

*Developer and Licensee Technical Support Documentation*

**PFS-102**

**Character Name Designation Methodology  
in Production First Software Fonts**

**November 11, 1994**

**Production First Software**

P.O.Box 31528 • San Francisco, CA 94131-0528 • (415) 431-FONT(-3668)

## 1.0 Introduction

Character and character glyph names are important for at least two reasons:

- 1) They provide human-readable data tags which help to provide more immediate and direct human access to data, and also are useful in documentation of data.
- 2) They are required in the architecture and management of PostScript and PostScript fonts. Names are used in encoding, composition and decomposition (for composite characters), data sorting, and other uses.

There are many ways from which to derive a complete set of character names or a procedure to derive them:

- 1) Preëxisting sets of character names which have been in wide use since PostScript was first introduced (“legacy names”);
- 2) ISO standards relating to character descriptions or names (e.g. SGML standard names in ISO 8879, or ISO-originated names in ISO 10036);
- 3) Algorithmically-derived character names;
- 4) Names based on a catalog or registry (e.g. IBM standard names, names derived from the AFII glyph register, names derived from an all-encompassing character set or encoding like ISO 10646);
- 5) Names derived from a meaningful sequence of numbers (e.g. from 0 to  $16^{N+1}$ ); and
- 6) possibly other ways, including a hierarchical combination of the above.

This report describes the principles leading to the convention used to name characters and character glyphs for Production First Software fonts created from 1990 to the present, and the reasons for using it. It should be noted that the ISO 8879 SGML character names are not consistent with the goals and principles mentioned above. Also, the ISO 10036 standard lists character description names (many of which are several words long), are not suitable character names due to their length, descriptive inconsistency, algorithmic irregularity, and inconsistency with legacy names. Note: at the time this report was prepared, Unicode 1.1 was being introduced. This was the first version of Unicode conformant with ISO 10646. Previous versions were not conformant. Whether or not future versions will continue to be conformant is another question, because the time required to adopt amendments or changes to Unicode is much shorter than the time required for a corresponding process at ISO.

It should be emphasized that the naming rules and principles described in this report were designed after careful consideration of the ramifications and needs of nationalism, but counterbalanced with the fact that modern computer technology was originated and evolved in the English-speaking world. This is particularly sensitive in the case of character naming for non-Latin (Roman) scripts, since an alphabet letterform name may refer to one character glyph shape (and character) in one country but a different glyph shape and character in another country. While not present in the Latin script, this problem does arise in Arabic, Cyrillic, Greek, and other scripts. This sensitivity may also have influenced the evolution of the multibyte encoding standards like Unicode and ISO 10646.

It has been proposed by some developers that AFII (Association for Font Information Interchange) Glyph Register signature numbers be used to identify glyphs for the case where a numerically-derived name is desired. This would be implemented as 'AFII\_ \_ \_ \_ \_' where 9 decimal characters could be used. Production First Software does not use AFII-based descriptors because:

- 1) The names cannot span a 32-bit code space inherent in ISO 10646 (over 4 million numbers) using 5 decimal digits.
- 2) The names are too long, especially if AFII names are used for a base character and one or more diacritical marks constituting a composite character which is not already in the Glyph Register.
- 3) It is conceivable that cases could arise where different character names could be used which refer to the same character glyph. Such a case would be where a new composite character name uses AFII

names for the glyph components, and then at a later time, that new composite is registered so as to be referable using a single AFII name. Another case is where consistent English names can be constructed or assigned, but an AFII name could also be used.

4) All variants (due to script, style within same script, combining characters, non-combining characters, true alternates, set position, or scale) are assignable to the AFII Glyph Register. Therefore, the register is like a Minestrone soup, some ingredients of which represent style variations (which should logically be classified as font variants) and some of which do not. What agency arbitrates whether or not an AFII-derived name, or which AFII-derived name if there is more than one possible, is the “official” name for a character or glyph?

5. AFII-derived names are not immediately human-understandable names, and this could become a very significant disadvantage.

