## *Electronic Edition*

This file is part of the electronic edition of *The Unicode Standard, Version 5.0*, provided for online access, content searching, and accessibility. It may not be printed. Bookmarks linking to specific chapters or sections of the whole Unicode Standard are available at

http://www.unicode.org/versions/Unicode5.0.0/bookmarks.html

## *Purchasing the Book*

For convenient access to the full text of the standard as a useful reference book, we recommend purchasing the printed version. The book is available from the Unicode Consortium, the publisher, and booksellers. Purchase of the standard in book format contributes to the ongoing work of the Unicode Consortium. Details about the book publication and ordering information may be found at

http://www.unicode.org/book/aboutbook.html

## *Joining Unicode*

You or your organization may benefit by joining the Unicode Consortium: for more information, see Joining the Unicode Consortium at

http://www.unicode.org/consortium/join.html

*Library of Congress Cataloging-in-Publication Data*

The Unicode Standard / the Unicode Consortium ; edited by Julie D. Allen ... [et al.]. — Version 5.0.
p. cm.
Includes bibliographical references and index.
ISBN 0-321-48091-0 (hardcover : alk. paper)
1. Unicode (Computer character set) I. Allen, Julie D.
II. Unicode Consortium.
QA268.U545 2007
005.7'22—dc22

2006023526



ISBN 0-321-48091-0
Text printed in the United States on recycled paper at Courier in Westford, Massachusetts.
First printing, October 2006

Chapter 4

# *Character Properties*

> ***Disclaimer***
>
> The content of all character property tables has been verified as far as possible by the Unicode Consortium. However, in case of conflict, the most authoritative version of the information for Version 5.0.0 is that supplied in the Unicode Character Database on the Unicode Web site. *The contents of all the tables in this chapter may be superseded or augmented by information in future versions of the Unicode Standard.*

The Unicode Standard associates a rich set of semantics with characters and, in some instances, with code points. The support of character semantics is required for conformance; see *Section 3.2, Conformance Requirements*. Where character semantics can be expressed formally, they are provided as machine-readable lists of character properties in the Unicode Character Database (UCD). This chapter gives an overview of character properties, their status and attributes, followed by an overview of the UCD and more detailed notes on some important character properties. For a further discussion of character properties, see Unicode Technical Report #23, "Unicode Character Property Model."

***Status and Attributes.*** Character properties may be normative or informative. Normative properties are those required for conformance. The following sections discuss important properties identified by their status. Many Unicode character properties can be overridden by implementations as needed. *Section 3.2, Conformance Requirements*, specifies when such overrides must be documented. A few properties, such as Noncharacter_Code_Point, may not be overridden. See *Section 3.5, Properties*, for the formal discussion of the status and attributes of properties.

***Consistency of Properties.*** The Unicode Standard is the product of many compromises. It has to strike a balance between uniformity of treatment for similar characters and compatibility with existing practice for characters inherited from legacy encodings. Because of this balancing act, one can expect a certain number of anomalies in character properties. For example, some pairs of characters might have been treated as canonical equivalents but are left unequivalent for compatibility with legacy differences. This situation pertains to U+00B5 µ MICRO SIGN and U+03BC μ GREEK SMALL LETTER MU, as well as to certain Korean jamo.

In addition, some characters might have had properties differing in some ways from those assigned in this standard, but those properties are left as is for compatibility with existing practice. This situation can be seen with the halfwidth voicing marks for Japanese (U+FF9E HALFWIDTH KATAKANA VOICED SOUND MARK and U+FF9F HALFWIDTH KATAKANA SEMI-VOICED SOUND MARK), which might have been better analyzed as spacing combining marks, and with the conjoining Hangul jamo, which might have been better analyzed as an initial base character followed by formally combining medial and final characters. In the interest of efficiency and uniformity in algorithms, implementations may take advantage of such reanalyses of character properties, as long as this does not conflict with the conformance requirements with respect to normative properties. See *Section 3.5, Properties*; *Section 3.2, Conformance Requirements*; and *Section 3.3, Semantics*, for more information.

## 4.1 Unicode Character Database

The Unicode Character Database (UCD) consists of a set of files that define the Unicode character properties and internal mappings. For each property, the files determine the assignment of property values to each code point. The UCD also supplies recommended property aliases and property value aliases for textual parsing and display in environments such as regular expressions.

The properties include the following:

- Name
- General Category (basic partition into letters, numbers, symbols, punctuation, and so on)
- Other important general characteristics (whitespace, dash, ideographic, alphabetic, noncharacter, deprecated, and so on)
- Display-related properties (bidirectional class, shaping, mirroring, width, and so on)
- Casing (upper, lower, title, folding—both simple and full)
- Numeric values and types
- Script and Block
- Normalization properties (decompositions, decomposition type, canonical combining class, composition exclusions, and so on)
- Age (version of the standard in which the code point was first designated)
- Boundaries (grapheme cluster, word, line, and sentence)
- Standardized variants

See the Unicode Character Database for more details on the character properties, their distribution across files, and the file formats.

***Unihan Database.*** In addition, a large number of properties specific to CJK ideographs are defined in the Unicode Character Database. These properties include source information, radical and stroke counts, phonetic values, meanings, and mappings to many East Asian standards. These properties are documented in the file Unihan.txt, also known as the *Unihan Database*. For a complete description of the properties in the Unihan Database, see the documentation file Unihan.html in the Unicode Character Database. (See also "Online Unihan Database" in *Section B.6, Other Unicode Online Resources*.)

Many properties apply to both ideographs and other characters. These are not specified in the Unihan Database.

