6.2.2 8 bit data

8 bit data is user defined

Padding:	CR (coded as 0x0D) in the case of an 8 bit character set
	Otherwise - user defined
Character table:	User Specific

Deprecated for all use except for UTF-8 (ISO/IEC 10646 [10]). The UTF-8 text should be in NFC (Normal Form C) (Unicode). For UTF-8, the mappings in subclause 6.2.3 apply when sending as well as receiving an SMS message. **This coding must not be used for USSD nor CBS, and should not be used for SMS.**

When splitting a message text into submessages (using UTF-8), there must be no cut inside of a UTF-8 byte sequence.

6.2.3 UCS2

Bits per character:	16 (bits per code unit; a character in planes 1-17 are coded by 2 16-bit code units)
CBS/USSD pad character:	CR
Character table:	ISO/IEC 10646 [10]

While this specification refers to UCS2 (actually, UCS2 big endian), implementations should handle UTF-16BE, which extends UCS2 (big endian) beyond the BMP (Basic Multilingual Plane) to the 17 (0 to 16) planes of ISO/IEC 10646 while keeping the 16-bit encoding for the BMP (Plane 0). The UCS2/UTF16 should be in NFC (Normal Form C) (Unicode).

When splitting a message text into submessages (using this encoding), there must be no cut inside of a 16-bit unit (if UCS2 is supported), nor between "high" surrogate and a "low" surrogate (the pair encodes for a character in plane 1 to 17; if UTF-16 is supported). (There should be no surrogate not in such a pairs; that would be an encoding error.)

UCS2(BE) (UTF-16BE) should not be used for USSD; a 7-bit alphabet, preferably 0x00, should be used. (However, *****, **#**, **+**, **0**-**9** and **a**-**z** (lowercase!) are encoded the same in all non-deprecated 7-bit alphabets, in particular for USSD commands, and 2-letter CBS lowercase language tags.)

Note that for CBS, the 2-letter language tag prefix for UCS2 encoded messages are encoded in the 7-bit 0x00 alphabet in the first two bytes; it should be lowercase. (SMS does not have language tags of any kind.)

Furthermore, if any of the 7-bit alphabets 0x10 and onwards can be used for an SMS/CBS message, use of that 7-bit alphabet is preferred, as that can more than half the length (in bytes) of the message, allowing for fewer submessages and allowing a longer (in characters) total message. (Note that alphabets 0x00-0x0F are deprecated for SMS/CBS messages.)

For messages originally encoded in a UCS encoding, the following mappings should be used, both upon sending (before encoding in a 7-bit encoding, if that is to be used), as well as upon receiving a UCS2 SMS/CBS message:

- U+0000-U+0008, U+000E-U+001F, U+007F-U+009A, U+009C-U+00A0: map to U+FFFD.
- U+0009: map to U+0020. U+000B, U+2028-U+2029: map to U+000A. U+061C: map to U+200F.
- U+2000-U+200B: map to U+FFFD. U+200C: map to U+00B7 (use middle dot as non-joiner). U+200D: map to U+0640 (use kashida as joiner).
- U+202A-U+202E, U+205F-U+206F, U+D800-U+F8FF, U+FDD0-U+FDEF, U+FEFF, U+FFFC, U+xFFFE-U+xFFFF (where *x* is in {0, …, 10_{16}}: map toU+FFFD.

In case there are any bidi RTL characters in the message, a *simplified* Unicode bidi algorithm should be used before display of the message. Bidi AL is handled as bidi R, except Arabic punctuation which is handled as ON; all bidi EN and AN handled as bidi L; all bidi ET, ES, CS handled as bidi ON. Furthermore, the 'paragraph direction' is always LTR for SMS/CBS messages. Note also that there are no nestable bidi controls in SMS/CBS (or they are not interpreted in the simplified bidi algorithm, in case the mappings above have not been applied). Furthermore, Arabic is a ‘cursive’ script and Arabic characters thus should be cursive shaped (initial, medial, final, isolated forms), and both Arabic and Hebrew may have (certain) combining characters applied. Bidi mirroring (when displaying RTL text sections) applies, but only those that can be done via using the data in BidiMirroring.txt (Unicode).

Brahmic (Indic) scripts (if supported) also should be handled specially for rendering; the details of which are beyond the scope of this standard. Further, while most ‘emoji’ characters are isolated small coloured drawings, many emoji are made from ‘emoji sequences’, multiple emoji characters creating one emoji to display.

For the Latin (“default”) and Greek scripts, most text for “European” languages (and Vietnamese) can be handled be handled by “precomposed” letters. However, to be able to fit these into one 7-bit alphabet each, we rely combining characters, in particular for East European languages and Vietnamese as well as polytonic Greek (modern Greek orthography is monotonic Greek, and that is covered without the need for combining characters in the 7-bit alphabet for Greek). The easiest way to handle that is to upon reception of a message in Latin (0x10) or Greek (0x11) 7-bit alphabet, is to first convert to Unicode/10646 and then apply Normal Form C (NFC). Upon sending, first apply NFC, then decompose letters that need to be represented in decomposed form in the Latin or Greek 7-bit alphabet.