60	bracketleft		120	89	x		149	0	.notdef

150	0	.notdef	179	113	daggerdbl	208	137	emdash
151	0	.notdef	180	114	periodcentered	209	0	.notdef
152	0	.notdef	181	0	.notdef	210	0	.notdef
153	0	.notdef	182	115	paragraph	211	0	.notdef
154	0	.notdef	183	116	bullet	212	0	.notdef
155	0	.notdef	184	117	quotesinglbase	213	0	.notdef
156	0	.notdef	185	118	quotedblbase	214	0	.notdef
157	0	.notdef	186	119	quotedblright	215	0	.notdef
158	0	.notdef	187	120	guillemotright	216	0	.notdef
159	0	.notdef	188	121	ellipsis	217	0	.notdef
160	0	.notdef	189	122	perthousand	218	0	.notdef
161	96	exclamdown	190	0	.notdef	219	0	.notdef
162	97	cent	191	123	questiondown	220	0	.notdef
163	98	sterling	192	0	.notdef	221	0	.notdef
164	99	fraction	193	124	grave	222	0	.notdef
165	100	yen	194	125	acute	223	0	.notdef
166	101	florin	195	126	circumflex	224	0	.notdef
167	102	section	196	127	tilde	225	138	AE
168	103	currency	197	128	macron	226	0	.notdef
169	104	quotesingle	198	129	breve	227	139	ordfeminine
170	105	quotedblleft	199	130	dotaccent	228	0	.notdef
171	106	guillemotleft	200	131	dieresis	229	0	.notdef
172	107	guilsinglleft	201	0	.notdef	230	0	.notdef
173	108	guilsinglright	202	132	ring	231	0	.notdef
174	109	fi	203	133	cedilla	232	140	Lslash
175	110	fl	204	0	.notdef	233	141	Oslash
176	0	.notdef	205	134	hungarumlaut	234	142	OE
177	111	endash	206	135	ogonek	235	143	ordmasculine
178	112	dagger	207	136	caron	236	0	.notdef

237	0	.notdef	244	0	.notdef	251	149	germandbls
238	0	.notdef	245	145	dotlessi	252	0	.notdef
239	0	.notdef	246	0	.notdef	253	0	.notdef
240	0	.notdef	247	0	.notdef	254	0	.notdef
241	144	ae	248	146	lslash	255	0	.notdef
242	0	.notdef	249	147	oslash			
243	0	.notdef	250	148	oe			

ExpertEncoding (code / SID / name)

0	0	.notdef	18	0	.notdef	36	231	dollaroldstyle
1	0	.notdef	19	0	.notdef	37	232	dollarsuperior
2	0	.notdef	20	0	.notdef	38	233	ampersandsmall
3	0	.notdef	21	0	.notdef	39	234	Acutesmall
4	0	.notdef	22	0	.notdef	40	235	parenleftsuperior
5	0	.notdef	23	0	.notdef	41	236	parenrightsuperior
6	0	.notdef	24	0	.notdef	42	237	twodotenleader
7	0	.notdef	25	0	.notdef	43	238	onedotenleader
8	0	.notdef	26	0	.notdef	44	13	comma
9	0	.notdef	27	0	.notdef	45	14	hyphen
10	0	.notdef	28	0	.notdef	46	15	period
11	0	.notdef	29	0	.notdef	47	99	fraction
12	0	.notdef	30	0	.notdef	48	239	zerooldstyle
13	0	.notdef	31	0	.notdef	49	240	oneoldstyle
14	0	.notdef	32	1	space	50	241	twooldstyle
15	0	.notdef	33	229	exclamsmall	51	242	threeoldstyle
16	0	.notdef	34	230	Hungarumlautsmall	52	243	fouroldstyle
17	0	.notdef	35	0	.notdef	53	244	fiveoldstyle

