Technical Reports

L2/08-384

Proposed Update

Unicode Standard Annex #44

UNICODE CHARACTER DATABASE

Version	Unicode 5.2 draft 2
Authors	Mark Davis (markdavis@google.com) and Ken Whistler (ken@unicode.org)
Date	2008-8-27
This Version	http://www.unicode.org/reports/tr44/tr44-3.html
Previous Version	http://www.unicode.org/reports/tr44/tr44-2.html
Latest Version	http://www.unicode.org/reports/tr44/
Revision	3

Summary

This annex consolidates information documenting the Unicode Character Database.

Status

*This is a **draft** document which may be updated, replaced, or superseded by other documents at any time. Publication does not imply endorsement by the Unicode Consortium. This is not a stable document; it is inappropriate to cite this document as other than a work in progress.*

> ***A Unicode Standard Annex (UAX)** forms an integral part of the Unicode Standard, but is published online as a separate document. The Unicode Standard may require conformance to normative content in a Unicode Standard Annex, if so specified in the Conformance chapter of that version of the Unicode Standard. The version number of a UAX document corresponds to the version of the Unicode Standard of which it forms a part.*

Please submit corrigenda and other comments with the online reporting form [Feedback]. Related information that is useful in understanding this annex is found in Unicode Standard Annex #41, "Common References for Unicode Standard Annexes." For the latest version of the Unicode Standard, see [Unicode]. For a list of current Unicode Technical Reports, see [Reports]. For more information about versions of the Unicode Standard, see [Versions]. For any errata which may apply to this annex, see [Errata].

Contents

Reviewers please note that the entire text of this annex has been *extensively* rewritten for this proposed update, to account for the complete incorporation of the former content of UCD.html into the text of the annex. No attempt has been made to use change bars, as in this case change bars would obscure more than they would illuminate about the changed text. Review the entire document as you would new material. The content of Version 5.1.0 of UCD.html has also been extensively edited and reorganized in the interest of text flow and clarity, and substantial new documentation has been added to cover gaps noted during the editorial process. See also Modifications.

Warning: *the information in this annex does not completely describe the use and interpretation of Unicode character properties and behavior. It must be used in conjunction with the data in the other files in the Unicode Character Database, and relies on the notation and definitions supplied in The Unicode Standard. All chapter references are to Version 5.0.0 of the standard unless otherwise indicated.*

1 Introduction

The Unicode Standard is far more than a simple encoding of characters. The standard also associates a rich set of semantics with each encoded character—properties that are required for interoperability and correct behavior in implementations, as well as for Unicode conformance. These semantics are embodied in the Unicode Character Database (UCD), a collection of data files which contain the Unicode character code points and character names. The data files define the Unicode character properties and mappings between Unicode characters (such as case mappings).

This annex describes the UCD and provides a guide to the various documentation files associated with it.

The latest version of the UCD is always located on the Unicode Web site at:

http://www.unicode.org/Public/UNIDATA/

The specific files for the UCD associated with this version of the Unicode Standard (5.2.0) are located at:

http://www.unicode.org/Public/5.2.0/

Stable, archived versions of the UCD associated with all earlier versions of the Unicode Standard can be accessed from:

http://www.unicode.org/ucd/

For a description of the changes in the UCD for this version and earlier versions, see UCD Change History.

2 Conformance

The Unicode Character Database is an integral part of the Unicode Standard.

The UCD contains normative property and mapping information required for implementation of various Unicode algorithms such as the Unicode Bidirectional Algorithm, Unicode Normalization, and Unicode Casefolding. The data files also contain additional informative and provisional character property information.

Each specification of a Unicode algorithm, whether specified in the text of [Unicode] or in one of the Unicode Standard Annexes, designates which data file(s) in the UCD are needed to provide normative property information required by that algorithm.

For information on the meaning and application of the terms, *normative*, *informative*, and *provisional*, see Section 3.5, "Properties" in [Unicode].

For information about the applicable terms of use for the UCD, see the Unicode Terms of Use.

2.1 Simple and Derived Properties

Some character properties in the UCD are simple properties. This status has no bearing on whether or not the properties are normative, but merely indicates that their values are

not derived from some combination of other properties.

Other character properties are derived. This means that their values are derived by rule from some other combination of properties. Generally such rules are stated as set operations, and may or may not include explicit exception lists for individual characters.

Sometimes simple properties are defined merely to make the statement of the rule defining a derived property more compact or general. Such properties are known as contributory properties. Sometimes these contributory properties are defined to encapsulate the messiness inherent in exception lists. At other times, a contributory property may be defined to help stabilize the definition of an important derived property which is subject to stability guarantees.

Derived character properties are not considered second-class citizens among Unicode character properties. They are defined to make implementation of important algorithms easier to state. Included among the first-class derived properties important for such implementations are: Uppercase, Lowercase, XID_Start, XID_Continue, Math, and Default_Ignorable_Code_Point, all defined in DerivedCoreProperties.txt, and derived properties for the optimization of normalization, defined in DerivedNormalizationProps.txt.

Implementations should simply use the derived properties, and should not try to rederive them from lists of simple properties and collections of rules, because of the chances for error and divergence when doing so.

If there are any cases of mismatches between the definition of a derived property as listed in DerivedCoreProperties.txt or similar data files in the UCD, and the definition of a derived property as a set definition rule, the explicit listing in the data file should *always* be taken as the normative definition of the property.

Definitions of property derivations are provided for information only, typically in comment fields in the data files. Such definitions may be refactored, refined, or corrected over time. To ensure that there is never any ambiguity between versions of the standard, even if the definition of a derivation is changed at some point in time, the exact property listing in the data files for any given version of the standard is always the truth for that property value for that version—and will never change for that version.

2.2 Use of Default Values

Unicode character properties have default values. Default values are the value or values that a character property takes for an unassigned code point, or in some instances, for designated subranges of code points, whether assigned or unassigned. For example, the default value of a binary Unicode character property is always "N".