***Stability.*** While the Unicode Consortium strives to minimize changes to character property data, occasionally character properties must be updated. When this situation occurs, a new version of the Unicode Character Database is created, containing updated data files. Data file changes are associated with specific, numbered versions of the standard; character properties are never silently corrected between official versions.

Each version of the Unicode Character Database, once published, is absolutely stable and will never change. Implementations or specifications that refer to a specific version of the UCD can rely upon this stability. Detailed policies on character encoding stability as they relate to properties are found in *Appendix F, Unicode Encoding Stability Policies*. See the subsection "Policies" in *Section B.6, Other Unicode Online Resources*. See also the discussion of versioning and stability in *Section 3.1, Versions of the Unicode Standard*.

***Aliases.*** Character properties and their values are given formal aliases to make it easier to refer to them consistently in specifications and in implementations, such as regular expressions, which may use them. These aliases are listed exhaustively in the Unicode Character Database, in the data files PropertyAliases.txt and PropertyValueAliases.txt.

Many of the aliases have both a long form and a short form. For example, the General Category has a long alias "General_Category" and a short alias "gc". The long alias is more comprehensible and is usually used in the text of the standard when referring to a particular character property. The short alias is more appropriate for use in regular expressions and other algorithmic contexts.

In comparing aliases programmatically, loose matching is appropriate. That entails ignoring case differences and any whitespace, underscore, and hyphen characters. For example, "GeneralCategory", "general_category", and "GENERAL-CATEGORY" would all be considered equivalent property aliases. See UCD.html in the Unicode Character Database for further discussion of property and property value matching.

For each character property whose values are not purely numeric, the Unicode Character Database provides a list of value aliases. For example, one of the values of the Line_Break property is given the long alias "Open_Punctuation" and the short alias "OP".

Property aliases and property value aliases can be combined in regular expressions that pick out a particular value of a particular property. For example, "\p{lb=OP}" means the Open_Punctuation value of the Line_Break property, and "\p{gc=Lu}" means the Uppercase_Letter value of the General_Category property.

Property aliases define a namespace. No two character properties have the same alias. For each property, the set of corresponding property value aliases constitutes its own namespace. No constraint prevents property value aliases for *different* properties from having the same property value alias. Thus "B" is the short alias for the Paragraph_Separator value of the Bidi_Class property; "B" is also the short alias for the Below value of the Canonical_Combining_Class property. However, because of the namespace restrictions, any combination of a property alias plus an appropriate property value alias is guaranteed to constitute a unique string, as in "\p{bc=B}" versus "\p{ccc=B}".

For a recommended use of property and property value aliases, see Unicode Technical Standard #18, "Unicode Regular Expressions." Aliases are also used for normatively referencing properties, as described in *Section 3.1, Versions of the Unicode Standard*.

***CD-ROM and Online Availability.*** A copy of the 5.0.0 version of the UCD is provided on the CD-ROM. All versions of the UCD are available online on the Unicode Web site. See the subsections "Online Unicode Character Database" and "Online Unihan Database" in *Section B.6, Other Unicode Online Resources*.

## 4.2 Case—Normative

*Case* is a normative property of characters in certain alphabets whereby characters are considered to be variants of a single letter. These variants, which may differ markedly in shape and size, are called the *uppercase* letter (also known as *capital* or *majuscule*) and the *lowercase* letter (also known as *small* or *minuscule*). The uppercase letter is generally larger than the lowercase letter.

Because of the inclusion of certain composite characters for compatibility, such as U+01F1 LATIN CAPITAL LETTER DZ, a third case, called *titlecase*, is used where the first character of a word must be capitalized. An example of such a character is U+01F2 LATIN CAPITAL LETTER D WITH SMALL LETTER Z. The three case forms are UPPERCASE, Titlecase, and lowercase.

For those scripts that have case (Latin, Greek, Coptic, Cyrillic, Glagolitic, Armenian, Deseret, and archaic Georgian), uppercase characters typically contain the word *capital* in their names. Lowercase characters typically contain the word *small*. However, this is not a reliable guide. The word *small* in the names of characters from scripts other than those just listed has nothing to do with case. There are other exceptions as well, such as small capital letters that are not formally uppercase. Some Greek characters with *capital* in their names are actually titlecase. (Note that while the archaic Georgian script contained upper- and lowercase pairs, they are not used in modern Georgian. See *Section 7.7, Georgian*.) The authoritative source for case of Unicode characters is the specification of lowercase, uppercase, and titlecase properties in the Unicode Character Database.

### *Case Mapping*

The default case mapping tables defined in the Unicode Standard are normative, but may be overridden to match user or implementation requirements. The Unicode Character Database contains five files with case mapping information, as shown in *Table 4-1*. Full case mappings for Unicode characters are obtained by using the basic mappings from UnicodeData.txt and extending or overriding them where necessary with the mappings from SpecialCasing.txt. Full case mappings may depend on the context surrounding the character in the original string.

Some characters have a "best" single-character mapping in UnicodeData.txt as well as a full mapping in SpecialCasing.txt. Any character that does not have a mapping in these files is considered to map to itself. For more information on case mappings, see *Section 5.18, Case Mappings*.

**Table 4-1.** Sources for Case Mapping Information

| File Name | Description |
|---|---|
| UnicodeData.txt | Contains the case mappings that map to a single character. These do not increase the length of strings, nor do they contain context-dependent mappings. |
| SpecialCasing.txt | Contains additional case mappings that map to more than one character, such as "ß" to "SS". Also contains context-dependent mappings, with flags to distinguish them from the normal mappings, as well as some locale-dependent mappings. |
| CaseFolding.txt | Contains data for performing locale-independent case folding, as described in "Caseless Matching," in *Section 5.18, Case Mappings*. |
| DerivedCoreProperties.txt | Contains definitions of the properties Lowercase and Uppercase. |
| PropList.txt | Contains the definition of the property Soft_Dotted. |

The single-character mappings in UnicodeData.txt are insufficient for languages such as German. Therefore, only legacy implementations that cannot handle case mappings that increase string lengths should use UnicodeData.txt case mappings alone.

