3.6 Compatibility Zone

The Compatibility Zone of the Unicode standard contains characters that can be mapped to other areas.

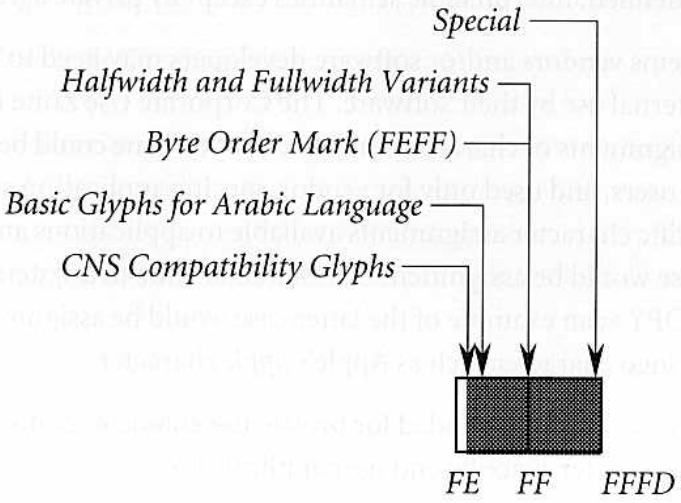

Figure 3-14. Compatibility Zone

Compatibility Zone U+FE00 → U+FFEF

Description. The Compatibility Zone is so named because it contains miscellaneous glyphs, contextual or orientational variants, and width variants that can legitimately be mapped to other characters in the Unicode standard, but which require specific Unicode values for compatibility with pre-existing character standards. Receivers of Compatibility Zone characters, if they so desire, are free to replace those characters by corresponding regular Unicode characters under any circumstances without being considered non-conformant. Tables for canonical mappings between Compatibility Zone characters and characters in the Unicode character encoding are located in Chapter 5.

Compatibility Zone characters are provided solely for backwards compatibility to existing standards. The content of the Compatibility Zone includes characters which are in widespread use in existing implementations, but whose semantics or usage (such as vertical forms, contextual shapes, small, half width or full width variants) is fundamentally at odds with the way in which the Unicode character encoding generally handles characters. Compatibility Zone characters are publicly defined for Unicode code interchange, but are *not* candidates for inclusion in the Unicode standard.

Standards. The Compatibility Zone includes basic contextual forms of Arabic glyphs that are encoded on some corporate code pages, as well as spacing forms of Arabic accents. The Compatibility Zone also contains characters for compatibility with CNS 11643, JIS X 0208, and KS C 5601, and their implementation in "shift" forms in corporate standards and other coded implementations of the CJK standards.

CNS 11643 Compatibility Glyphs. U+FE30 → U+FE4F contain vertical rendering forms for a number of glyphs, as well as other glyphs included for compatibility with CNS 11643. The vertical rendering forms should be mapped to their non-vertical prototypes, as Unicode-conformant software should adjust horizontal and vertical rendering forms by rendering context, rather than encoding them as distinct characters.

Small Glyph Variants. U+FE50 → U+FE6F contain small glyph variants for compatibility with CNS 11643. The semantic of these small variants is unclear, although they may be intended for superscript rendering. They should be mapped to corresponding characters in the Unicode standard.

Contextual Shape Forms for Arabic Glyphs. U+FE70 → U+FEFE contain contextual shape forms for initial, medial, final, and isolated forms of Arabic glyphs, for compatibility with pre-existing standards and implementations which use these forms as characters. They can be replaced by

letters from the Arabic block (U+0600 → U+06FF). Implementations can handle contextual glyph shaping by rendering rules when accessing glyphs from fonts, rather than by encoding contextual shapes as characters.

Halfwidth and Fullwidth Variants. The Unicode standard includes the basic halfwidth Katakana and Hangul as well as fullwidth ASCII characters for compatibility with the various CJK coding standards. The term "halfwidth" refers to characters which are, in East Asian computer implementations, rendered at half the standard width of a normal ideographic character. "Fullwidth" refers to the normal width of a Hangul or Chinese character. In many Japanese systems, for instance, ASCII characters print as halfwidth, but are cloned in a fullwidth version in the JIS character set; the fullwidth version of ASCII characters are encoded in the Compatibility Zone to distinguish them from the halfwidth version when necessary for backwards compatibility with preexisting data. In the Unicode standard, the preferred way to distinguish between full- and halfwidth characters is through font changes.

Encoding structure. The Unicode block for the Compatibility Zone is divided into the following ranges:

U+FE00 → U+FE2F	Currently unassigned
U+FE30 → U+FE44	Vertical glyph variants
U+FE45 → U+FE48	Currently unassigned
U+FE49 → U+FE4F	Overscores and underscores
U+FE50 → U+FE6B	Small glyph variants
U+FE6C → U+FE6F	Currently unassigned
U+FE70 → U+FEFC	Glyphs for basic Arabic
U+FEFD → U+FEFE	Currently unassigned
U+FEFF	Byte order mark
U+FF00	Currently unassigned
U+FF01 → U+FF5E	Fullwidth ASCII
U+FF5F	Currently unassigned
U+FF60 → U+FF9F	Halfwidth Katakana
U+FFA0 → U+FFDF	Halfwidth Hangul
U+FFE0 → U+FFE6	Fullwidth symbols
U+FFE7 → U+FFEF	Currently unassigned

Special U+FFF0 → U+FFFD

The fourteen Unicode values from U+FFF0 → U+FFFD are reserved for special character definition. The only special character currently defined is U+FFFD REPLACEMENT CHARACTER, which is the general substitute character in the Unicode standard. That character can be substituted for any "unknown" character in another encoding which cannot be mapped in terms of known Unicode values (see Appendix C).

In addition, there are two 16-bit unsigned hexadecimal values which are defined NOT to be Unicode character values, and one value which may be used to flag and test for correct byte order polarity. U+FEFF BYTE ORDER MARK and U+FFFE are the (byte-swapped) mirror image of each other. They are not control characters that select the byte order of text; rather, their function is to assure recipients that they are looking at a correctly byte-ordered file.

U+FEFF. This Unicode special character is defined to be a signal of correct byte-order polarity. An application may use this signal character to explicitly enable the "big-endian" or "little-endian" byte order to be determined in Unicode text which may exist in either byte order (for example in networks which mix Intel and Motorola or RISC CPU architectures for data storage). U+FEFF is the "correct" or legal order; finding a value U+FFFE is a signal that text of the "incorrect" byte order for an interpreting process has been encountered.

U+FFFE. The 16-bit unsigned hexadecimal value U+FFFE is *not* a Unicode character value, and should be taken as a signal that Unicode characters should be byte-swapped before interpretation. U+FFFE should only be interpreted as an incorrectly byte-swapped version of U+FEFF.

U+FFFF. The 16-bit unsigned hexadecimal value U+FFFF is *not* a Unicode character value, and can be used by an application as a error code or other non-character value. The specific interpretation of U+FFFF is not defined by the Unicode standard, so it can be viewed as a kind of private-use non-character.

Encoding Structure. The Unicode block for Special characters is divided into the following ranges:

U+FFF0 → U+FFFC	Currently unassigned
U+FFFD	Replacement character
U+FEFF	Byte order mark
U+FFFE	Not a character; signal value
U+FFFF	Not a character