For the formal discussion of default values, see D26 in Section 3.5, "Properties" in [Unicode]. For conventions related to default values in various data files of the UCD, see File Format Conventions. For documentation regarding the particular default values of individual Unicode character properties, see the Property Table.

2.3 Stability of Releases

Just as for the Unicode Standard as a whole, each version of the UCD, once published, is absolutely stable and will *never* change. Each released version is archived in a directory on the Unicode web site, with a directory number associated with that version. URLs pointing to that version's directory are also stable and will be maintained in

perpetuity.

Any errors discovered for a released version of the UCD are noted in [Errata], and if appropriate will be corrected in a *subsequent* version of the UCD.

Stability guarantees constraining how Unicode character properties can (or cannot) change between releases of the UCD are documented in the Unicode Consortium Stability Policies [Stability].

3 Documentation

This annex provides the core documentation for the UCD, but additional information about character properties is available in other parts of the standard and in additional documentation files contained within the UCD itself.

3.1 Character Properties in the Standard

The formal definitions related to character properties used by the Unicode Standard are documented in Section 3.5, "Properties" in [Unicode]. Understanding those definitions and related terminology is essential to the appropriate use of Unicode character properties.

See Section 4.1, "Unicode Character Database", in [Unicode] for a general discussion of the UCD and its use in defining properties. The rest of Chapter 4 provides important explanations regarding the meaning and use of various normative character properties.

3.2 The Character Property Model

For a general discussion of the property model which underlies the definitions associated with the UCD, see UTR #23: The Unicode Character Property Model [UTR23]. That technical report is informative, but over the years various content from it has been incorporated into normative portions of the Unicode Standard, particularly for the definitions in Chapter 3.

UTR #23 also discusses string functions and their relation to character properties.

3.3 NamesList.html

NamesList.html formally describes (in BNF) the format of the NamesList.txt data file, the file which is used to drive the printing of the Unicode code charts and names list. See also Section 17.1, "Character Names List", in [Unicode] for a detailed discussion of the conventions used in the Unicode names list.

3.4 StandardizedVariants.html

StandardizedVariants.html documents standardized variants, showing a representative glyph for each. It is closely tied to the data file, StandardizedVariants.txt, which defines those sequences normatively.

3.5 Unihan and UAX #38

UAX #38, Unicode Han Database (Unihan) [UAX38] describes the format and content of Unihan.txt, the data file which collects together all property information for CJK Unified Ideographs. That annex also specifies in detail which of the Unihan character properties

are normative, informative, or provisional.

It is important to note that Unihan.txt and its associated documentation is aimed *only* at CJK Unified Ideographs. It does not have as its scope legacy East Asian character sets as a whole, which also contain many non-CJK characters. As a result, while Unihan.txt contains extensive and detailed mapping information for CJK Unified Ideographs, it must be supplemented from other sources to establish mapping tables for various important commercial and national character set standards from East Asia.

3.6 Data File Comments

In addition to the specific documentation files for the UCD, individual data files often contain extensive header comments describing their content and any special conventions used in the data.

In some instances, individual property definition sections also contain comments with information about how the property may be derived. Such comments are informative; while they are intended to convey the intent of the derivation, in case of any mismatch between a statement of a derivation in a comment field and the actual listing of the derived property, it is the list which is to be taken as normative. See Simple and Derived Properties.

3.7 Obsolete Documentation Files

UCD.html was formerly the primary documentation file for the UCD. Its content has been wholly incorporated into this document, as of Version 5.2.0.

Unihan.html was formerly the primary documentation file for Unihan.txt. Its content has been wholly incorporated into [UAX38], as of Version 5.1.0.

Much earlier versions of the Unicode Standard contained small, segmented documentation files, UnicodeCharacterDatabase.html, PropList.html, and DerivedProperties.html, which were later incorporated into UCD.html.

4 UCD Files

The heart of the UCD consists of the data files themselves. This section describes the directory structure for the UCD, the format conventions for the data files, and provides documentation for data files not documented elsewhere in this annex.

4.1 Directory Structure

Each version of the UCD is released in a separate, numbered directory under the *Public* directory on the Unicode web site. The content of that directory is complete for that release. It is also stable—once released, it will be archived permanently in that directory, unchanged, at a stable URL.

The specific files for the UCD associated with this version of the Unicode Standard (5.2.0) are located at:

http://www.unicode.org/Public/5.2.0/

4.1.1 UCD Files Proper

The UCD proper is located in the *ucd* subdirectory of the numbered version directory. That directory contains all of the documentation files and most of the data files for the UCD, including some data files for derived properties.

Although all UCD data files are version-specific for a release and most contain internal date and version stamps, the file names of the released data files do not differ from version to version. When linking to a version-specific data file, the version will be indicated by the version number of the directory for the release.

All files for derived extracted properties are in the *extracted* subdirectory of the *ucd* subdirectory. See Derived Extracted Properties for documentation regarding those data files and their content.

A number of auxiliary properties are specified in files in the *auxiliary* subdirectory of the *ucd*
subdirectory. In Version 5.2.0 it contains data files specifying properties associated with UAX #29, Unicode Text Segmentation [UAX29] and with UAX #14, Unicode Line Breaking Algorithm [UAX14], as well as test data for those algorithms. See Segmentation Test Files and Documentation for more information about the test data.

4.1.2 UCD XML Files

The XML version of the UCD is located in the *ucdxml* subdirectory of the numbered version directory. See the UCD in XML for more details.

4.1.3 Charts

The code charts specific to a version of Unicode are archived as a single large pdf file in the *charts*
subdirectory of the numbered version directory. See the readme.txt in that subdirectory and the general web page explaining the Unicode Code Charts for more details.