A set of charts that show the latest case mappings is also available on the Unicode Web site. See "Charts" in *Section B.6, Other Unicode Online Resources*.

## 4.3 Combining Classes—Normative

Each combining character has a normative canonical *combining class*. This class is used with the Canonical Ordering Algorithm to determine which combining characters interact typographically and to determine how the canonical ordering of sequences of combining characters takes place. Class *zero* combining characters act like base letters for the purpose of determining canonical order. Combining characters with non-zero classes participate in

reordering for the purpose of determining the canonical order of sequences of characters. (See *Section 3.11, Canonical Ordering Behavior*, for a description of the algorithm.)

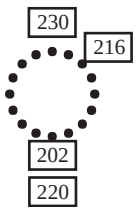
The list of combining characters and their canonical combining class appears in the Unicode Character Database. Most combining characters are nonspacing.

The canonical order of character sequences does *not* imply any kind of linguistic correctness or linguistic preference for ordering of combining marks in sequences. For more information on rendering combining marks, see *Section 5.13, Rendering Nonspacing Marks*.

Class zero combining marks are never reordered by the Canonical Ordering Algorithm. Except for class zero, the exact numerical values of the combining classes are of no importance in canonical equivalence, although the relative magnitude of the classes is significant. For example, it is crucial that the combining class of the cedilla be lower than the combining class of the dot below, although their exact values of 202 and 220 are not important for implementations.

Certain classes tend to correspond with particular rendering positions relative to the base character, as shown in *Figure 4-1*.

**Figure 4-1.** Positions of Common Combining Marks




### *Reordrant, Split, and Subjoined Combining Marks*

In some scripts, the rendering of combining marks is notably complex. This is true in particular of the Brahmi-derived scripts of South and Southeast Asia, whose vowels are often encoded as class zero combining marks in the Unicode Standard, known as *matras* for the Indic scripts.

In the case of simple combining marks, as for the accent marks of the Latin script, the normative Unicode combining class of that combining mark typically corresponds to its positional placement with regard to a base letter, as described earlier. However, in the case of the combining marks representing vowels (and sometimes consonants) in the Brahmi-derived scripts, all of the combining marks are given the normative combining class of zero, regardless of their positional placement within an *aksara*. The placement and rendering of a class zero combining mark cannot be derived from its combining class alone, but rather depends on having more information about the particulars of the script involved. In some instances, the position may migrate in different historical periods for a script or may even differ depending on font style.

Such matters are not treated as normative character properties in the Unicode Standard, because they are more properly considered properties of the glyphs and fonts used for rendering. However, to assist implementers, earlier versions of the Unicode Standard did subcategorize some class zero combining marks, pointing out significant types that need to be handled consistently. That earlier subcategorization is extended and refined in this section.

***Reordrant Class Zero Combining Marks.*** In many instances in Indic scripts, a vowel is represented in logical order *after* the consonant of a syllable, but is displayed *before* (to the left of) the consonant when rendered. Such combining marks are termed *reordrant* to reflect their visual reordering to the left of a consonant (or, in some instances, a consonant cluster). Special handling is required for selection and editing of these marks. In particular, the possibility that the combining mark may be reordered left past a cluster, and not simply past the immediate preceding character in the backing store, requires attention to the details for each script involved.

The *visual* reordering of these reordrant class zero combining marks has nothing to do with the reordering of combining character sequences in the Canonical Ordering Algorithm. All of these marks are *class zero* and thus are *never* reordered by the Canonical Ordering Algorithm or during normalization. The reordering is purely a presentational issue for *glyphs* during rendering of text.

*Table 4-2* lists reordrant class zero combining marks in the Unicode Standard.

**Table 4-2.** Class Zero Combining Marks—Reordrant

| Script | Code Points |
|---|---|
| Devanagari | 093F |
| Bengali | 09BF, 09C7, 09C8 |
| Gurmukhi | 0A3F |
| Gujarati | 0ABF |
| Oriya | 0B47 |
| Tamil | 0BC6, 0BC7, 0BC8 |
| Malayalam | 0D46, 0D47, 0D48 |
| Sinhala | 0DD9, 0DDA, 0DDB |
| Myanmar | 1031 |
| Khmer | 17C1, 17C2, 17C3 |
| Balinese | 1B3E, 1B3F |
| Buginese | 1A19, 1A1B |

In addition, there are historically related vowel characters in the Thai and Lao scripts that, for legacy reasons, are not treated as combining marks. Instead, for Thai and Lao, these vowels are represented in the backing store in visual order and require no reordering for rendering. The trade-off is that they have to be rearranged logically for searching and sorting. Because of that processing requirement, these characters are given a formal character property assignment, the Logical_Order_Exception property, as listed in *Table 4-3*. See PropList.txt in the Unicode Character Database.

**Table 4-3.** Thai and Lao Logical Order Exceptions

| Script | Code Points |
|---|---|
| Thai | 0E40..0E44 |
| Lao | 0EC0..0EC4 |

***Split Class Zero Combining Marks.*** In addition to the reordrant class zero combining marks, there are a number of class zero combining marks whose representative glyph typically consists of two parts, which are split into different positions with respect to the consonant (or consonant cluster) in an aksara. Sometimes these glyphic pieces are rendered both to the left and the right of a consonant. Sometimes one piece is rendered above or below the consonant and the other piece is rendered to the left or the right. Particularly in the instances where some piece of the glyph is rendered to the left of the consonant, these split class zero combining marks pose similar implementation problems as for the reordrant marks.