54	245	sixoldstyle		83	264	ssuperior		112	289	Psmall
55	246	sevenoldstyle		84	265	tsuperior		113	290	Qsmall
56	247	eightoldstyle		85	0	.notdef		114	291	Rsmall
57	248	nineoldstyle		86	266	ff		115	292	Ssmall
58	27	colon		87	109	fi		116	293	Tsmall
59	28	semicolon		88	110	fl		117	294	Usmall
60	249	commasuperior		89	267	ffi		118	295	Vsmall
61	250	threequartersemdash		90	268	ffl		119	296	Wsmall
62	251	periodsuperior		91	269	parenleftinferior		120	297	Xsmall
63	252	questionsmall		92	0	.notdef		121	298	Ysmall
64	0	.notdef		93	270	parenrightinferior		122	299	Zsmall
65	253	asuperior		94	271	Circumflexsmall		123	300	colonmonetary
66	254	bsuperior		95	272	hyphensuperior		124	301	onefitted
67	255	centsuperior		96	273	Gravesmall		125	302	rupiah
68	256	dsuperior		97	274	Asmall		126	303	Tildesmall
69	257	esuperior		98	275	Bsmall		127	0	.notdef
70	0	.notdef		99	276	Csmall		128	0	.notdef
71	0	.notdef		100	277	Dsmall		129	0	.notdef
72	0	.notdef		101	278	Esmall		130	0	.notdef
73	258	isuperior		102	279	Fsmall		131	0	.notdef
74	0	.notdef		103	280	Gsmall		132	0	.notdef
75	0	.notdef		104	281	Hsmall		133	0	.notdef
76	259	lsuperior		105	282	Ismall		134	0	.notdef
77	260	msuperior		106	283	Jsmall		135	0	.notdef
78	261	nsuperior		107	284	Ksmall		136	0	.notdef
79	262	osuperior		108	285	Lsmall		137	0	.notdef
80	0	.notdef		109	286	Msmall		138	0	.notdef
81	0	.notdef		110	287	Nsmall		139	0	.notdef
82	263	rsuperior		111	288	Osmall		140	0	.notdef

141	0	.notdef		170	311	Caronsmall		199	0	.notdef
142	0	.notdef		171	0	.notdef		200	326	zerosuperior
143	0	.notdef		172	312	Dotaccentsmall		201	150	onesuperior
144	0	.notdef		173	0	.notdef		202	164	twosuperior
145	0	.notdef		174	0	.notdef		203	169	threesuperior
146	0	.notdef		175	313	Macronsmall		204	327	foursuperior
147	0	.notdef		176	0	.notdef		205	328	fivesuperior
148	0	.notdef		177	0	.notdef		206	329	sixsuperior
149	0	.notdef		178	314	figuredash		207	330	sevensuperior
150	0	.notdef		179	315	hypheninferior		208	331	eightsuperior
151	0	.notdef		180	0	.notdef		209	332	ninesuperior
152	0	.notdef		181	0	.notdef		210	333	zeroinferior
153	0	.notdef		182	316	Ogoneksmall		211	334	oneinferior
154	0	.notdef		183	317	Ringsmall		212	335	twoinferior
155	0	.notdef		184	318	Cedillasmall		213	336	threeinferior
156	0	.notdef		185	0	.notdef		214	337	fourinferior
157	0	.notdef		186	0	.notdef		215	338	fiveinferior
158	0	.notdef		187	0	.notdef		216	339	sixinferior
159	0	.notdef		188	158	onequarter		217	340	seveninferior
160	0	.notdef		189	155	onehalf		218	341	eightinferior
161	304	exclamdownsmall		190	163	threequarters		219	342	nineinferior
162	305	centoldstyle		191	319	questiondownsmall		220	343	centinferior
163	306	Lslashsmall		192	320	oneeighth		221	344	dollarinferior
164	0	.notdef		193	321	threeeighths		222	345	periodinferior
165	0	.notdef		194	322	fiveeighths		223	346	commainferior
166	307	Scaronsmall		195	323	seveneighths		224	347	Agravesmall
167	308	Zcaronsmall		196	324	onethird		225	348	Aacutesmall
168	309	Dieresissmall		197	325	twothirds		226	349	Acircumflexsmall
169	310	Brevesmall		198	0	.notdef		227	350	Atildesmall