4.1.4 Beta Review Considerations

Prior to the formal release for any particular version of the UCD, a beta review is conducted. The beta review files are located in the same directory that is later used for the released UCD, but during the beta review period, the subdirectory structure differs somewhat and may contain temporary files, including documentation of diffs between deltas for the beta review. Also, during the beta review, all data file names *are* suffixed with version numbers and delta numbers. So a typical file name during beta review may be "PropList-5.2.0d13.txt" instead of the finally released "PropList.txt".

Notices contained in a ReadMe.txt file in the UCD directory during the beta review period also make it clear that that directory contains preliminary material under review, rather than a final, stable release.

4.1.5 File Directory Differences for Early Releases

The UCD in XML
was introduced in Version 5.1.0, so UCD directories prior to that do not contain the *ucdxml* subdirectory.

UCD directories prior to Version 4.1.0 do not contain the *auxiliary* subdirectory.

UCD directories prior to Version 3.2.0 do not contain the *extracted* subdirectory.

The general structure of the file directory for a released version of the UCD described above applies to Versions 4.1.0 and later. Prior to Version 4.1.0, versions of the UCD were not self-contained, complete sets of data files for that version, but instead only contained any new data files or any data files which had *changed* since the prior release.

The directory naming conventions and the file naming conventions also differed prior to Version 4.1.0. So, for example, Version 4.0.0 of the UCD is contained in a directory named *4.0-Update*, and Version 4.0.1 of the UCD in a directory named *4.0-Update1*. Furthermore, for these earlier versions, the data file names *do* contain explicit version numbers.

It is important to understand and keep these differences in mind when accessing any version of the UCD earlier than Version 4.1.0. Full details on the exact collection of data files associated with the release of any version of the UCD prior to Version 4.1.0 can be found online by referring to the component listings at Enumerated Versions.

4.2 File Format Conventions

Files in the UCD use the format conventions described in this section, unless otherwise specified.

4.2.1 Data Fields

- Each line of data consists of fields separated by semicolons. The fields are numbered starting with zero.
- The first field (0) of each line in the Unicode Character Database files represents a code point or range. The remaining fields (1..n) are properties associated with that code point.
- Leading and trailing spaces within a field are not significant. However, no leading or trailing spaces are allowed in any field of UnicodeData.txt.

4.2.2 Code Points and Sequences

- Code points are expressed as hexadecimal numbers with four to six digits. They are written without the "U+" prefix.
- When a data field contains a sequence of code points, spaces separate the code points.

4.2.3 Code Point Ranges

- A range of code points is specified by the form "X..Y".
- Each code point in a range has the associated property value specified on a data file. For example (from Blocks.txt):

```
0000..007F; Basic Latin
0080..00FF; Latin-1 Supplement
```

- For backward compatibility, ranges in the file UnicodeData.txt are specified by entries for the start and end characters of the range, rather than by the form "X..Y". The start character is indicated by a range identifier, followed by a comma and the

string "First", in angle brackets. This entry takes the place of a regular character name in field 1 for that line. The end character is indicated on the next line with the same range identifier, followed by a comma and the string "Last", in angle brackets:

```
4E00;<CJK Ideograph, First>;Lo;0;L;;;;;N;;;;;
9FC3;<CJK Ideograph, Last>;Lo;0;L;;;;;N;;;;;
```

When using this convention, the names of all characters in the range are algorithmically derivable. See [Unicode] for more information on derivation of character names for such ranges.

4.2.4 Comments

- U+0023 NUMBER SIGN ("#") is used to indicate comments: all characters from the number sign to the end of the line are considered part of the comment, and are disregarded when parsing data.
- In many files, the comments on data lines use a common format, as illustrated here (from Scripts.txt):

```
09B2          ; Bengali # Lo       BENGALI LETTER LA
```

- The first part of a comment using this common format is the General_Category value, provided for information. This is followed by the character name for the code point in the first field (0).
- The printing of the General_Category value is suppressed in instances where it would be redundant, as for DerivedGeneralCategory.txt, in which the value of the property value in the data field is already the General_Category value.
- The symbol "L&" indicates characters of General_Category Lu, Ll, or Lt (uppercase, lowercase, or titlecase letter). For example:

```
0386          ; Greek # L&       GREEK CAPITAL LETTER ALPHA WITH TONOS
```

  This usage of L& is the same as the derived LC value for the General_Category property, as documented in PropertyValueAliases.txt.
- When the data line contains a range of code points, this common format for a comment also indicates a range of character names, separated by "..", as illustrated here:

```
00BC..00BE ; numeric # No [3] VULGAR FRACTION ONE QUARTER..VULGAR FRACTION THREE
```

- Code point ranges in the data files are calculated so that they all have the same General_Category value (or LC). While this convention results in more ranges than are strictly necessary, it makes the contents of the ranges clearer.
- When a code point range occurs, the number of items in the range is included in the comment (in square brackets), immediately following the General_Category value.
- The comments are purely informational, and may change format or be omitted in the future. They should not be parsed for content.

4.2.5 Code Point Labels

- Surrogate code points, private-use characters, control codes, noncharacters, and unassigned code points have no names. When such code points are listed in the data files, for example to list their General_Category values, the comments use code point labels instead of character names. For example (from DerivedCoreProperties.txt):

```
2065..2069    ; Default_Ignorable_Code_Point # Cn   [5] <reserved-2065>..<reserv
```

- Code point labels use one of the tags as documented in the table below, followed by "-" and the code point in hexadecimal. The entire label is then enclosed in angle brackets.