*Table 4-4* lists split class zero combining marks in the Unicode Standard, subgrouped by positional patterns.

**Table 4-4.** Class Zero Combining Marks—Split

| Glyph Positions | Script | Code Points |
|---|---|---|
| **Left and right** | Bengali | 09CB, 09CC |
| | Oriya | 0B4B |
| | Tamil | 0BCA, 0BCB, 0BCC |
| | Malayalam | 0D4A, 0D4B, 0D4C |
| | Sinhala | 0DDC, 0DDE |
| | Khmer | 17C0, 17C4, 17C5 |
| | Balinese | 1B40, 1B41 |
| **Left and top** | Oriya | 0B48 |
| | Sinhala | 0DDA |
| | Khmer | 17BE |
| **Left, top, and right** | Oriya | 0B4C |
| | Sinhala | 0DDD |
| | Khmer | 17BF |
| **Top and right** | Oriya | 0B57 |
| | Kannada | 0CC0, 0CC7, 0CC8, 0CCA, 0CCB |
| | Limbu | 1925, 1926 |
| | Balinese | 1B43 |
| **Top and bottom** | Telugu | 0C48 |
| | Tibetan | 0F73, 0F76, 0F77, 0F78, 0F79, 0F81 |
| | Balinese | 1B3C |
| Top, bottom, and right | Balinese | 1B3D |
| Bottom and right | Balinese | 1B3B |

One should pay very careful attention to all split class zero combining marks in implementations. Not only do they pose issues for rendering and editing, but they also often have canonical equivalences defined involving the separate pieces, when those pieces are also encoded as characters. As a consequence, the split combining marks may constitute exceptional cases under normalization. Some of the Tibetan split combining marks are discouraged from use.

The split vowels also pose difficult problems for understanding the standard, as the *phonological* status of the vowel phonemes, the *encoding* status of the characters (including any canonical equivalences), and the *graphical* status of the glyphs are easily confused, both for native users of the script and for engineers working on implementations of the standard.

***Subjoined Class Zero Combining Marks.*** Brahmi-derived scripts that are not represented in the Unicode Standard with a virama may have class zero combining marks to represent subjoined forms of consonants. These correspond graphologically to what would be represented by a sequence of virama + consonant in other related scripts. The subjoined consonants do not pose particular rendering problems, at least not in comparison to other combining marks, but they should be noted as constituting an exception to the normal pattern in Brahmi-derived scripts of consonants being represented with base letters. This exception needs to be taken into account when doing linguistic processing or searching and sorting.

*Table 4-5* lists subjoined class zero combining marks in the Unicode Standard.

**Table 4-5.** Class Zero Combining Marks—Subjoined

| Script | Code Points |
|---|---|
| Tibetan | 0F90..0F97, 0F99..0FBC |
| Limbu | 1929, 192A, 192B |

These Limbu consonants, while logically considered subjoined combining marks, are rendered mostly at the lower right of a base letter, rather than directly beneath them.

***Strikethrough Class Zero Combining Marks.*** The Kharoshthi script is unique in having some class zero combining marks for vowels that are struck through a consonant, rather than being placed in a position around the consonant. These are also called out in *Table 4-6* specifically as a warning that they may involve particular problems for implementations.

**Table 4-6.** Class Zero Combining Marks—Strikethrough

| Script | Code Points |
|---|---|
| Kharoshthi | 10A01, 10A06 |

## 4.4 Directionality—Normative

Directional behavior is interpreted according to the Unicode Bidirectional Algorithm (see Unicode Standard Annex #9, "The Bidirectional Algorithm"). For this purpose, all characters of the Unicode Standard possess a normative *directional* type. The directional types left-to-right and right-to-left are called *strong types*, and characters of these types are called strong directional characters. Left-to-right types include most alphabetic and syllabic characters as well as all Han ideographic characters. Right-to-left types include Arabic, Hebrew, Phoenician, Syriac, Thaana, N'Ko, and Kharoshthi, and most punctuation specific to those scripts. In addition, the Unicode Bidirectional Algorithm uses *weak types* and *neutrals*. Interpretation of directional properties according to the Unicode Bidirectional Algorithm is needed for layout of right-to-left scripts such as Arabic and Hebrew.

For the directional types of Unicode characters, see the Unicode Character Database.

## 4.5 General Category—Normative

The Unicode Character Database defines a *General Category* for all Unicode code points. This General Category constitutes a partition of the code points into several major classes, such as letters, punctuation, and symbols, and further subclasses for each of the major classes.

Each Unicode code point is assigned a General Category value. Each value of the General Category is defined as a two-letter abbreviation, where the first letter gives information about a major class and the second letter designates a subclass of that major class. In each class, the subclass "other" merely collects the remaining characters of the major class. For example, the subclass "No" (Number, other) includes all characters of the Number class that are not a decimal digit or letter. These characters may have little in common besides their membership in the same major class.

*Table 4-7* enumerates the General Category values, giving a short description of each value. See *Table 2-3* for the relationship between General Category values and basic types of code points.

A common use of the General Category of a Unicode character is to assist in determination of boundaries in text, as in Unicode Standard Annex #29, "Text Boundaries." Other common uses include determining language identifiers for programming, scripting, and markup, as in Unicode Standard Annex #31, "Identifier and Pattern Syntax," and in regular expression languages such as Perl. For more information, see Unicode Technical Standard #18, "Unicode Regular Expression Guidelines."