228	351	Adieresissmall	238	361	Icircumflexsmall	248	371	Oslashsmall
229	352	Aringsmall	239	362	Idieresissmall	249	372	Ugravesmall
230	353	AEsmall	240	363	Ethsmall	250	373	Uacutesmall
231	354	Ccedillasmall	241	364	Ntildesmall	251	374	Ucircumflexsmall
232	355	Egravesmall	242	365	Ogravesmall	252	375	Udieresissmall
233	356	Eacutesmall	243	366	Oacutesmall	253	376	Yacutesmall
234	357	Ecircumflexsmall	244	367	Ocircumflexsmall	254	377	Thornsmall
235	358	Edieresissmall	245	368	Otildesmall	255	378	Ydieresissmall
236	359	Igravesmall	246	369	Odieresissmall			
237	360	Iacutesmall	247	370	OEsmall			

Appendix C
Predefined Charsets

All charsets are presented in GID order beginning with GID 1. (The .notdef glyph is implicitly GID 0 and is therefore not shown.)

ISOAdobe (SID / name)

1	space	21	four	41	H
2	exclam	22	five	42	I
3	quotedbl	23	six	43	J
4	numbersign	24	seven	44	K
5	dollar	25	eight	45	L
6	percent	26	nine	46	M
7	ampersand	27	colon	47	N
8	quoteright	28	semicolon	48	O
9	parenleft	29	less	49	P
10	parenright	30	equal	50	Q
11	asterisk	31	greater	51	R
12	plus	32	question	52	S
13	comma	33	at	53	T
14	hyphen	34	A	54	U
15	period	35	B	55	V
16	slash	36	C	56	W
17	zero	37	D	57	X
18	one	38	E	58	Y
19	two	39	F	59	Z
20	three	40	G	60	bracketleft

61	backslash	93	bar	125	acute
62	bracketright	94	braceright	126	circumflex
63	asciicircum	95	asciitilde	127	tilde
64	underscore	96	exclamdown	128	macron
65	quoteleft	97	cent	129	breve
66	a	98	sterling	130	dotaccent
67	b	99	fraction	131	dieresis
68	c	100	yen	132	ring
69	d	101	florin	133	cedilla
70	e	102	section	134	hungarumlaut
71	f	103	currency	135	ogonek
72	g	104	quotesingle	136	caron
73	h	105	quotedblleft	137	emdash
74	i	106	guillemotleft	138	AE
75	j	107	guilsinglleft	139	ordfeminine
76	k	108	guilsinglright	140	Lslash
77	l	109	fi	141	Oslash
78	m	110	fl	142	OE
79	n	111	endash	143	ordmasculine
80	o	112	dagger	144	ae
81	p	113	daggerdbl	145	dotlessi
82	q	114	periodcentered	146	lslash
83	r	115	paragraph	147	oslash
84	s	116	bullet	148	oe
85	t	117	quotesinglbase	149	germandbls
86	u	118	quotedblbase	150	onesuperior
87	v	119	quotedblright	151	logicalnot
88	w	120	guillemotright	152	mu
89	x	121	ellipsis	153	trademark
90	y	122	perthousand	154	Eth
91	z	123	questiondown	155	onehalf
92	braceleft	124	grave	156	plusminus

157	Thorn	181	Egrave	205	atilde
158	onequarter	182	Iacute	206	ccedilla
159	divide	183	Icircumflex	207	eacute
160	brokenbar	184	Idieresis	208	ecircumflex
161	degree	185	Igrave	209	edieresis
162	thorn	186	Ntilde	210	egrave
163	threequarters	187	Oacute	211	iacute
164	twosuperior	188	Ocircumflex	212	icircumflex
165	registered	189	Odieresis	213	idieresis
166	minus	190	Ograve	214	igrave
167	eth	191	Otilde	215	ntilde
168	multiply	192	Scaron	216	oacute
169	threesuperior	193	Uacute	217	ocircumflex
170	copyright	194	Ucircumflex	218	odieresis
171	Aacute	195	Udieresis	219	ograve
172	Acircumflex	196	Ugrave	220	otilde
173	Adieresis	197	Yacute	221	scaron
174	Agrave	198	Ydieresis	222	uacute
175	Aring	199	Zcaron	223	ucircumflex
176	Atilde	200	aacute	224	udieresis
177	Ccedilla	201	acircumflex	225	ugrave
178	Eacute	202	adieresis	226	yacute
179	Ecircumflex	203	agrave	227	ydieresis
180	Edieresis	204	aring	228	zcaron

Expert (SID / name)