Code Point Label Tags

Tag	General_Category	Note
reserved	Cn	Noncharacter_Code_Point=F
noncharacter	Cn	Noncharacter_Code_Point=T
control	Cc	
private-use	Co	
surrogate	Cs	

4.2.6 Multiple Values

- When a file contains the specification for multiple properties, the second field specifies the name of the property and the third field specifies the property value. For example (from DerivedNormalizationProps.txt):

```
03D2  ; FC_NFKC; 03C5           # L&  GREEK UPSILON WITH HOOK SYMBOL
03D3  ; FC_NFKC; 03CD           # L&  GREEK UPSILON WITH ACUTE AND HOOK SYMBOL
```

4.2.7 Binary Property Values

- For binary properties, the second field specifies the name of the applicable property, with the implied value of the property being "True". Only the ranges of characters with the binary property value of "Y" (= True) are listed. For example (from PropList.txt):

```
1680       ; White_Space # Zs      OGHAM SPACE MARK
180E       ; White_Space # Zs      MONGOLIAN VOWEL SEPARATOR
2000..200A ; White_Space # Zs [11] EN QUAD..HAIR SPACE
```

4.2.8 Default Values

- Entries for a code point may be omitted in a data file if the code point has the default value for the property in question.
- For string properties, including the definition of foldings, the default value is the code point of the character itself.
- For miscellaneous properties which take strings as values, such as the Unicode

Name property, the default value is a null string.

- For binary properties, the default value is always "N" (= False) and is always omitted.
- For other types of properties, the default value is listed in a comment. For example (from Scripts.txt):

```
#  All code points not explicitly listed for Script
#  have the value Unknown (Zzzz).
```

- Default values may also be listed in specially formatted comment lines, using the keyword "@missing". For example:

```
# @missing: 0000..10FFFF; Unknown
```

- Because of the legacy format constraints for UnicodeData.txt, that file contains no specific information about default values for properties. The default values for fields in UnicodeData.txt are documented instead in the UnicodeData.txt entry in the Property Table section below.

Some default values for common catalog, enumeration, and numeric properties are listed in the table below:

Default Values for Properties

Property Name	Default Value
Age	unassigned
Block	No_Block
Canonical_Combining_Class	Not_Reordered (= 0)
Decomposition_Type	None
East_Asian_Width	Neutral
Numeric_Value	NaN
Script	Unknown (= Zzzz)

Default values for some Unicode character properties such as Bidi_Class are complex. See the relevant annexes and other documentation for more details.

4.2.9 Text Encoding

- The data files use UTF-8. Unless otherwise noted, non-ASCII characters only appear in comments.
- For legacy reasons, NamesList.txt is exceptional; it is encoded in Latin-1. See NamesList.html
- Segmentation test data files, such as WordBreakTest.txt, make use of non-ASCII (UTF-8) characters as delimiters for data fields.

4.2.10 Other Conventions

- In some test data files, segments of the test data are distinguished by a line starting with an "@" sign. For example (from NormalizationTest.txt):

```
@Part1 # Character by character test
```

4.2.11 Other File Formats

- The data format for Unihan.txt differs from the standard format. See the discussion of Unihan and UAX #38 earlier in this annex for more information.
- The format for NamesList.txt, which documents the Unicode names list and which is used programmatically to drive the formatting program for Unicode code charts, also differs significantly from regular UCD data files. See NamesList.html
- Index.txt is another exception. It uses a tab-delimited format, with field 0 consisting of an index entry string, and field 1 a code point. Index.txt represents the data printed in Section I.1, "Unicode Names Index" in [Unicode]. It is also used to help maintain the online Unicode Character Name Index.
- The various segmentation test data files make use of "#" to delimit comments, but have distinct conventions for their data fields. See the documentation in their header sections for details of the data field formats for those files.
- The XML version of the UCD has its own file format conventions. In those files, "#" is used to stand for the code point in algorithmically derivable character names such as CJK UNIFIED IDEOGRAPH-4E00, so as to allow for name sharing in more compact representations of the data. See UAX #42, Unicode Character Database in XML [UAX42] for details.

4.3 File List

The exact list of files associated with any particular version of the UCD is available on the web site by referring to the component listings at Enumerated Versions.

The majority of the data files in the UCD provide specifications of character properties for Unicode characters. Those files and their contents are documented in detail in the Property Table section below.

The data files in the *extracted* subdirectory constitute reformatted listings of single character properties extracted from UnicodeData.txt or other primary data files. The reformatting is provided to make it easier to see the particular set of characters having certain values for enumerated properties, or to separate the statement of that property from other properties defined together in UnicodeData.txt. These extracted, derived data files are further documented in the Derived Extracted Properties section below.

The UCD also contains a number of test data files, whose purpose is to provide standard test cases useful in verifying the implementation of complex Unicode algorithms. See the Test Files section below for more documentation.

The remaining files in the Unicode Character Database do not directly specify Unicode properties. The important ones and their functions are listed in the table below. The Status column indicates whether the file (and its content) is considered **N**ormative, **I**nformative, or **P**rovisional.

Other Files in the UCD

File Name	Reference	Status	Description
Index.txt	Chapter 17	I	Index to Unicode characters, as printed in the Unicode Standard.
NamesList.txt	Chapter 17	I	Names list used for production of the code charts, derived from UnicodeData.txt. It contains additional annotations.
NamesList.html	Chapter 17	I	Documents the format of NamesList.txt.
StandardizedVariants.txt	Chapter 16	N	Lists all the standardized variant sequences that have been defined, plus a textual description of their desired appearance.
StandardizedVariants.html	Chapter 16	N	A derived documentation file, generated from StandardizedVariants.txt, plus a list of sample glyphs showing the desired appearance of each standardized variant.
NamedSequences.txt	[UAX34]	N	Lists the names for all approved named sequences.
NamedSequencesProv.txt	[UAX34]	P	Lists the names for all provisional named sequences.

For more information about these files and their use, see the referenced annexes or chapters of Unicode Standard.

4.4 Zipped Files

Starting with Version 4.1.0, zipped versions of all of the UCD files, both data files and documentation files, are available under the *Public/zipped* directory on the Unicode web site. Each collection of zipped files is located there in a numbered subdirectory corresponding to that version of the UCD.

Two different zipped files are provided for each version:

- **Unihan.zip** is the zipped version of the very large Unihan database file, Unihan.txt.
- **UCD.zip**
  is the zipped version of all of the rest of the UCD data files, excluding Unihan.txt.