This property is also used to support common APIs such as `isDigit()`. Common functions such as `isLetter()` and `isUppercase()` do not extend well to the larger and more complex repertoire of Unicode. While it is possible to naively extend these functions to Unicode using the General Category and other properties, they will not work for the entire

**Table 4-7.** General Category

| | |
|---|---|
| Lu | Letter, uppercase |
| Ll | Letter, lowercase |
| Lt | Letter, titlecase |
| Lm | Letter, modifier |
| Lo | Letter, other |
| Mn | Mark, nonspacing |
| Mc | Mark, spacing combining |
| Me | Mark, enclosing |
| Nd | Number, decimal digit |
| Nl | Number, letter |
| No | Number, other |
| Pc | Punctuation, connector |
| Pd | Punctuation, dash |
| Ps | Punctuation, open |
| Pe | Punctuation, close |
| Pi | Punctuation, initial quote (may behave like Ps or Pe depending on usage) |
| Pf | Punctuation, final quote (may behave like Ps or Pe depending on usage) |
| Po | Punctuation, other |
| Sm | Symbol, math |
| Sc | Symbol, currency |
| Sk | Symbol, modifier |
| So | Symbol, other |
| Zs | Separator, space |
| Zl | Separator, line |
| Zp | Separator, paragraph |
| Cc | Other, control |
| Cf | Other, format |
| Cs | Other, surrogate |
| Co | Other, private use |
| Cn | Other, not assigned (including noncharacters) |

range of Unicode characters and range of tasks for which people use them. For more appropriate approaches, see Unicode Standard Annex #31, "Identifier and Pattern Syntax"; Unicode Standard Annex #29, "Text Boundaries"; *Section 5.18, Case Mappings*; and *Section 4.10, Letters, Alphabetic, and Ideographic*.

## 4.6 Numeric Value—Normative

*Numeric value* is a normative property of characters that represent *numbers*. This group includes characters such as fractions, subscripts, superscripts, Roman numerals, currency numerators, encircled numbers, and script-specific digits. In many traditional numbering systems, letters are used with a numeric value. Examples include Greek and Hebrew letters

as well as Latin letters used in outlines (II.A.1.b). These special cases are not included here as numbers to prevent simplistic parsers from treating these letters numerically by mistake.

*Decimal digits* form a large subcategory of numbers consisting of those digits that can be used to form decimal-radix numbers. They include script-specific digits, but exclude characters such as Roman numerals and Greek acrophonic numerals. (Note that <1, 5> = 15 = fifteen, but <I, V> = IV = four.) Decimal digits also exclude the compatibility subscript or superscript digits to prevent simplistic parsers from misinterpreting their values in context. Numbers other than decimal digits can be used in numerical expressions and may be interpreted by a numeric parser, but it is up to the implementation to determine such specialized uses.

The Unicode Standard assigns distinct codes to the particular digits that are specific to a given script. Examples are the digits used with the Arabic script, Chinese numbers, or those of the Indic scripts. For naming conventions relevant to Arabic digits, see the introduction to *Section 8.2, Arabic*.

The Unicode Character Database gives the numeric values of Unicode characters that normally represent numbers.

### *Ideographic Numeric Values*

CJK ideographs also may have numeric values. The primary numeric ideographs are shown in *Table 4-8*. When used to represent numbers in decimal notation, zero is represented by U+3007. Otherwise, zero is represented by U+96F6.

**Table 4-8.** Primary Numeric Ideographs

| Code Point | Value |
|---|---|
| U+96F6 | 0 |
| U+4E00 | 1 |
| U+4E8C | 2 |
| U+4E09 | 3 |
| U+56DB | 4 |
| U+4E94 | 5 |
| U+516D | 6 |
| U+4E03 | 7 |
| U+516B | 8 |
| U+4E5D | 9 |
| U+5341 | 10 |
| U+767E | 100 |
| U+5343 | 1,000 |
| U+4E07 | 10,000 |
| U+5104 | 100,000,000 (10,000 × 10,000) |
| U+4EBF | 100,000,000 (10,000 × 10,000) |
| U+5146 | 1,000,000,000,000 (10,000 × 10,000 × 10,000) |

Ideographic accounting numbers are commonly used on checks and other financial instruments to minimize the possibilities of misinterpretation or fraud in the representation of

numerical values. The set of accounting numbers varies somewhat between Japanese, Chinese, and Korean usage. *Table 4-9* gives a fairly complete listing of the known accounting characters. Some of these characters are ideographs with other meanings pressed into service as accounting numbers; others are used only as accounting numbers.

**Table 4-9.** Ideographs Used as Accounting Numbers

| Number | Multiple Uses | Accounting Use Only |
|---|---|---|
| 1 | U+58F9, U+58F1 | U+5F0C |
| 2 | | U+8CAE, U+8CB3, U+8D30, U+5F10, U+5F0D |
| 3 | U+53C3, U+53C2 | U+53C1, U+5F0E |
| 4 | U+8086 | |
| 5 | U+4F0D | |
| 6 | U+9678, U+9646 | |
| 7 | U+67D2 | |
| 8 | U+634C | |
| 9 | U+7396 | |
| 10 | U+62FE | |
| 100 | U+964C | U+4F70 |
| 1,000 | U+4EDF | |
| 10,000 | U+842C | |

In Japan, U+67D2 is also pronounced *urusi*, meaning "lacquer," and is treated as a variant of the standard character for "lacquer," U+6F06.

The Unicode Character Database gives the most up-to-date and complete listing of primary numeric ideographs and ideographs used as accounting numbers, including those for CJK repertoire extensions beyond the Unified Repertoire and Ordering.