## 2.0 Character and Character Glyph Name Requirements

Note: both characters and glyphs can have assignable names. Production First Software glyph names are most of the time a subset of the set of character names, exceptions existing for Arabic and Indic scripts where there are glyph names which do not necessarily match corresponding character names due to the complex contextual relationships of the scripts. However, for the sake of textual brevity, the term “character name” will also be understood to apply to “glyph name” except for specific cases which will be identified when they arise.

The naming convention used for Production First Software fonts is a hierarchical collection of naming rules. The following major principles apply:

I. Names must be constructed for certain specific classes of characters:

- 1) composite characters (base character + ... + base character + diacritical mark + ... + diacritical mark);
- 2) characters composed from combinations of other base characters (ligatures, tied letters, N-graphs);
- 3) characters with physical descriptors as part of their name (connoting some aspect of shape, direction, or placement);
- 4) characters with descriptors connoting alternate styles of functionally identical characters.

II. Composite character names must be created from existing base character names and diacritical marks, with ISOLatin-1 and legacy names given preference. A hierarchical stacking order is generally evoked in constructing composite character names.

III. Names must be parseable if possible, so as to be able to extract descriptive information usable for categorization, character substitution, and post-creation complex or composite synthetic on-the-fly construction.

IV. Except for ISOLatin-1 and legacy names, a base character name particle precedes adjectives and adverbs which describe it in the formation of a name, so as to enable similar names to sort together.

V. Complicated composite names must possess a commutative property for glyph names which are used to derive the full name or to be usable in synthetic on-the-fly construction.

VI. Qualifier descriptors are defined so as to be backward compatible and consistent with as many widely-used ISOLatin-1 and legacy character names as possible.

VII. Cultural, geographic, or linguistic name variants or name components and transliterated or Latin-phonetized non-Latin name components are to be avoided unless they have been in common typographic use (like the letterforms of the Greek alphabet), with two exceptions. (Note 1, Note 2)

VIII. Subject to VII, non-numerically derived alphabetic name components must be in English.

IX. Names used for diacritical marks are to be spelled out in transliterated English in the native language in which they are first commonly used in typography, provided that language uses the Latin script. If the language does not use the Latin script, a numerically-derived name is to be used. This was actually the method used to name many legacy-named diacritical marks. For example, the Vietnamese tone mark has a character name 'hoi' because Latin script is used to spell modern Vietnamese, and some of the other tone marks used in Vietnamese are represented by diacritical marks already coded in ISO



## I. Base Character, Script, Style

The base character name can either be a name spelled in English for in the primary script or a numerically-derived name (starting with a Latin letter). The script qualifier is absent except for characters which require it for distinction. This qualifier should be used judiciously to avoid a situation where cross-substitution between scripts occurs.

Script qualifiers in use include:

'armenian' - Armenian ('arm' and 'armen' pose conflicts with preexisting names and name qualifiers)

'cyr' - Cyrillic-derived base characters

'greek' - Greek punctuation marks

'latin' - base characters of Latin script

'phon' - IPA extension base characters

'viet' - Vietnamese clone of otherwise identical diacritical marks but specially positioned so as to form diacritical mark ligatures.

Style qualifiers in use include:

'APL' - for a specific glyph designed specifically for APL programming language

'cursive' - for a specific glyph constrained to that style independent of the actual typeface design

'dblstruck' - for a specific glyph constrained to that style independent of the actual typeface design

'fraktur' - for a specific glyph constrained to that style independent of the actual typeface design

'mod' - IPA modifier letterform

'oldstyle' - antique alternate form which resides in the same font

'roman' - for a specific glyph designed to represent Roman numerals

'script' - for a specific glyph constrained to that style independent of the actual typeface design

'swash' - for a specific glyph constrained to that style independent of the actual typeface design

Many primary descriptor qualifiers are in use. Examples include 'circled', 'head', 'logo', 'parallel'

## II. Appendages

“—” 'cross', 'crosslong' - used to designate a representation of a horizontal stroke (Note 4)

“/” 'slash', 'slashmark' - used to designate a representation of a diagonal stroke (Note 4)

“\” 'backslash' - used to designate a representation of a backward-inclined stroke

“|” 'bar'

The horizontal mark is called a 'cross' from the term “crossbrace” and “crossing a tee.”