This bifurcation allows for better management of downloading version-specific information, because Unihan.zip contains all the pertinent CJK-related property information, while UCD.zip contains all of the rest of the UCD property information, for those who may not need the voluminous CJK data.

In versions of the UCD prior to Version 4.1.0, zipped copies of Unihan.txt are provided in the same directory as the UCD data files. These zipped files are only posted for versions of the UCD in which Unihan.txt itself was updated.

4.5 UCD in XML

Starting with Version 5.1.0, a set of XML data files using that schema are also released with each version of the UCD. Those data files make it possible to import and process the UCD property data using standard XML parsing tools, instead of the specialized parsing required for the various individual data files of the UCD.

4.5.1 UAX #42

UAX #42, Unicode Character Database in XML [UAX42] defines an XML schema which is used to incorporate all of the Unicode character property information into the XML version of the UCD. See that annex for details of the schema and conventions regarding the grouping of property values for more compact representations.

4.5.2 XML File List

The XML version of the UCD is contained in the *ucdxml* subdirectory of the UCD. The files are all zipped. The list of files is shown in the table below:

XML File List

File Name	CJK	non-CJK
ucd.all.flat.zip	x	x
ucd.all.grouped.zip	x	x
ucd.nounihan.flat.zip		x
ucd.nounihan.grouped.zip		x
ucd.unihan.flat.zip	x	
ucd.unihan.grouped.zip	x	

The "flat" file versions simply list all attributes with no particular compression. The "grouped" file versions apply the grouping mechanism described in [UAX42] to cut down on the size of the data files.

5 Properties

This section documents the Unicode character properties, relating them in detail to the particular UCD data files in which they are specified. For enumerated properties in particular, this section also documents the actual values which those properties can have.

An index of all the non-CJK character properties by name can be found below in the Property Summary section. For a comparable index of CJK character propertes, see UAX #38, Unicode Han Database (Unihan) [UAX38].

5.1 Property Table

The big property table below specifies the list of character properties defined in each data file of the UCD.

For each data file in the UCD there is a separate section of the property table. In that section, the first column lists the character properties specified in that file.

The data files which define a single property or a small number of properties are listed first, followed by the data files which define a large number of properties: DerivedCoreProperties.txt, DerivedNormalizationProps.txt, PropList.txt, and UnicodeData.txt.

For UnicodeData.txt the default property values are listed in the first column in parentheses after the property name, with the special convention (<code point>) indicating that code point itself is the default value.

The second column in the property table indicates the type of the property, according to the following key:

Property Type Key

Property Type	Symbol	Examples
Catalog	C	Age, Block
Enumeration	E	Joining_Type, Line_Break
Binary	B	Uppercase, White_Space
String	S	Uppercase_Mapping, Case_Folding
Numeric	N	Numeric_Value
Miscellaneous	M	Name, Jamo_Short_Name

- **Catalog**
  properties have enumerated values which are expected to be regularly extended in successive versions of the Unicode Standard. This distinguishes them from Enumeration properties.
- **Enumeration**
  properties have enumerated values which constitute a logical partition space; new values will generally not be added to them in successive versions of the standard.
- **Binary**
  properties are a special case of Enumeration properties, which have exactly two values: Yes and No (or True and False).
- **String**
  properties are typically mappings from a Unicode code point to another Unicode code point or sequence of Unicode code points; examples include case mappings and decomposition mappings.
- **Miscellaneous**
  properties are those properties that do not fit neatly into the other property categories; they currently include character names, comments about characters, and the Unicode_Radical_Stroke property (a combination of numeric values) documented in UAX #38, Unicode Han Database (Unihan) [UAX38].

The third column in the property table indicates the status of the property: **N**ormative or **I**nformative.

Finally, the fourth column in the property table provides a description of the property or properties. This includes information on derivation for derived properties, as well as references to locations in the standard where the property is defined or discussed in detail.

In the section of the table for UnicodeData.txt, the data field numbers are also supplied in parentheses at the start of the description.

For a few entries in the property table, values specified in the fields in a data file only contribute to a full definition of a Unicode character property. For example, the values in field 1 (Name) in UnicodeData.txt do not provide all the values for the Name property for all code points; Jamo.txt
must also be used, and the Name property for CJK Unified Ideographs is derived by rule.

Properties marked as *stabilized*
are no longer actively maintained, nor are they extended as new characters are added.

None of the Unicode character properties should be used simply on the basis of the descriptions in the Property Table without consulting the relevant discussions in the Unicode Standard. Because of the enormous variety of characters in the repertoire of the Unicode Standard, character properties tend not to be self-evident in application, even when the names of the properties may seem familiar from their usage with much smaller legacy character encodings.

Property Table

ArabicShaping.txt			
Joining_Type Joining_Group	E	N	Basic Arabic and Syriac character shaping pro initial, medial and final shapes. See Section 8
BidiMirroring.txt			
Bidi_Mirroring_Glyph	S	I	Informative mapping for substituting charact implementation of bidirectional mirroring. Th of characters with the Bidi_Mirrored property characters that normally are displayed with tl mirrored glyph. See UAX #9: The Unicode Bid Algorithm [UAX9]. Do not confuse this with t property itself.
Blocks.txt			
Block	C	N	List of block names, which are arbitrary nam code points. See Chapter 17 in [Unicode].
CompositionExclusions.txt			
Composition_Exclusion	B	N	Properties for normalization. See UAX #15: U Normalization Forms [UAX15]. Unlike other fi CompositionExclusions simply lists the relev
CaseFolding.txt			
Simple_Case_Folding Case_Folding	S	N	Mapping from characters to their case-folde informative file containing normative derived *Derived from UnicodeData and SpecialCasing* **Note**: The case foldings are omitted in the da the same as the code point itself.
DerivedAge.txt			
Age	C	N/I	This file shows when various code points wer designated/assigned in successive versions c standard. The Age property is normative in the sense tl completely specified based on when a charac the standard. However, DerivedAge.txt is pro information. The value of the Age property fo can be derived by analysis of successive vers and Age is not used normatively in the specif Unicode algorithm.
EastAsianWidth.txt			
East_Asian_Width	E	I	Properties for determining the choice of wide glyphs in East Asian contexts. Property value UAX #11: East Asian Width [UAX11].
HangulSyllableType.txt			
Hangul_Syllable_Type	E	N	The values L, V, T, LV, and LVT used in Chapt

