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

Appendix F

Unicode Encoding Stability Policies

Unlike many other standards, the Unicode Standard is continually expanding—new characters are added to meet a variety of uses, ranging from technical symbols to letters for archaic languages. Character properties are also expanded or revised to meet implementation requirements.

In each new version of the Unicode Standard, the Unicode Consortium may add characters or make certain changes to characters that were encoded in a previous version of the standard. However, the Consortium imposes limitations on the types of changes that can be made in an effort to minimize the impact on existing implementations.

This appendix reproduces the text of the policies of the Unicode Consortium regarding character encoding stability in force at the time of publication. The most up-to-date version of these polices is found on the Unicode Web site. See the subsection "Policies" in *Section B.6, Other Unicode Online Resources*. For more information, see "Stability" in *Section 3.1, Versions of the Unicode Standard*.

While these policies guide the development of the standard, they are not formally part of the specification. Most fundamentally, these policies are intended to ensure that text encoded in one version of the standard remains valid and unchanged in later versions. In many cases, the constraints imposed by these stability policies allow implementers to simplify support for particular features of the standard, with the assurance that their implementations will not be invalidated by a later update to the standard.

In this appendix, the notation Unicode N.n+ means "The Unicode Standard, Version N.n and all subsequent versions." See also *Section 3.1, Versions of the Unicode Standard*.

F.1 Encoding Stability Policies for the Unicode Standard

Encoding Stability

Applicable Version: Unicode 2.0+

Once a character is encoded, it will not be moved or removed.

This policy ensures that implementers can always depend on each version of the Unicode Standard being a superset of the previous version. The Unicode Standard may deprecate the character (that is, formally discourage its use), but it will not reallocate, remove, or reassign the character.

- Ordering of characters is handled via collation, not by moving characters to different code points. For more information, see Unicode Technical Standard #10, "Unicode Collation Algorithm," and the Unicode FAQ on the Unicode Web site at http://www.unicode.org/faq/.

Name Stability

Applicable Version: Unicode 2.0+

Once a character is encoded, its character name will not be changed.

Together with the limitations in name syntax, this policy allows implementations to create unique identifiers from character names. The character names are used to distinguish between characters and do not always express the full meaning of each character. They are designed to be used programmatically and, therefore, must be stable.

In some cases the original name chosen to represent the character is inaccurate in one way or another. Any such inaccuracies are dealt with by adding annotations to the character name list (see *Chapter 17, Code Charts*) or by adding descriptive text to the standard.

- It is possible to produce translated names for the characters, to make the information conveyed by the name accessible to non-English speakers.
- In cases of outright errors in character names such as misspellings, a character may be given a formal name alias.

Formal Name Alias Stability

Applicable Version: Unicode 5.0+

Formal aliases, once assigned to a character, will not be changed or removed.

Formal aliases are defined in the file NameAliases.txt in the Unicode Character Database and listed in the character code charts.

Named Character Sequence Stability

Applicable Version: Unicode 5.0+

Named character sequences will not be changed or removed.

This stability guarantee applies both to the name of the named character sequence and to the sequence of characters so named.

Named character sequences are defined in the file NamedSequences.txt in the Unicode Character Database. For more information on named character sequences, see Unicode Standard Annex #34, "Unicode Named Character Sequences."

- There are also provisional named character sequences, which are included in the Unicode Character Database but are not covered by this stability policy.

Name Uniqueness

Applicable Version: Unicode 2.0+

> ***The names of characters, formal aliases, and named character sequences are unique within a shared namespace.***

The names of characters, named character sequences, and formal aliases for characters share a single namespace in which each name uniquely identifies either a single character or a single named character sequence. The definition of uniqueness is not just a simple comparison of the characters—instead, the loose matching rules from UCD.html in the Unicode Character Database are used.

Normalization Stability

Applicable Version: Unicode 3.1+

> ***If a string contains only characters from a given version of the Unicode, and it is put into a normalized form in accordance with that version of Unicode, then the result will also be in that normalized form according to any subsequent version of Unicode.***
>
> ***The result will also be in that normalized form according to any prior version of the standard that contains all of the characters in the string (back to the first applicable version, Unicode 3.1).***

In particular, once a character is encoded, its canonical combining class and decomposition mapping will not be changed in a way that will destabilize normalization. Thus the following constraints will be maintained under all circumstances.

> ***Decomposition Mapping: The decomposition mapping may not be changed except for the correction of exceptional errors which meet all of the following conditions (1–3):***
>
> > ***1. There is a clear and evident error identified in the Unicode Character Database (such as a typographic mistake).***
> >
> > ***2. The error constitutes a clear violation of the identity stability policy.***
> >
> > ***3. The correction of such an error does not violate the following constraints (a–d):***
> >
> > > ***a. No character will be given a decomposition mapping when it did not previously have one.***
> > >
> > > ***b. No decomposition mapping will be removed from a character.***
> > >
> > > ***c. No decomposition mapping will change in type (canonical to compatibility, or vice versa).***

d. The number of characters in a decomposition mapping will not change.