## 4.7 Bidi Mirrored—Normative

*Bidi Mirrored* is a normative property of characters such as parentheses, whose images are mirrored horizontally in text that is laid out from right to left. For example, U+0028 LEFT PARENTHESIS is interpreted as *opening parenthesis*; in a left-to-right context it will appear as "(", while in a right-to-left context it will appear as the mirrored glyph ")".

Paired delimiters are mirrored even when they are used in unusual ways, as, for example, in the mathematical expressions [a,b) or ]a,b[. If any of these expression is displayed from right to left, then the mirrored glyphs are used. Because of the difficulty in interpreting such expressions, authors of bidirectional text need to make sure that readers can determine the desired directionality of the text from context.

For some mathematical symbols, the "mirrored" form is not an exact mirror image. For example, the direction of the circular arrow in U+2232 CLOCKWISE CONTOUR INTEGRAL reflects the direction of the integration in coordinate space, not the text direction. In a right-to-left context, the integral sign would be mirrored, but the circular arrow would retain its direction. In a similar manner, the bidi-mirrored form of U+221B CUBE ROOT

would be composed of a mirrored radix symbol with a non-mirrored digit "3". For more information, see Unicode Technical Report #25, "Unicode Support for Mathematics."

The list of mirrored characters appears in the Unicode Character Database. Note that mirroring is not limited to paired characters, but that any character with the mirrored property will need two mirrored glyphs—for example, U+222B INTEGRAL. This requirement is necessary to render the character properly in a bidirectional context. It is the default behavior in Unicode text. (For more information, see the "Semantics of Paired Punctuation" subsection in *Section 6.2, General Punctuation*.)

This property is not to be confused with the related *Bidi Mirroring Glyph* property, an informative property, that can assist in rendering mirrored characters in a right-to-left context. For more information, see BidiMirroring.txt in the Unicode Character Database.

## 4.8 Name—Normative

All Unicode characters have unique names that serve as formal, unique identifiers for each character. Unicode character names contain only uppercase Latin letters A through Z, digits, space, and hyphen-minus; this convention makes it easy to generate computer-language identifiers automatically from the names. (See UAX #34, "Unicode Named Character Sequences," for more information on the character name syntax.) The character names in the Unicode Standard are identical to those of the English-language edition of ISO/IEC 10646.

Where possible, character names are derived from existing conventional names of a character or symbol in English, but in many cases the character names nevertheless differ from traditional names widely used by relevant user communities. The character names of symbols and punctuation characters often describe the shape, rather than the function because these characters are used in many different contexts.

Character names are listed in *Chapter 17, Code Charts*.

***Ideographs and Hangul Syllables.*** Names for ideographs and Hangul syllables are derived algorithmically. Unified CJK ideographs are named CJK UNIFIED IDEOGRAPH-*x*, where *x* is replaced with the hexadecimal Unicode code point—for example, CJK UNIFIED IDEOGRAPH-4E00. Similarly, compatibility CJK ideographs are named "CJK COMPATIBILITY IDEOGRAPH-*x*". The names of Hangul syllables are generated as described in "Hangul Syllable Names" in *Section 3.12, Conjoining Jamo Behavior*.

***Control Codes.*** In the Unicode Standard, all control codes (characters with General Category=Cc) have been given the special value <control> instead of a unique character name. However, for control characters, the values of the informative Unicode 1.0 name property match the names of control functions from ISO/IEC 6429. (See *Section 4.9, Unicode 1.0 Names*.) ISO/IEC 10646 does not define names for control codes.

***Named Character Sequences.*** In some instances, character sequences are given a normative name in the Unicode Standard. These characters are from the same namespace as char-

acter names and are unique. For details, see Unicode Standard Annex #34, "Unicode Named Character Sequences." Named character sequences are not listed in *Chapter 17, Code Charts*.

***Stability.*** Once assigned, a character name is immutable. It will never be changed in subsequent versions of the Unicode Standard. Implementers and users can rely on the fact that a character name uniquely represents a given character. The same is true for named character sequences.

***Character Name Aliases.*** Sometimes errors in a character name are discovered after publication. Because character names are immutable, such errors are not corrected by changing the names. However, in some instances, the Unicode Standard publishes a corrected name as a normative *character name alias*. Character name aliases are themselves immutable once published and are also guaranteed to be unique in the namespace for character names. A character may have more than one normative character name alias.

A normative character name alias is different from the informative aliases listed in *Chapter 17, Code Charts*. Informative aliases merely point out other common names in use for a given character. They are not immutable, are not guaranteed to be unique, and therefore cannot serve as an identifier for a character. Their main purpose is to help readers of the standard to locate particular characters.

***User Interfaces.*** A list of Unicode character names may not always be the most appropriate set of choices to present to a user in a user interface. Many common characters do not have a single name for all English-speaking user communities and, of course, their native name in another language is likely to be different altogether. The names of many characters in the Unicode Standard are based on specific Latin transcription of the sounds they represent. There are often competing transcription schemes. For all these reasons, it can be more effective for a user interface to use names that were translated or otherwise adjusted to meet the expectations of the targeted user community. By also listing the formal character name, a user interface could ensure that users can unambiguously refer to the character by the name documented in the Unicode Standard.

***Character Name Matching.*** Character names are constructed so that they can easily be transposed into identifiers in another context, such as a computer language. In matching identifiers constructed from character names, it is possible to ignore case, whitespace, and all medial hyphens except the hyphen in U+1180 HANGUL JUNGSEONG O-E and still result in a unique match. For example, "ZERO WIDTH SPACE" is equivalent to "zero-width-space" or "ZeroWidthSpace", but "CHARACTER -A" is not equivalent to "CHARACTER A" because the hyphen is not medial.