To distinguish a horizontal from a vertical mark, the vertical mark is called a 'bar' from the consideration that the typical characterization of a jail cell has predominantly vertical spaced steel members blocking all openings, and a criminal in jail is said to be “behind bars.”

A 'horn' is a comma-like appendage which attaches at its lowest point to a position near the top of a character bowl.

A 'hook' is an appendage which attaches either to a stroke or to a bowl, but is longer than a 'horn'.

## III. Quantity Qualifiers

'single' (last qualifier)

'singl' (additional qualifiers follow) - for “single”

'dbl' - for “double”

'tripl' - for “triple”

'quadrupl' for “quadruple”

## V. Image (more significant, less significant)

'encl' - “enclosing”; the glyph image encloses the glyph of the base character.

'inv' - “inverted”; a glyph image turned upside down.

'refl' - “reflected”; the glyph image as reflected in a horizontal mirror or pool of water. This is equivalent to “inverted reversed.” A synonym for “flipped vertical.”

'rev' - “reversed”; a synonym for “flipped horizontally.”

## V. Appendage Modifiers

'hookpal' and 'hookretro' are alterations to the most common 'hook' appendage. Adjectives such as 'pal' and 'retro' (as well as others such as 'mark', etc.) constitute “appendage modifiers” and follow the particle 'hook' so that a sort made on <base character>hook yields all variants.

## VI. Size Qualifiers

'baby' - a glyph at lowercase height but compressed further so as to exclude anything below the under-shoot

'large' - a lowercase-like letterform scaled to upper case height

'long' - a vertically elongated design

'longleftleg' - a glyph with an extended left leg

'longrightleg' - a glyph with an extended right leg

'small' - capital-like letterform scaled down to lower case height

## VII. Set Position

'above' - above x-height for a combining character

'below' - below baseline for a combining character

'higher' - partially above x-height for a combining character but higher than 'super'

'inferior' - partially below baseline for a base character

'lower' - partially below baseline for a combining character but lower than 'sub'

'sub' - partially below baseline for a combining character

'super' - partially above x-height for a combining character

'superior' - partially above x-height for a base character

'top' - above or near Capheight

The most common set position of a glyph need not include a set position component to the name. This rule is being adopted to improve and assure compatibility with many legacy names.

## VIII. Contextual Behavior Qualifiers

'initial' - directionally first character in word

'medial' - interior character in word

'final' - directionally last character in word

'stackup' - stack above previous character

'stackbelow' - stack below previous character

These qualifiers are used only if the glyph name does not otherwise distinguish contextual behavior.

Note: isolate behavior (a glyph form not altered by contextual behavior) does not use a qualifier

## IX. Color Qualifiers

'black' - general fill or process color

'cyan' - process color

'magenta' - process color

'yellow' - process color

## X. Spacing Behavior Qualifiers

'nosp' - non-spacing

'space' - indicates a type of space character

### 5.0 ISOLatin-1 and Legacy Names

A number of name exceptions arise because of legacy names. These must be handled in a manner which would not exclude those names from being correctly handled in future fonts. This can be done in two ways:

- 1) use the legacy name as the primary name even if it violates some of the rules and principles described above, as long as an ambiguity does not arise; and
- 2) ignore the legacy name in the case of an ambiguity and go with a new name.

Some of the most important exceptions are described below. This is by no means an exhaustive list.