Jamo.txt			
Jamo_Short_Name	M	N	The Hangul Syllable names are derived from Names, as described in Chapter 3 in [Unicode
LineBreak.txt			
Line_Break	E	N	Properties for line breaking. For more inform #14: Unicode Line Breaking Algorithm [UAX1
GraphemeBreakProperty.txt			
Grapheme_Cluster_Break	E	I	See UAX #29: Unicode Text Segmentation [U/
SentenceBreakProperty.txt			
Sentence_Break	E	I	See UAX #29: Unicode Text Segmentation [U/
WordBreakProperty.txt			
Word_Break	E	I	See UAX #29: Unicode Text Segmentation [U/
NameAliases.txt			
Name_Alias	M	N	Normative formal aliases for characters with as described in Chapter 4 in [Unicode]. These match the formal aliases published in the Un code charts.
NormalizationCorrections.txt			
used in Decomposition Mappings	S	N	NormalizationCorrections lists code point dif *Normalization Corrigenda*. For more informa Unicode Normalization Forms [UAX15].
Scripts.txt			
Script	C	I	Script values for use in regular expressions. I information, see UAX #24: Unicode Script Prc
SpecialCasing.txt			
Uppercase_Mapping Lowercase_Mapping Titlecase_Mapping	S	I	Data for producing (in combination with the mappings from UnicodeData.txt) the full case
Unihan.txt (for more information, see [UAX38])			
Numeric_Type Numeric_Value	E	I	The characters tagged with kPrimaryNumeric kAccountingNumeric, and kOtherNumeric are Numeric_Type *numeric*, and the values indica Most characters have these properties based UnicodeData.txt. See Numeric_Type.
Unicode_Radical_Stroke	M	I	The Unicode radical-stroke count, based on kRSUnicode.
DerivedCoreProperties.txt			

Alphabetic	B	I	Characters with the Alphabetic property. For see Chapter 4 in [Unicode]. *Generated from: Lu + Ll + Lt + Lm + Lo + N Other_Alphabetic*
Default_Ignorable_Code_Point	B	N	For programmatic determination of default ig points. New characters that should be ignore (unless explicitly supported) will be assigned permitting programs to correctly handle the of such characters when not otherwise suppo information, see the FAQ Display of Unsuppo and *Section 5.20*, "Default Ignorable Code Po *Generated from Other_Default_Ignorable_Code_Point + Cf (format characters) + Variation_Selector - White_Space - FFF9..FFFB (annotation characters) - 0600..0603, 06DD, 070F (exceptional Cf c should be visible)*
Lowercase	B	I	Characters with the Lowercase property. For see Chapter 4 in [Unicode]. *Generated from: Ll + Other_Lowercase*
Grapheme_Base	B	I	For programmatic determination of graphem boundaries. For more information, see UAX # Segmentation [UAX29]. *Generated from: [0..10FFFF] - Cc - Cf - Cs - - Grapheme_Extend*
Grapheme_Extend	B	I	For programmatic determination of graphem boundaries. For more information, see UAX # Segmentation [UAX29]. *Generated from: Me + Mn + Other_Graphem* **Note:** Depending on an application's interpre (private use), they may be either in Graphem Grapheme_Extend, or in neither.
Grapheme_Link	B	I	Deprecated property, formerly proposed for determination of grapheme cluster boundari *Generated from: Canonical_Combining_Class*
ID_Start	B	I	Used to determine programming identifiers,

ID_Continue	B	I	UAX #31: Unicode Identifier and Pattern Synt
Math	B	I	Characters with the Math property. For more Chapter 4 in [Unicode]. *Generated from: Sm + Other_Math*
Uppercase	B	I	Characters with the Uppercase property. For see Chapter 4 in [Unicode]. *Generated from: Lu + Other_Uppercase*
XID_Start	B	I	Used to determine programming identifiers,
XID_Continue	B	I	UAX #31: Unicode Identifier and Pattern Synt
DerivedNormalizationProps.txt			
Full_Composition_Exclusion	B	N	Characters that are excluded from compositi explicitly in CompositionExclusions.txt, plus of *Singleton Decompositions* and *Non-Starte* as documented in that data file.
Expands_On_NFC Expands_On_NFD Expands_On_NFKC Expands_On_NFKD	B	N	Characters that expand to more than one cha specified normalization form.

FC_NFKC_Closure	S	N	Characters that require extra mappings for c Folding plus Normalization Form KC. Charact this property have a third field with the mapp
			
			*Generated with the following, where **Fold** is d default fold operation (excluding the Turkic-*
			
			`b = NFKC(Fold(a));`
			`c = NFKC(Fold(b));`
			`if (c != b) add mapping from a to c`
			`to the set of mappings that constitute the FC`
			
			Note: The FC_NFKC_Closure value is omitted it is the same as the code point itself.