1	space	233	ampersandsmall	238	onedotenleader
229	exclamsmall	234	Acutesmall	13	comma
230	Hungarumlautsmall	235	parenleftsuperior	14	hyphen
231	dollaroldstyle	236	parenrightsuperior	15	period
232	dollarsuperior	237	twodotenleader	99	fraction

239	zerooldstyle	267	ffi	299	Zsmall
240	oneoldstyle	268	ffl	300	colonmonetary
241	twooldstyle	269	parenleftinferior	301	onefitted
242	threeoldstyle	270	parenrightinferior	302	rupiah
243	fouroldstyle	271	Circumflexsmall	303	Tildesmall
244	fiveoldstyle	272	hyphensuperior	304	exclamdownsmall
245	sixoldstyle	273	Gravesmall	305	centoldstyle
246	sevenoldstyle	274	Asmall	306	Lslashsmall
247	eightoldstyle	275	Bsmall	307	Scaronsmall
248	nineoldstyle	276	Csmall	308	Zcaronsmall
27	colon	277	Dsmall	309	Dieresissmall
28	semicolon	278	Esmall	310	Brevesmall
249	commasuperior	279	Fsmall	311	Caronsmall
250	threequartersemdash	280	Gsmall	312	Dotaccentsmall
251	periodsuperior	281	Hsmall	313	Macronsmall
252	questionsmall	282	Ismall	314	figuredash
253	asuperior	283	Jsmall	315	hypheninferior
254	bsuperior	284	Ksmall	316	Ogoneksmall
255	centsuperior	285	Lsmall	317	Ringsmall
256	dsuperior	286	Msmall	318	Cedillasmall
257	esuperior	287	Nsmall	158	onequarter
258	isuperior	288	Osmall	155	onehalf
259	lsuperior	289	Psmall	163	threequarters
260	msuperior	290	Qsmall	319	questiondownsmall
261	nsuperior	291	Rsmall	320	oneeighth
262	osuperior	292	Ssmall	321	threeeighths
263	rsuperior	293	Tsmall	322	fiveeighths
264	ssuperior	294	Usmall	323	seveneighths
265	tsuperior	295	Vsmall	324	onethird
266	ff	296	Wsmall	325	twothirds
109	fi	297	Xsmall	326	zerosuperior
110	fl	298	Ysmall	150	onesuperior

164	twosuperior	343	centinferior	361	Icircumflexsmall
169	threesuperior	344	dollarinferior	362	Idieresissmall
327	foursuperior	345	periodinferior	363	Ethsmall
328	fivesuperior	346	commainferior	364	Ntildesmall
329	sixsuperior	347	Agravesmall	365	Ogravesmall
330	sevensuperior	348	Aacutesmall	366	Oacutesmall
331	eightsuperior	349	Acircumflexsmall	367	Ocircumflexsmall
332	ninesuperior	350	Atildesmall	368	Otildesmall
333	zeroinferior	351	Adieresissmall	369	Odieresissmall
334	oneinferior	352	Aringsmall	370	OEsmall
335	twoinferior	353	AEsmall	371	Oslashsmall
336	threeinferior	354	Ccedillasmall	372	Ugravesmall
337	fourinferior	355	Egravesmall	373	Uacutesmall
338	fiveinferior	356	Eacutesmall	374	Ucircumflexsmall
339	sixinferior	357	Ecircumflexsmall	375	Udieresissmall
340	seveninferior	358	Edieresissmall	376	Yacutesmall
341	eightinferior	359	Igravesmall	377	Thornsmall
342	nineinferior	360	Iacutesmall	378	Ydieresissmall

Expert Subset (SID / name)

1	space	239	zerooldstyle	28	semicolon
231	dollaroldstyle	240	oneoldstyle	249	commasuperior
232	dollarsuperior	241	twooldstyle	250	threequartersemdash
235	parenleftsuperior	242	threeoldstyle	251	periodsuperior
236	parenrightsuperior	243	fouroldstyle	253	asuperior
237	twodotenleader	244	fiveoldstyle	254	bsuperior
238	onedotenleader	245	sixoldstyle	255	centsuperior
13	comma	246	sevenoldstyle	256	dsuperior
14	hyphen	247	eightoldstyle	257	esuperior
15	period	248	nineoldstyle	258	isuperior
99	fraction	27	colon	259	lsuperior