Canonical Combining Class: Once a character is assigned, the canonical combining class will not change.

If an implementation normalizes a string that contains characters that are not assigned in the version of Unicode that it supports, that string might not be in normalized form according to a future version of Unicode. For example, suppose that a Unicode 4.0 program normalizes a string that contains new Unicode 4.1 characters. That string might not be normalized according to Unicode 4.1.

In versions prior to Unicode 4.1, there were exceptional cases where the normalization algorithm had to be applied twice to put a string into normalized form. See Unicode Standard Annex #15, "Normalization Forms."

Identity Stability

Applicable Version: Unicode 1.1+

Once a character is encoded, its properties may still be changed, but not in such a way as to change the fundamental identity of the character.

The Consortium will endeavor to keep the values of the other properties as stable as possible, but some circumstances may arise that require changing them. Particularly in the situation where the Unicode Standard first encodes less well-documented characters and scripts, the exact character properties and behavior initially may not be well known.

As more experience is gathered in implementing the characters, adjustments in the properties may become necessary. Examples of such properties include, but are not limited to, the following:

- General_Category
- Case mappings
- Bidirectional properties
- Compatibility decomposition tags (such as <font> or <compat>)
- Representative glyphs

However, character properties will not be changed in a way that would affect character identity. For example, the representative glyph for U+0061 "A" cannot be changed to "B"; the General_Category for U+0061 "A" cannot be changed to Ll (lowercase letter); and the decomposition mapping for U+00C1 (Á) cannot be changed to <U+0042, U+0301> (B, ´).

Property Value Stability

Values of certain properties are limited by the constraints listed in *Table F-1*. The applicable version is given in the first column.

Table F-1. Constraints on Property Values

Applicable Versions	Constraints
Unicode 1.1.5+	Combining classes are limited to the values 0 to 255.
Unicode 1.1.5+	All characters other than those of General Category M* have the combining class 0.
Unicode 2.0+	Canonical and compatibility mappings are always in canonical order, and the resulting recursive decomposition will also be in canonical order.
Unicode 2.0+	Canonical mappings are always limited either to a single value or to a pair. The second character in the pair cannot itself have a canonical mapping.
Unicode 2.0+	Canonical mappings are always limited so that no string when normalized to NFC expands to more than 3× in length (measured in code units).
Unicode 2.1.3+	The General_Category values will not be further subdivided.
Unicode 3.0.0+	The Bidi_Category values will not be further subdivided.
Unicode 3.0.1+	Case folding mappings are limited so that no string when case folded expands to more than 3× in length (measured in code units).
Unicode 3.1+	The Noncharacter_Code_Point property is an immutable code point property, which means that its property values for all Unicode code points will never change.
Unicode 4.0+	The Bidirectional properties will be assigned so as to preserve canonical equivalence.
Unicode 4.1+	All characters with the Lowercase property and all characters with the Uppercase property have the Alphabetic property.
Unicode 4.1+	The Pattern_Syntax and Pattern_Whitespace properties are immutable code point properties, which means that their property values for all Unicode code points will never change.

These constraints ensure that implementers can simplify or optimize certain aspects of their support for character properties. For further description of these invariants, see the file UCD.html in the Unicode Character Database.

Identifier Stability

Applicable Version: Unicode 3.0+

> ***All strings that are valid default Unicode identifiers will continue to be valid default Unicode identifiers in all subsequent versions of Unicode. Furthermore, default identifiers never contain characters with the Pattern_Syntax or Pattern_Whitespace properties.***

If a string qualifies as an identifier under one version of Unicode, it will qualify as an identifier under all future versions. The reverse is not true—an identifier under Version 5.0 may not be an identifier under Version 4.0—it may contain a character that was unassigned under Unicode 4.0, or (very rarely) a Unicode 4.0 character that was not an identifier character in Unicode 4.0, but became one in Unicode 5.0.

For more information, see Unicode Standard Annex #31, "Identifier and Pattern Syntax."

Case Folding Stability

Applicable Version: Unicode 5.0+

> ***Caseless matching of Unicode strings used for identifiers is stable.***

Case folding stability ensures that identifiers created in different versions of Unicode can be reliably matched in a case-insensitive manner. For more information on identifiers, see Unicode Standard Annex #31, "Identifier and Pattern Syntax." Identifiers commonly exclude compatibility decomposable characters; therefore this policy formally applies only to strings normalized with NFKC. The toCaseFold() operation used for caseless matching is defined by rule R4 under "Default Case Conversion" in *Section 3.13, Default Case Algorithms*.

The formal statement of this policy is:

> ***For each string S containing characters only from a given Unicode version, toCasefold(NFKC(S)) under that version is identical to toCasefold(NFKC(S)) under any later version of Unicode.***