#### 1) 'micron' at <00b5>

'micron' used to resolve conflict with Greek 'mu'. A micron is a recognized unit of length used in astrophysics and molecular biology, and is a useful name; and because character names should not be prefixes. The name 'micro' is a prefix, and therefore should not be used as a name. The prefix "micro" appears in recognized scientific units, such as "microbar" (abbreviated as  $\mu\text{B}$  in meteorology), "microröntgen" (abbreviated as  $\mu\text{r}$  in radiation oncology and health physics), "microgram" (abbreviated as  $\mu\text{g}$  in molecular biology and physics), "microfarad" and "microCoulomb" (abbreviated  $\mu\text{f}$  or  $\text{uf}$  and  $\mu\text{C}$  in electrical engineering). ISO 10027 calls <00b5> a "micro sign". This is unfortunate terminology for the reasons given above, which probably originated by a report writer who stumbled across some of the units described above and associated 'micro' with the  $\mu$ , but did not realize that the names of those units were each constructed from a prefix named from the Greek adjective for "small" ( $\mu\kappa\rho\varsigma$ ). The unit of measurement's name (sometimes also spelled 'mikron') was derived from the neuter noun analogous to the Greek adjective. There is also a problem of style. The 'micron' (as well as the prefix "micro") is usually represented traditionally as a slanted, sans serif letterform, even when used with a serifed typeface. Accordingly, it is visually different than a Greek 'mu' in the same typeface. While this traditional representation may have originally arisen in the hot metal type era because a 'mu' was not available in each typeface (and perhaps the only 'mu' available was from an italic sans serif typeface), it has become a customary visual appearance for this letterform; and the general scientific paper readership would probably either frown to see something different or misread it. The represented appearance of 'micron' in the American Mathematical Society's database is always slanted and sans serif.

#### 2) 'Idot' at <0130>

The legacy name for the diacritical mark of a 'dot' which is located above the x-height is 'dotaccent', so as to distinguish it from a 'dot' placed over a single glyph form which requires it to represent a base character glyph (e.g. **i** or **j**). However, 'Idot' is not a composite character in the usual sense, but a base letterform that is distinct and just happens to look like a hypothetical 'Idotaccent' if such a character were to exist. 'Idot' is not really a legacy name but is a current name which is conceptually valid for describing what the letterform actually is: a capital I with a dot over it, a dot which is not a diacritical mark. 'Idot' is covered in this category to explain what appears to be a naming inconsistency with composite characters which include a 'dotaccent'. Examples of these are 'Bdotaccent' at <1e02> and other characters in the 1E\_ \_ block using b, D, d, F, f, H, h, M, m, N, n, P, p, R, r, S, s, T, t, W, w, X, x, Y, and y base character glyphs.

#### 3. 'dotaccent' at <02d9> and other character names with 'accent'

The name 'dotaccent' is ambiguous, because current optimal font architectural lexicography (as well as ISO 10646) mandates different diacritical marks with 'dot' in their names for over-placement, under-placement, or other positional placements. The necessity of this is obvious if duplicate character names are to be avoided. 'dotaccent' is a bad name because:

a) The name component "accent" is superfluous in any diacritical mark character name and adds to the length of the name without providing uniqueness to the name which can be achieved in a more direct manner. There is a possible reason for appending "accent" if a name would be indicative or more than one glyph. For example, diacritical marks 'commaabove' and 'commabelow' could have been named 'commaaccent' and 'commabelow' or 'commaaccentbelow'. The name 'dotaccent' was retained solely to insure compatibility with the legacy name. However, there are a number of diacritical marks with "comma" as part of their name: <02BB> 'commainvabovemod', <02BD> 'commarevabovemod', <0312> 'commainvabove', <0313> 'commaabove', <0314> 'commarevabove', <0315> 'commaaboveright', and <0326> 'commabelow'. The same notion applies to diacritical marks having the names with 'dot' in them. Since a large number of all of these type names are involved with pair kern data, including the 'accent' qualifier within their names would significantly increase the amount of data in metric files.

b) There is no indication of positional placement.

A better name would have been 'dotabove'. The name 'dotaccent' is retained as the spacing diacrit, with the name 'dotaccentnosp' used to denote the non-spacing version of the same mark at <0307>, and 'dotbelow' for the non-spacing mark at <0323>. This is in accordance with the naming principles described above. The choice of 'dotaccent' in and of itself is not that bad from a length standpoint (only 1 extra character in the name in lieu of using 'dotabove').