NFD_Quick_Check NFKD_Quick_Check NFC_Quick_Check NFKC_Quick_Check	E	N	For property values, see Decompositions and (Abbreviated names: NFD_QC, NFKD_QC, NFC
PropList.txt			
ASCII_Hex_Digit	B	N	ASCII characters commonly used for the repr hexadecimal numbers.
Bidi_Control	B	N	Format control characters which have specifi Unicode Bidirectional Algorithm [UAX9].
Dash	B	I	Punctuation characters explicitly called out a Unicode Standard, plus their compatibility eq these have the General_Category value Pd, bu

			General_Category value Sm because of their mathematics.
Deprecated	B	N	For a machine-readable list of deprecated ch characters will ever be removed from the sta usage of deprecated characters is strongly di
Diacritic	B	I	Characters that linguistically modify the mea character to which they apply. Some diacritic combining characters, and some combining diacritics.
Extender	B	I	Characters whose principal function is to ext shape of a preceding alphabetic character. Ty length and iteration marks.
Hex_Digit	B	I	Characters commonly used for the represent hexadecimal numbers, plus their compatibili
Hyphen (Stabilized as of 3.2)	B	I	Dashes which are used to mark connections words, plus the *Katakana middle dot*. The *Ka* functions like a hyphen, but is shaped like a dash.
Ideographic	B	I	Characters considered to be CJKV (Chinese, J and Vietnamese) ideographs.
IDS_Binary_Operator	B	N	Used in Ideographic Description Sequences.
IDS_Trinary_Operator	B	N	Used in Ideographic Description Sequences.
Join_Control	B	N	Format control characters which have specifi control of cursive joining and ligation.
Logical_Order_Exception	B	N	There are a small number of characters that order. These characters require special handl processing.
Noncharacter_Code_Point	B	N	Code points permanently reserved for interna
Other_Alphabetic	B	I	Used in deriving the Alphabetic property.
Other_Default_Ignorable_Code_Point	B	N	Used in deriving the Default_Ignorable_Code
Other_Grapheme_Extend	B	N	Used in deriving the Grapheme_Extend prop
Other_ID_Continue	B	N	Used for backward compatibility of ID_Contin
Other_ID_Start	B	N	Used for backward compatibility of ID_Start.
Other_Lowercase	B	I	Used in deriving the Lowercase property.
Other_Math	B	I	Used in deriving the Math property.
Other_Uppercase	B	I	Used in deriving the Uppercase property.
Pattern_Syntax	B	N	Used for pattern syntax as described in UAX
Pattern_White_Space	B	N	Identifier and Pattern Syntax [UAX31].
Quotation_Mark	B	I	Punctuation characters that function as quota
Radical	B	N	Used in Ideographic Description Sequences.
Soft_Dotted	B	N	Characters with a "soft dot", like *i* or *j*. An acc these characters causes the dot to disappear *above* can be added where required, such as
STerm	B	I	Sentence Terminal. Used in UAX #29: Unicod Segmentation [UAX29].

Terminal_Punctuation	B	I	Punctuation characters that generally mark tl units.
Unified_Ideograph	B	N	Used in Ideographic Description Sequences.
Variation_Selector	B	N	Indicates characters that are Variation Selecto the behavior of these characters, see StandardizedVariants.html, Section 16.4, Var [Unicode], and the Unicode Ideographic Varia [UTS37].
White_Space	B	N	Separator characters and control characters v treated by programming languages as "white purpose of parsing elements. **Note:** ZERO WIDTH SPACE and ZERO WIDTH N are not included, because their functions are line-break control. Their names are unfortun in this respect. **Note:** There are other senses of "whitespace" different set of characters.
UnicodeData.txt			
Name (<none>)	M	N	(1) These names match exactly the names pu code charts of the Unicode Standard. The der Syllable names are omitted from this file; see derivation.
General_Category (Cn)	E	N	(2) This is a useful breakdown into various ch which can be used as a default categorizatior implementations. For the property values, se Category Values.
Canonical_Combining_Class (0)	N	N	(3) The classes used for the Canonical Orderi the Unicode Standard. This property could be either an enumerated property or a numeric principal use of the property is in terms of th For the property value names associated with numeric values, see DerivedCombiningClass. Combining Class Values.
Bidi_Class (L, AL, R)	E	N	(4) These are the categories required by the U Bidirectional Algorithm. For the property valu Bidirectional Class Values. For more informat The Unicode Bidirectional Algorithm [UAX9]. The default property values depend on the co given in DerivedBidiClass.txt
Decomposition_Type (None) Decomposition_Mapping (<code point>)	E S	N	(5) This field contains both values, with the t brackets. The decomposition mappings exac decomposition mappings published with the in the Unicode Standard. For more informatic Decomposition Mappings.

			Note: The decomposition mapping is omitted the decomposition mapping is the same as tl itself.
Numeric_Type (None) Numeric_Value (NaN)	E N	N	(6) If the character has the *decimal digit* prop in Chapter 4 in [Unicode], then the value of tl represented with an integer value in fields 6,
	E N	N	(7) If the character has the *digit* property, bu digit, then the value of that digit is represent value in fields 7 and 8. This covers digits tha handling, such as the compatibility superscri
	E N	N	(8) If the character has the *numeric* property, Chapter 4 in [Unicode], the value of that char represented with a positive or negative integ number in this field. This includes fractions s example, "1/5" for U+2155 VULGAR FRACTIC Some characters have these properties based the Unihan data file. See Numeric_Type, Han.
Bidi_Mirrored (N)	B	N	(9) If the character is a "mirrored" character i text, this field has the value "Y"; otherwise "N "Bidi Mirrored—Normative" of [Unicode]. *Do with the Bidi_Mirroring_Glyph property.*
Unicode_1_Name (<none>)	M	I	(10) Old name as published in Unicode 1.0. T provided when it is significantly different fro name for the character. The value of field 10 characters does not always match the Unicod Instead, field 10 contains ISO 6429 names fo functions, for printing in the code charts.
ISO_Comment (<none>)	M	I	(11) ISO 10646 comment field. It appears in 10646 names list, or contains an asterisk to note.
Simple_Uppercase_Mapping (<code point>)	S	N	(12) Simple uppercase mapping (single chara If a character is part of an alphabet with case has a simple uppercase equivalent, then the equivalent is in this field. The simple mappin character result, where the full mappings ma multi-character results. For more informatio Case Mapping. **Note:** The simple uppercase is omitted in the uppercase is the same as the code point itsel

Simple_Lowercase_Mapping (<code point>)	S	N	(13) Simple lowercase mapping (single chara **Note:** The simple lowercase is omitted in the lowercase is the same as the code point itsel
Simple_Titlecase_Mapping (<code point>)	S	N	(14) Simple titlecase mapping (single charact **Note:** The simple titlecase may be omitted in titlecase is the same as the uppercase.