260	msuperior	314	figuredash	330	sevensuperior
261	nsuperior	315	hypheninferior	331	eightsuperior
262	osuperior	158	onequarter	332	ninesuperior
263	rsuperior	155	onehalf	333	zeroinferior
264	ssuperior	163	threequarters	334	oneinferior
265	tsuperior	320	oneeighth	335	twoinferior
266	ff	321	threeeighths	336	threeinferior
109	fi	322	fiveeighths	337	fourinferior
110	fl	323	seveneighths	338	fiveinferior
267	ffi	324	onethird	339	sixinferior
268	ffl	325	twothirds	340	seveninferior
269	parenleftinferior	326	zerosuperior	341	eightinferior
270	parenrightinferior	150	onesuperior	342	nineinferior
272	hyphensuperior	164	twosuperior	343	centinferior
300	colonmonetary	169	threesuperior	344	dollarinferior
301	onefitted	327	foursuperior	345	periodinferior
302	rupiah	328	fivesuperior	346	commainferior
305	centoldstyle	329	sixsuperior		

Appendix D
Example CFF Font

This appendix illustrates the CFF format with an example font. The font shown is a subset with just the .notdef and space glyphs of the Times* font program that has been renamed. This font has no subrs and uses predefined encoding and charset.

Binary dump (147 bytes):

```
0000000   0100 0401 0001 0101 1341 4243 4445 462b   |?.??.????ABCDEF+|
0000010   5469 6d65 732d 526f 6d61 6e00 0101 011f   |Times-Roman.????|
0000020   f81b 00f8 1c02 f81d 03f8 1904 1c6f 000d   |??.??????????o.?|
0000030   fb3c fb6e fa7c fa16 05e9 11b8 f112 0003   |?<?n?|????????.?|
0000040   0101 0813 1830 3031 2e30 3037 5469 6d65   |?????001.007Time|
0000050   7320 526f 6d61 6e54 696d 6573 0000 0002   |s RomanTimes...?|
0000060   0101 0203 0e0e 7d99 f92a 99fb 7695 f773   |??????}??*??v??s|
0000070   8b06 f79a 93fc 7c8c 077d 99f8 5695 f75e   |??????|??}??V??^|
0000080   9908 fb6e 8cf8 7393 f710 8b09 a70a df0b   |???n??s?????????|
0000090   f78e 14                                    |???             |
```

Annotated dump:

```
### Header (00000000)
major  =1
minor  =0
hdrSize=4
offSize=1
### Name INDEX (00000004)
count  =1
offSize=1
--- offset[index]=value
[0]=1 [1]=19
--- object[index]=<value>
[0]=<ABCDEF+Times-Roman>
### Top DICT INDEX (0000001b)
count  =1
offSize=1
--- offset[index]=value
[0]=1 [1]=31
--- object[index]=<value>
[0]=<391 version 392 FullName 393 FamilyName 389 Weight 28416
UniqueID -168 -218 1000 898 FontBBox 94 CharStrings 45 102 Private>
### String INDEX (0000003e)
count  =3
offSize=1
--- offset[index]=value
[0]=1 [1]=8 [2]=19 [3]=24
--- object[index]=<value>
[0]=<001.007> [1]=<Times Roman> [2]=<Times>
### Global Subrs INDEX (0000005c)
count  =0
### CharStrings INDEX (0000005e)
count  =2
offSize=1
--- offset[index]=value
[0]=1 [1]=2 [2]=3
--- object[index]=<value>
[0]=<endchar> [1]=<endchar>
### Private DICT (00000066)
-14 14 662 14 -226 10 223 0 BlueValues 262 8 -488 1 OtherBlues
-14 14 450 10 202 14 FamilyBlues -218 1 479 8 124 0 FamilyOtherBlues
28 StdHW 84 StdVW 250 defaultWidthX
```