#### 4) 'dotless\_ ' character glyphs

'dotlessi' is a legacy name for a glyph which is used twofold: as a true letterform glyph in the Turkish Roman alphabet, and as a base glyph used to internally construct composite characters composed of diacritical marks applied to the letter **i** above the x-height. (Composite characters based on **i** requiring diacritics used below the baseline still can use **i** as a base glyph.) An almost similar situation exists with **j**, except that 'dotlessj' (which is the **j** counterpart of 'dotlessi') is not (yet) a true letterform glyph in some Roman alphabet. The name 'dotlessi' is also a bad name, because the adjective precedes the base character name, and the name does not start with the letter corresponding to the base character most similar to it, and there is no "dot" glyph component in 'dotlessi'. A sort on character names starting with "i" would not come up with 'dotlessi'. A better name would have been 'idotless'. Nevertheless, for consistency, its **j** counterpart was named 'dotlessj' rather than 'jdotless'. 'dotlessj' is not encoded in ISO 10646 or Unicode because it refers only to a glyph. ('ff', 'ffi', and 'ffl' are also glyphs, but they were encoded as exceptions.) However, the real world situation is that these named but unencoded glyphs cannot be used externally from a font with some font architectures, such as TrueType. (This is, of course, not true with PostScript fonts). This is why 'ff' et al. were encoded in ISO 10646, but 'dotlessj' was not. Production First Software encodes 'dotlessj' in the Private Zone at <e02e>.

A few other base characters are based on 'dotlessj', such as 'dotlessjcross' at <025f> and 'dotlessjcrosshook' at <0284> .

#### 5. The 'fhook' / 'florin' problem

'fhook' is placed at <0192> in ISO 10646. 'florin' (a Dutch currency denomination) is not coded. There is a style difference: 'florin' is usually represented by an inclined serifed **f**, but 'fhook' is a standard **f** with a hook added. Since 'florin' was coded in legacy (PostScript) fonts, it was later placed in ANSI encoding by Microsoft, who then hardwired location <0192> as its Unicode location in Windows 3.1 and WindowsNT. If the correct glyph is placed at <0192>, then an 'fhook' shows up in ANSI, inconsistent with the florin in legacy fonts, and a 'florin' would not be available. To solve this problem, Production First Software places the 'florin' (incorrectly) at <0192> and the 'fhook' in the Private Zone at <f836>.

6. 'Dmacron'/'dmacron' vs. 'Dslash'/'dslash' or 'Dcross'/'dcross'

Apple encodings uses the name of 'dmacron' for a glyph having a horizontal stroke through the **d** ascender. This turns out to be ambiguous because one of the African Roman alphabets requires a true 'Dmacron'/'dmacron'. Therefore, Production First Software represents the true 'Dmacron'/'dmacron' (which, by necessity and as an exception, must violate the naming rule regarding maintaining legacy names, described above). They are encoded in the Private Zone at <f844> and <f845>.

7. 'Dcross' vs. 'Dslash'

'Dcross' at <0110> is designed with a long horizontal stroke so as to be useful for classical Vietnamese. 'Dslash' at <0189> is designed with either a diagonal stroke or a short horizontal stroke, so as to be consistent with other stroked letterforms used in African languages, which are not yet encoded in ISO 10646 or Unicode.

8. 'apostrophe' vs. 'commaabove'

The mark 'apostrophe' is represented in Unicode as a modifier letter at <02bc>. A number of composite characters use an 'apostrophe' as an alternate form to using a 'caron'. However, these are named using the primary ('caron') names with an alternate qualifier. Only one character to date ('napostrophe' at <0149>) actually makes use of 'apostrophe'.

