The Unicode Standard
Version 6.0 – Core Specification

To learn about the latest version of the Unicode Standard, see http://www.unicode.org/versions/latest/.

The Unicode Standard / the Unicode Consortium ; edited by Julie D. Allen ... [et al.]. — Version 6.0.
Includes bibliographical references and index.
ISBN 978-1-936213-01-6 (http://www.unicode.org/versions/Unicode6.0.0/)
1. Unicode (Computer character set) I. Allen, Julie D. II. Unicode Consortium.
QA268.U545 2011

ISBN 978-1-936213-01-6
Published in Mountain View, CA
February 2011

Appendix A

Notational Conventions

This appendix describes the typographic conventions used throughout this core specification.

Code Points

In running text, an individual Unicode code point is expressed as U+*n*, where *n* is four to six hexadecimal digits, using the digits 0–9 and uppercase letters A–F (for 10 through 15, respectively). Leading zeros are omitted, unless the code point would have fewer than four hexadecimal digits—for example, U+0001, U+0012, U+0123, U+1234, U+12345, U+102345.

- U+0416 is the Unicode code point for the character named CYRILLIC CAPITAL LETTER ZHE.

The *U+* may be omitted for brevity in tables or when denoting ranges.

A range of Unicode code points is expressed as *U+xxxx–U+yyyy* or *xxxx..yyyy*, where *xxxx* and *yyyy* are the first and last Unicode values in the range, and the long dash or two dots indicate a contiguous range inclusive of the endpoints. For ranges involving supplementary characters, the code points in the ranges are expressed with five or six hexadecimal digits.

- The range U+0900–U+097F contains 128 Unicode code points.
- The Plane 16 private-use characters are in the range 100000..10FFFD.

Character Names

In running text, a formal Unicode name is shown in small capitals (for example, GREEK SMALL LETTER MU), and alternative names (aliases) appear in italics (for example, *umlaut*). Italics are also used to refer to a text element that is not explicitly encoded (for example, *pasekh alef*) or to set off a non-English word (for example, the Welsh word *ynghyd*).

For more information on Unicode character names, see *Section 4.8, Name*.

For notational conventions used in the code charts, see *Section 17.1, Character Names List*.

Character Blocks

When referring to the normative names of character blocks in the text of the standard, the character block name is titlecased and is used with the term "block." For example:

> the Latin Extended-B block

Optionally, an exact range for the character block may also be cited:

> the Alphabetic Presentation Forms block (U+FB00..U+FB4F)

These references to normative character block names should not be confused with the headers used throughout the text of the standard, particularly in the block description chapters, to refer to particular ranges of characters. Such headers may be abbreviated in

various ways and may refer to subranges within character blocks or ranges that cross character block boundaries. For example:

Latin Ligatures: U+FB00–U+FB06

The definitive list of normative character block names is Blocks.txt in the Unicode Character Database.

Sequences

A sequence of two or more code points may be represented by a comma-delimited list, set off by angle brackets. For this purpose, angle brackets consist of U+003C LESS-THAN SIGN and U+003E GREATER-THAN SIGN. Spaces are optional after the comma, and U+ notation for the code point is also optional—for example, "<U+0061, U+0300>".

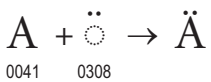
When the usage is clear from the context, a sequence of characters may be represented with generic short names, as in "<a, grave>", or the angle brackets may be omitted.

In contrast to sequences of code points, a sequence of one or more code *units* may be represented by a list set off by angle brackets, but without comma delimitation or U+ notation. For example, the notation "<nn nn nn nn>" represents a sequence of bytes, as for the UTF-8 encoding form of a Unicode character. The notation "<nnnn nnnn>" represents a sequence of 16-bit code units, as for the UTF-16 encoding form of a Unicode character.

Rendering

A figure such as *Figure A-1* depicts how a sequence of characters is typically rendered.

Figure A-1. Example of Rendering



The sequence under discussion is depicted on the left of the arrow, using representative glyphs and code points below them. A possible rendering of that sequence is depicted on the right side of the arrow.

Properties and Property Values

The names of properties and property values appear in titlecase, with words connected by an underscore—for example, General_Category or Uppercase_Letter. In some instances, short names are used, such as gc=Lu, which is equivalent to General_Category = Uppercase_Letter. Long and short names for all properties and property values are defined in the Unicode Character Database; see also *Section 3.5, Properties*.

Occasionally, and especially when discussing character properties that have single words as names, such as *age* and *block,* the names appear in lowercase italics.