Appendix E
PostScript File Structure

CFF data is enclosed in a wrapper and treated as a file when used by a PostScript interpreter that supports the FontSet resource. The wrapper allows the file to be directly executed by the interpreter, and permits insertion or removal by a DSC (Document Structuring Convention)-aware driver. A well-formatted FontSet file has the following structure:

```
%!PS-Adobe-3.0 Resource-FontSet
%%DocumentNeededResources: ProcSet (FontSetInit)
%%EndComments
%%IncludeResource: ProcSet (FontSetInit)
%%BeginResource: FontSet (<fontsetname>)
%%Title: (FontSet/<fontsetname>)
%%Version: <realversion> [<intrevision>]
/FontSetInit /ProcSet findresource begin
%%BeginData: <m> Binary Bytes
/<fontsetname> <n> StartData<space><n-binary-data-bytes>
%%EndData
%%EndResource
%%EOF
```

A single space (0x20) follows the StartData procedure call which is followed immediately by <n> binary data bytes. Note that the %%EndData DSC comment must be preceded by a newline character (that is not part of the binary data) so that it is correctly parsed. The count <m> in the %%BeginData: DSC is greater than that used by the StartData procedure call by the number of characters in the call itself.

All the (0) offsets in the <binary data> are relative to the start of the <binary data>. That is, the byte following the space that terminates StartData is numbered 0.

Appendix F
Embedded PostScript

A Top DICT may contain at most one embedded PostScript operator. It must be in the main font dictionary, not in the Private dictionary. If present, this string is executed after the font dictionary has been completely constructed, but before definefont. At that time, the interpreter:

- Pushes the top-level font dictionary (that is under construction) on the dictionary stack;

- Executes the PostScript string;

- Pops the dictionary stack.

At the time of execution, the FontInfo, Encoding, CharStrings, and all other elements of the font dictionary have been defined, including ones that are set to default values. (There is no Private dictionary, however. In CFF, the Private dictionary is never processed as PostScript, so there is no opportunity for embedded PostScript to alter it.)

A CFF CIDFont may contain an embedded PostScript operator in the Top DICT or in any FDs.

If it is in the Top DICT, the embedded PostScript string is processed as described above. This occurs after the font dictionary has been completely constructed (including all FDArray sub-dictionaries), but before definefont.

If it is in one of the nested sub-dictionaries in the FDArray, it is executed after the sub-dictionary has been completely constructed (including default values), but before it has been incorporated as an element of the main font dictionary. The sub-dictionary is on the dictionary stack; there is no way to access the main dictionary.

A CFF consumer that does not interpret PostScript can ignore the embedded PostScript string. The font should still work, but without the feature that the embedded PostScript would have activated.

Appendix G
Related Documentation

The following documents may be consulted for further information on Adobe font technology. All are available at http://partners.adobe.com/asn/developer/technotes.html.

- Adobe Type 1 Font Format. Addison-Wesley, 1991; ISBN 0-201-57044-0.

- PostScript Language Reference, Third Edition. Addison-Wesley, 1999; ISBN 0-201-37922-8.

- Technical Note #5014: "Adobe CMap and CIDFont Files Specification."

- Technical Note #5015: "The Type 1 Font Format Supplement." This document contains all updates to the Type 1 format.

- Technical Note #5040: "Supporting Downloadable PostScript Fonts." Describes how Type 1 fonts have traditionally been packaged for use in the Macintosh® and Windows® environments – specifically, the use of POST resources for Macintosh fonts and the PFB compressed binary format for Windows fonts.

- Technical Note #5088: "Font Naming Issues." In addition to a discussion of general font name issues, this document explains the naming conventions for multiple master fonts.

- Technical Note #5092: "CID-Keyed Font File Format Overview."

- Technical Note #5177: "Type 2 Charstring Format."

Appendix H
CFF DICT Encoding

One-byte CFF DICT Operators

Dec	Hex	Operator	Dec	Hex	Operator
0	00	version	18	12	Private
1	01	Notice	19	13	Subrs
2	02	FullName	20	14	defaultWidthX
3	03	FamilyName	21	15	nominalWidthX
4	04	Weight	22	16	-Reserved-
5	05	FontBBox	23	17	-Reserved-
6	06	BlueValues	24	18	-Reserved-
7	07	OtherBlues	25	19	-Reserved-
8	08	FamilyBlues	26	1a	-Reserved-
9	09	FamilyOtherBlues	27	1b	-Reserved-
10	0a	StdHW	28[2]	1c	shortint
11	0b	StdVW	29	1d	longint
12[1]	0c	escape	30	1e	BCD
13	0d	UniqueID	31	1f	-Reserved-
14	0e	XUID	32–246	20–f6	<numbers>
15	0f	charset	247–254[3]	f7–fe	<numbers>
16	10	Encoding	255	ff	-Reserved-
17	11	CharStrings			

1. First byte of a 2-byte operator.
2. First byte of a 3-byte sequence specifying a number.
3. First byte of a 2-byte sequence specifying a number.