9. 'spacenbr' vs. 'nbspace'

Note, that with space characters, the particle 'space' follows the characterization, because 'space' is a qualifier. Examples include 'space' at <0020>, 'nbspace' (no-break space) at <00a0>, and 'hairspace' at <200a>. Some legacy fonts use the name 'spacenbr' and some use 'nbspace'.

10. 'periodcentered' is an ambiguous character

'periodcentered' character can serve several different purposes. It is more urgently needed as a diacritical mark for Catalan characters, but it can also serve as punctuation. It was originally incorporated in ISO 8859 as a diacritical mark. Unfortunately, the name 'periodcentered' (a legacy name) is a bad choice of names and leads to a conclusion that it is not a diacritical mark. A better name would have been: 'dotcentered'. Since the principle is that a change in location of the diacrit earns the creation of a unique character, 'dotaccent' covers the (centered) position above the x-height and 'dotbelow' covers the (centered) position below the baseline.

6.0 Algorithmically-Derived Composite Character Names: Stacking Hierarchy  
Character names built up from <base char> + <diacritical mark> stack up diacritical mark names outward from the base character:

<diacritical mark-4>  
...  
<diacritical mark-3>  
<base character>  
<diacritical mark-1>  
...  
<diacritical mark-2>

The character name spelling, however, follows the writing direction of the script. For left-to-right scripts, this becomes:

<base character><diacritical mark-1><diacritical mark-2><diacritical mark-3><diacritical mark-4>

An example of this (with 2 marks) is 'Udieresiscaaron' used in Pinyin and found in <01d9>. For right-to-left scripts, it is represented in text (the backing store) as: <diacritical mark 4><diacritical mark 3><diacritical mark 2><diacritical mark 1><base character>. but the corresponding composite name is still: <base character><diacritical mark-1><diacritical mark-2><diacritical mark-3><diacritical mark-4>

## 7.0 Symbol Names

Some exceptions exist. These will be detailed in an updated report.

## 8.0 ZapfDingbat® Names

A block of Unicode (starting with <2700>) is a similar replication of Adobe's ZapfDingbat font. This font has specific character names. However, since Production First Software fonts do not include this block (because it is not style-dependent and because it is widely available as a core system font), there is no naming issue.

## 9.0 Numerically-derived Names

Production First Software derives its numerically-based names on octets, starting from 0 or <0000> or <00000000> to  $16^4$  <ffff> or  $16^8$  <ffffffff>, preceded by a letter, so as to be able to be used as a valid PostScript name. The hexadecimal numbers are represented in Uppercase forms (e.g. <FB01>). This naming scheme is also carried over to 32-bit encoding (ISO10646) by considering each octet as a plane.

<u>Letter/Number</u>	<u>Designation</u>
u	Unicode or Basic multilingual plane (BMP) in ISO 10646
v	Unicode block extension not yet mapped in planes higher than 0
w	reserved for script or block not mapped in Unicode or ISO 10646 planes
x	reserved for script or block not mapped in Unicode or ISO 10646 planes
y	reserved for script or block not mapped in Unicode or ISO 10646 planes
z	reserved for script or block not mapped in Unicode or ISO 10646 planes
i<plane number>	<plane number, higher than 0>

This scheme has been formulated to minimize the length of derived names. For example, using a scheme like 'u0180' is better than 'un0180' or 'uni0180'. 'uni0180' requires 20% more characters than 'u0180'. In an AFM file representing a 2-byte font with 3,000 characters having 60,000 kern pairs (an AFM file size of about 4.3Mb), you increase the file size about 2Mb with 'uni- - -' names as opposed to 'u- - -' names, an increase of almost 50%. A 2-byte U14 TrueType font increases in size from about 850kb to 1.0 Mb, a 15% increase. This is due to the character names stored in the 'post' table. There are ramifications with other software, such as PostScript drivers which may have several sets of encoding vectors built into the preambles produced. Note that if the 'i<plane number>' naming method is used for plane 0 (the first plane of ISO 10646 and also identical to Unicode), then 'u0180' becomes 'i00000180' which is an unnecessarily long name. We therefore use the 'u- - -' naming method for plane 0.