Because Unicode character names do not contain an underscore ("_"), a common strategy is to replace the hyphen, the space, or both by "_" when constructing an identifier from a character name.

## 4.9 Unicode 1.0 Names

The *Unicode 1.0 character name* is an informative property of the characters defined in Version 1.0 of the Unicode Standard. The names of Unicode characters were changed in the process of merging the standard with ISO/IEC 10646. The Version 1.0 character names can be obtained from the Unicode Character Database. Where the Version 1.0 character name provides additional useful information, it is listed in *Chapter 17, Code Charts*. For example, U+00B6 PILCROW SIGN has its Version 1.0 name, PARAGRAPH SIGN, listed for clarity.

The status of the Version 1.0 character names in the case of control codes differs from that for other characters. *The Unicode Standard, Version 1.0,* gave names to the C0 control codes, U+0000..U+001F, U+007F, based on then-current practice for reference to ASCII control codes. Unicode 1.0 gave no names to the C1 control codes, U+0080..U+009F. Currently, the Unicode 1.0 character name property defined in the Unicode Character Database has been updated for the control codes to reflect the ISO/IEC 6429 standard names for control functions. Those names can be seen as annotations in *Chapter 17, Code Charts*. In a few instances, because of updates to ISO/IEC 6429, those names may differ from the names that actually occurred in Unicode 1.0. For example, the Unicode 1.0 name of U+0009 was HORIZONTAL TABULATION, but the ISO/IEC 6429 name for this function is CHARACTER TABULATION, and the commonly used alias is, of course, merely *tab*.

## 4.10 Letters, Alphabetic, and Ideographic

***Letters and Syllables.*** The concept of a letter is used in many contexts. Computer language standards often characterize identifiers as consisting of letters, syllables, ideographs, and digits, but do not specify exactly what a "letter," "syllable," "ideograph," or "digit" is, leaving the definitions implicitly either to a character encoding standard or to a locale specification. The large scope of the Unicode Standard means that it includes many writing systems for which these distinctions are not as self-evident as they may once have been for systems designed to work primarily for Western European languages and Japanese. In particular, while the Unicode Standard includes various "alphabets" and "syllabaries," it also includes writing systems that fall somewhere in between. As a result, no attempt is made to draw a sharp property distinction between letters and syllables.

***Alphabetic.*** The alphabetic property is an informative property of the primary units of alphabets and/or syllabaries, whether combining or noncombining. Included in this group would be composite characters that are canonical equivalents to a combining character sequence of an alphabetic base character plus one or more combining characters; letter digraphs; contextual variants of alphabetic characters; ligatures of alphabetic characters; contextual variants of ligatures; modifier letters; letterlike symbols that are compatibility equivalents of single alphabetic letters; and miscellaneous letter elements. Notably, U+00AA FEMININE ORDINAL INDICATOR and U+00BA MASCULINE ORDINAL INDICATOR are

simply abbreviatory forms involving a Latin letter and should be considered alphabetic rather than nonalphabetic symbols.

***Ideographic.*** The ideographic property is an informative property defined in the Unicode Character Database. The ideographic property is used, for example, in determining line breaking behavior. Characters with the ideographic property include Unified CJK Ideographs, CJK Compatibility Ideographs, and characters from other blocks—for example, U+3007 IDEOGRAPHIC NUMBER ZERO and U+3006 IDEOGRAPHIC CLOSING MARK. For more information about Han ideographs, see *Section 12.1, Han*. For more about ideographs and logosyllabaries in general, see *Section 6.1, Writing Systems*.

## 4.11 Properties Related to Text Boundaries

The determination of text boundaries, such as word breaks or line breaks, involves contextual analysis of potential break points and the characters that surround them. Such an analysis is based on the classification of all Unicode characters by their default interaction with each particular type of text boundary. For example, the Line_Break property defines the default behavior of Unicode characters with respect to line breaking.

A number of characters have special behavior in the context of determining text boundaries. These characters are described in more detail in the subsection on "Line and Word Breaking" in *Section 16.2, Layout Controls*. For more information about text boundaries and these characters, see Unicode Standard Annex #14, "Line Breaking Properties," and Unicode Standard Annex #29, "Text Boundaries."

## 4.12 Characters with Unusual Properties

The behavior of most characters does not require special attention in this standard. However, the characters in *Table 4-10* exhibit special behavior. Many other characters behave in special ways but are not noted here, either because they do not affect surrounding text in the same way or because their use is intended for well-defined contexts. Examples include the compatibility characters for block drawing, the symbol pieces for large mathematical operators, and many punctuation symbols that need special handling in certain circumstances. Such characters are more fully described in the following chapters.