Two-byte CFF DICT Operators

Dec	Hex	Operator	Dec	Hex	Operator
12 0	0c 00	Copyright	12 21	0c 15	PostScript
12 1	0c 01	isFixedPitch	12 22	0c 16	BaseFontName
12 2	0c 02	ItalicAngle	12 23	0c 17	BaseFontBlend
12 3	0c 03	UnderlinePosition	12 24	0c 18	-Reserved-
12 4	0c 04	UnderlineThickness	12 25	0c 19	-Reserved-
12 5	0c 05	PaintType	12 26	0c 1a	-Reserved-
12 6	0c 06	CharstringType	12 27	0c 1b	-Reserved-
12 7	0c 07	FontMatrix	12 28	0c 1c	-Reserved-
12 8	0c 08	StrokeWidth	12 29	0c 1d	-Reserved-
12 9	0c 09	BlueScale	12 30	0c 1e	ROS
12 10	0c 0a	BlueShift	12 31	0c 1f	CIDFontVersion
12 11	0c 0b	BlueFuzz	12 32	0c 20	CIDFontRevision
12 12	0c 0c	StemSnapH	12 33	0c 21	CIDFontType
12 13	0c 0d	StemSnapV	12 34	0c 22	CIDCount
12 14	0c 0e	ForceBold	12 35	0c 23	UIDBase
12 15	0c 0f	-Reserved-	12 36	0c 24	FDArray
12 16	0c 10	-Reserved-	12 37	0c 25	FDSelect
12 17	0c 11	LanguageGroup	12 38	0c 26	FontName
12 18	0c 12	ExpansionFactor	12 39– 12 255	0c 27– 0c ff	-Reserved-
12 19	0c 13	initialRandomSeed			
12 20	0c 14	SyntheticBase			

Appendix I
Changes Since Earlier Versions

The following changes and revisions have been made since the initial publication date of 18 November 1996.

Changes in the 16 March 2000 document

- Removed all material about multiple master and chameleon fonts.

- Section 4, *DICT Data* – specified the order of nibble data stored in a byte; also specified the range of values in the first byte of an operator or operand that can be used to distinguish between operators and operands.

- Section 12, *Encodings* – clarified that glyphs not specified by custom or predefined encodings are, by definition, unencoded.

- Section 15, *Private DICT Data* – added statement that Private DICT is required, but may be specified as having 0 length.

- Section 15, *Private DICT Data* – removed entries for *ForceBoldThreshhold* and *lenIV* from Table 23.

- Section 17, *PS File Structure* – moved this section to Appendix E; the previous Appendix E and all subsequent appendices moved accordingly.

- Section 18, *Copyright and Trademark Notices* – removed entire section.

- Section 22, *FDSelect* – added clarification of the values of GIDs.

- Added appendix summarizing CFF DICT encoding.

Note   *The version number of the format has not changed for this revision.*

Changes in the 18 March 1998 document

- Table 25, it was noted that the only value supported for lenIV is –1.

- Section 13, *Charsets* – statement added that CID fonts must not use predefined charsets.

- Section 22, *FDSelect* – first paragraph; the FD index is used to access one of the Font DICTs stored in the Font DICT INDEX (not the previously stated "FDArray").

Changes in the 15 October 1997 document

- Minor changes were made to clarify existing text.

- In Table 2, the string id range was changed to 0–64999.

- In Table 9, the default for the **FontBBox** array was specified as "0 0 0 0".

- The specification of multiple master fonts was changed in several sections.

- Section 17, *PostScript File Structure* – corrections made to DSC comments.

- Section 19, *Synthetic Fonts* – new capability added to specify encodings for synthetic fonts.

- Appendix D, *Example CFF Font* – corrected operator name in example; *FamilyBlueValues* changed to *FamilyBlues*.

Changes in the 16 December 1996 document

A variety of minor changes were made to clarify existing text; the technical content was not affected.