In 4-byte fonts, for a character which is to be assigned a name based on a numerically-derived (e.g. code point) algorithm, a character in plane 1 at location <0180> in that plane in 2-byte code space (or at location <00010180> in 4-byte code space) would be 'i00010180'. This scheme is capable of spanning a 32-bit code space, whereas the method of using the 'AFII' prescript uses the same number of characters in a character name but can only span a 16-bit codespace.

## 10.0 Private Zone

The Private Zone (<e000> to <f8ff>) is likely to cause great problems in the years to come. Unicode divides this range into an "End User" area starting at <e000> and going higher, and a "Corporate Use" area starting at <f8ff> and going lower. This is 4,096 characters for the E0 block and 2,304 characters for the F0 block, or a total of 6,400 characters. Production First Software uses both the E0 block (starting at <e000> and extending to <e8ff>) and the F8 block (starting at <f800> and extending to <f8ff>). The E0 block is used for glyphs not encoded in Unicode. The F8 block is used for alternate forms of glyphs for characters which are encoded.

## 11.0 Notes

Note 1 - Greek is the only non-Latin script where the script alphabet letterform characters are named using transliterated English. However, exceptions to not using transliterated non-Latin script letterform names are permitted if a non-Latin script letterform glyph is used to signify something else (such as a mathematical operator or currency sign). Examples include 'scruple' at <2108> and 'bet' at <2136> (which is pronounced when reading a mathematical equation as "bet", not "bet symbol").

Note 2 - This provides protection against various spellings of the same script alphabet letterform name in different spoken languages. An exception is with diacritical mark names, in which the opposite is usually true. For example, an 'acute' is usually spelled the same way in all languages using the Roman alphabet. However, in Cyrillic (the script used for spelling Russian) there is no 'acute' (which is a mark used to lengthen a sound), but there is a mark used to add (timing) stress, appearing identically to the 'acute'. It is described (in English) using various words like "tonic". These are coded as 'v0194' and 'v0195' for the Uppercase and lowercase versions in Production First Software fonts. A stress mark is also present in the International Phonetic Alphabet as 'stresspri' at <02c8> in Production First Software fonts, but it does not resemble the 'acute'. A related technique of using a script-like indicator to differentiate similarly-appearing marks is not workable because, if a mark qualifies to be defined separately, its transliterated native script name is sufficient and already a proven method.

Note 3 - This principle operates for the widespread benefit of many names, such as how to designate a numerically-derived name. There are some cases where the application of the other naming principles produce extremely long names (for example the Greek Extended (Polytonal) composite character 'omegadasiaperispomeniiprogegrammeni' at <1fa7>, having essentially three diacritical marks, in Production First Software fonts). However, for most character and glyph names, the naming principles described in this report minimize the average size of names.

Note 4 - Obviously, using the qualifier 'stroke' by itself is meaningless; because strokes can be applied to base glyphs in various positions. Rather than using 'strokehoriz', 'strokevert', 'strokeddiag', etc., the 'cross', 'bar', 'slash' approach is more meaningful, especially when combined with additional qualifiers where necessary. It is also consistent with legacy names (for example 'Lslash' and 'Islash'). Examples include 'Dcrosstop' at <018b>. The ISO 10027 name for this is "Latin Capital Letter D with Topbar". The problem with this name is that the stroke is horizontal, and another character was named "Latin Letter D with Stroke" at <0110> which has a horizontal stroke at midheight, and "Latin Letter L with Stroke" at <0141> which has a diagonal stroke. Similarly, there is a "Latin Capital Letter G with Stroke" at <01E4> used in Lappish. This is named 'Gcross' in Production First Software fonts. Unfortunately, there is also a "Latin Capital Letter G with (diagonal) Stroke" which is used in Central European languages (specifically, Latvian). Although not coded in ISO 10646/Unicode, it is coded in Production First Software fonts in a 1T supplement as 'Gslash' at <0192>. Eventually, some of the supplement glyphs may be added to the Private Zone.