**Table 4-10.** Unusual Properties

| Function | Description | Code Point and Name |
|---|---|---|
| Fraction formatting | *Section 6.2* | 2044 FRACTION SLASH |
| Special behavior with nonspacing marks | *Section 2.11*, *Section 6.2*, and *Section 16.2* | 0020 SPACE<br>00A0 NO-BREAK SPACE |
| Double nonspacing marks | *Section 7.9* | 035C COMBINING DOUBLE BREVE BELOW<br>035D COMBINING DOUBLE BREVE<br>035E COMBINING DOUBLE MACRON<br>035F COMBINING DOUBLE MACRON BELOW<br>0360 COMBINING DOUBLE TILDE<br>0361 COMBINING DOUBLE INVERTED BREVE<br>0362 COMBINING DOUBLE RIGHTWARDS ARROW BELOW |
| Combining half marks | *Section 7.9* | FE20 COMBINING LIGATURE LEFT HALF<br>FE21 COMBINING LIGATURE RIGHT HALF<br>FE22 COMBINING DOUBLE TILDE LEFT HALF<br>FE23 COMBINING DOUBLE TILDE RIGHT HALF |
| Cursive joining and ligation control | *Section 16.2* | 200C ZERO WIDTH NON-JOINER<br>200D ZERO WIDTH JOINER |
| Collation weighting and sequence interpretation | *Section 16.2* | 034F COMBINING GRAPHEME JOINER |
| Bidirectional ordering | *Section 16.2* | 200E LEFT-TO-RIGHT MARK<br>200F RIGHT-TO-LEFT MARK<br>202A LEFT-TO-RIGHT EMBEDDING<br>202B RIGHT-TO-LEFT EMBEDDING<br>202C POP DIRECTIONAL FORMATTING<br>202D LEFT-TO-RIGHT OVERRIDE<br>202E RIGHT-TO-LEFT OVERRIDE |
| Mathematical expression formatting | *Section 15.5* | 2061 FUNCTION APPLICATION<br>2062 INVISIBLE TIMES<br>2063 INVISIBLE SEPARATOR |
| Deprecated alternate formatting | *Section 16.3* | 206A INHIBIT SYMMETRIC SWAPPING<br>206B ACTIVATE SYMMETRIC SWAPPING<br>206C INHIBIT ARABIC FORM SHAPING<br>206D ACTIVATE ARABIC FORM SHAPING<br>206E NATIONAL DIGIT SHAPES<br>206F NOMINAL DIGIT SHAPES |
| Prefixed format control | *Section 8.2* and *Section 8.3* | 0600 ARABIC NUMBER SIGN<br>0601 ARABIC SIGN SANAH<br>0602 ARABIC FOOTNOTE MARKER<br>0603 ARABIC SIGN SAFHA<br>06DD ARABIC END OF AYAH<br>070F SYRIAC ABBREVIATION MARK |

**Table 4-10.** Unusual Properties (Continued)

| Function | Description | Code Point and Name |
|---|---|---|
| Brahmi-derived script dead-character formation | *Chapter 9*, *Chapter 10*, and *Chapter 11* | 094D DEVANAGARI SIGN VIRAMA<br>09CD BENGALI SIGN VIRAMA<br>0A4D GURMUKHI SIGN VIRAMA<br>0ACD GUJARATI SIGN VIRAMA<br>0B4D ORIYA SIGN VIRAMA<br>0BCD TAMIL SIGN VIRAMA<br>0C4D TELUGU SIGN VIRAMA<br>0CCD KANNADA SIGN VIRAMA<br>0D4D MALAYALAM SIGN VIRAMA<br>0DCA SINHALA SIGN AL-LAKUNA<br>0E3A THAI CHARACTER PHINTHU<br>1039 MYANMAR SIGN VIRAMA<br>1714 TAGALOG SIGN VIRAMA<br>1734 HANUNOO SIGN PAMUDPOD<br>17D2 KHMER SIGN COENG<br>1B44 BALINESE ADEG ADEG<br>A806 SYLOTI NAGRI SIGN HASANTA<br>10A3F KHAROSHTHI VIRAMA |
| Historical viramas with other functions | *Section 10.2* and *Section 10.4* | 0F84 TIBETAN MARK HALANTA<br>193B LIMBU SIGN SA-I |
| Mongolian variation selectors | *Section 13.2* | 180B MONGOLIAN FREE VARIATION SELECTOR ONE<br>180C MONGOLIAN FREE VARIATION SELECTOR TWO<br>180D MONGOLIAN FREE VARIATION SELECTOR THREE<br>180E MONGOLIAN VOWEL SEPARATOR |
| Generic variation selectors | *Section 16.4* | FE00..FE0F VARIATION SELECTOR-1..VARIATION SELECTOR-16<br>E0100..E01EF VARIATION SELECTOR-17..VARIATION SELECTOR-256 |
| Tag characters | *Section 16.9* | E0001 LANGUAGE TAG<br>E0020..E007F LANGUAGE TAG SPACE..CANCEL TAG |
| Ideographic variation indication | *Section 6.2* | 303E IDEOGRAPHIC VARIATION INDICATOR |
| Ideographic description | *Section 12.2* | 2FF0..2FFB IDEOGRAPHIC DESCRIPTION CHARACTER LEFT TO RIGHT..IDEOGRAPHIC DESCRIPTION CHARACTER OVERLAID |
| Interlinear annotation | *Section 16.8* | FFF9 INTERLINEAR ANNOTATION ANCHOR<br>FFFA INTERLINEAR ANNOTATION SEPARATOR<br>FFFB INTERLINEAR ANNOTATION TERMINATOR |
| Object replacement | *Section 16.8* | FFFC OBJECT REPLACEMENT CHARACTER |
| Code conversion fallback | *Section 16.8* | FFFD REPLACEMENT CHARACTER |

**Table 4-10.** Unusual Properties (Continued)

| Function | Description | Code Point and Name |
|---|---|---|
| Musical format control | *Section 15.11* | 1D173 MUSICAL SYMBOL BEGIN BEAM<br>1D174 MUSICAL SYMBOL END BEAM<br>1D175 MUSICAL SYMBOL BEGIN TIE<br>1D176 MUSICAL SYMBOL END TIE<br>1D177 MUSICAL SYMBOL BEGIN SLUR<br>1D178 MUSICAL SYMBOL END SLUR<br>1D179 MUSICAL SYMBOL BEGIN PHRASE<br>1D17A MUSICAL SYMBOL END PHRASE |
| Line break controls | *Section 16.2* | 00AD SOFT HYPHEN<br>200B ZERO WIDTH SPACE<br>2060 WORD JOINER |
| Byte order signature | *Section 16.8* | FEFF ZERO WIDTH NO-BREAK SPACE |