Miscellaneous

Phonemic transcriptions are shown between slashes, as in Khmer /khnyom/.

Phonetic transcriptions are shown between square brackets, using the International Phonetic Alphabet. (Full details on the IPA can be found on the International Phonetic Association's Web site, http://www2.arts.gla.ac.uk/IPA/ipa.html.)

A leading asterisk is used to represent an incorrect or nonoccurring linguistic form.

In this specification, the word "Unicode" when used alone as a noun refers to the Unicode Standard.

Unambiguous dates of the current common era, such as 1999, are unlabeled. In cases of ambiguity, CE is used. Dates before the common era are labeled with BCE.

The term *byte*, as used in this standard, always refers to a unit of eight bits. This corresponds to the use of the term *octet* in some other standards.

Extended BNF

The Unicode Standard and technical reports use an extended BNF format for describing syntax. As different conventions are used for BNF, *Table A-1* lists the notation used here.

Table A-1. Extended BNF

Symbols	Meaning
`x := ...`	production rule
`x y`	the sequence consisting of `x` then `y`
`x*`	zero or more occurrences of `x`
`x?`	zero or one occurrence of `x`
`x+`	one or more occurrences of `x`
`x | y`	either `x` or `y`
`( x )`	for grouping
`x || y`	equivalent to `(x | y | (x y))`
`{ x }`	equivalent to `(x)?`
`"abc"`	string literals ("_" is sometimes used to denote space for clarity)
`'abc'`	string literals (alternative form)
`sot`	start of text
`eot`	end of text
`\u1234`	Unicode code points within string literals or character classes
`\U00101234`	Unicode code points within string literals or character classes
`U+HHHH`	Unicode character literal: equivalent to '\uHHHH'
`U-HHHHHHHH`	Unicode character literal: equivalent to '\UHHHHHHHH'
`[gc=Lu]`	character class (syntax below)

In other environments, such as programming languages or markup, alternative notation for sequences of code points or code units may be used.

Character Classes. A *code point class* is a specification of an unordered set of code points. Whenever the code points are all assigned characters, it can also be referred to as a *character class*. The specification consists of any of the following:

- A literal code point
- A range of literal code points
- A set of code points having a given Unicode character property value, as defined in the Unicode Character Database (see PropertyAliases.txt and PropertyValueAliases.txt)
- Non-Boolean properties given as an expression `<property> = <property_value>` or `<property> ≠ <property_value>`, such as "`General_Category=Titlecase_Letter`"

- Boolean properties given as an expression `<property> = true` or `<property> ≠ true`, such as "`Uppercase=true`"
- Combinations of logical operations on classes

Further extensions to this specification of character classes are used in some Unicode Standard Annexes and Unicode Technical Reports. Such extensions are described in those documents, as appropriate.

A partial formal BNF syntax for character classes as used in this standard is given by the following:

```
char_class  := "[" char_class - char_class "]"            set difference
            := "[" item_list "]"
            := "[" property ("=" | "≠") property_value "]"
item_list   := item ("," item)?
item        := code_point                                 either literal or escaped
            := code_point - code_point                    inclusive range
```

Whenever any character could be interpreted as a syntax character, it must be escaped. Where no ambiguity would result (with normal operator precedence), extra square brackets can be discarded. If a space character is used as a literal, it is escaped. Examples are found in *Table A-2*.

Table A-2. Character Class Examples

Syntax	Matches
`[a-z]`	English lowercase letters
`[a-z]-[c]`	English lowercase letters except for c
`[0-9]`	European decimal digits
`[\u0030-\u0039]`	(same as above, using Unicode escapes)
`[0-9 A-F a-f]`	hexadecimal digits
`[\p{gc=Letter} \p{gc=Nonspacing_Mark}]`	all letters and nonspacing marks
`[\p{gc=L} \p{gc=Mn}]`	(same as above, using abbreviated notation)
`[^\p{gc=Unassigned}]`	all assigned Unicode characters
`[\u0600-\u06FF - \p{gc=Unassigned}]`	all assigned Arabic characters
`[\p{Alphabetic}]`	all alphabetic characters
`[^\p{Line_Break=Infix_Numeric}]`	all code points that do not have the line break property of Infix_Numeric

For more information about character classes, see Unicode Technical Standard #18, "Unicode Regular Expressions."

Operators

Operators used in this standard are listed in *Table A-3*.

Table A-3. Operators

Symbol	Meaning
→	is transformed to, or behaves like
↛	is not transformed to
/	integer division (rounded down)
%	modulo operation; equivalent to the integer remainder for positive numbers
¬	logical not