5.2 Derived Extracted Properties

A number of Unicode character properties have been separated out, reformatted, and listed in range format, one property per file. These files are located under the *extracted* directory of the UCD. The exact list of derived extracted files and the extracted properties they represent are given in the Extracted Properties table below.

The derived extracted files are provided purely as a reformatting of data for properties specified in other data files. In case of any inadvertant mismatch between the primary data files specifying those properties and these lists of extracted properties, the primary data files are taken as definitive.

Extracted Properties

File	Status	Property	Extracted from
DerivedBidiClass.txt	N	Bidi_Class	UnicodeData.txt, field 4
DerivedBinaryProperties.txt	N	Bidi_Mirrored	UnicodeData.txt, field 9
DerivedCombiningClass.txt	N	Canonical_Combining_Class	UnicodeData.txt, field 3
DerivedDecompositionType.txt	N/I	Decomposition_Type	the <tag> in UnicodeData.txt, field 5
DerivedEastAsianWidth.txt	I	East_Asian_Width	EastAsianWidth.txt, field 1
DerivedGeneralCategory.txt	N	General_Category	UnicodeData.txt, field 2
DerivedJoiningGroup.txt	N	Joining_Group	ArabicShaping.txt, field 2
DerivedJoiningType.txt	N	Joining_Type	ArabicShaping.txt, field 1
DerivedLineBreak.txt	N	Line_Break	LineBreak.txt, field 1
DerivedNumericType.txt	N	Numeric_Type	UnicodeData.txt, fields 6 through 8
DerivedNumericValues.txt	N	Numeric_Value	UnicodeData.txt, field 8

For the extraction of Decomposition_Type, characters with canonical decomposition mappings in field 5 of UnicodeData.txt have no tag. For those characters, the extracted value is Decomposition_Type=Canonical. For characters with compatibility decomposition mappings, there are explicit tags in field 5, and the value of Decomposition_Type is equivalent to those tags. The value Decomposition_Type=Canonical is normative. Other values for Decomposition_Type are informative.

Numeric_Value is extracted based on the actual numeric value of the data in field 8 of UnicodeData.txt.

Numeric_Type is extracted as follows. If fields 6, 7, and 8 in UnicodeData.txt are all non-empty, then Numeric_Type=Decimal. Otherwise, if fields 7 and 8 are both non-empty, then Numeric_Type=Digit. Otherwise, if field 8 is non-empty, then Numeric_Type=Numeric. The default value is Numeric_Type=None.

5.3 Property Summary

The following table provides a summary list of the Unicode character properties, excluding most of those specific to Unihan.txt. The properties are roughly organized into groups based on their usage. This grouping is primarily for documentation convenience and except for contributory properties, has no normative implications. The link on each

property leads its description in the Property Table above.

Property Summary Table

General	Normalization	CJK
Name	Canonical_Combining_Class	Ideographic
Name_Alias	Decomposition_Mapping	Unified_Ideograph
Block	Composition_Exclusion	Radical
Age	Full_Composition_Exclusion	IDS_Binary_Operator
General_Category	Decomposition_Type	IDS_Trinary_Operator
Script	FC_NFKC_Closure	Unicode_Radical_Stroke
White_Space	NFC_Quick_Check	**Miscellaneous**
Alphabetic	NFKC_Quick_Check	Math
Hangul_Syllable_Type	NFD_Quick_Check	Quotation_Mark
Noncharacter_Code_Point	NFKD_Quick_Check	Dash
Default_Ignorable_Code_Point	Expands_On_NFC	Hyphen
Deprecated	Expands_On_NFD	STerm
Logical_Order_Exception	Expands_On_NFKC	Terminal_Punctuation
Variation_Selector	Expands_On_NFKD	Diacritic
Case	**Shaping and Rendering**	Extender
Uppercase	Join_Control	Grapheme_Base
Lowercase	Joining_Group	Grapheme_Extend
Lowercase_Mapping	Joining_Type	Grapheme_Link (deprecated)
Titlecase_Mapping	Line_Break	Unicode_1_Name
Uppercase_Mapping	Grapheme_Cluster_Break	ISO_Comment
Case_Folding	Sentence_Break	**Contributory Properties**
Simple_Lowercase_Mapping	Word_Break	Other_Alphabetic
Simple_Titlecase_Mapping	East_Asian_Width	Other_Default_Ignorable_Coc
Simple_Uppercase_Mapping	**Bidirectional**	Other_Grapheme_Extend
Simple_Case_Folding	Bidi_Class	Other_ID_Start
Soft_Dotted	Bidi_Control	Other_ID_Continue
Identifiers	Bidi_Mirrored	Other_Lowercase
ID_Continue	Bidi_Mirroring_Glyph	Other_Math
ID_Start	**Numeric**	Other_Uppercase
XID_Continue	Numeric_Value	Jamo_Short_Name
XID_Start	Numeric_Type	
Pattern_Syntax	Hex_Digit	
Pattern_White_Space	ASCII_Hex_Digit	

5.3.1 Contributory Properties

Contributory properties contain sets of exceptions used in the generation of other properties derived from them. The contributory properties specifically concerned with identifiers and casing contribute to the maintenance of stability guarantees for properties and/or to invariance relationships between related properties. Other contributory properties are simply defined as a convenience for property derivation.

Most contributory properties have names using the pattern "Other_XXX" and are used to derive the corresponding "XXX" property. For example, the Other_Alphabetic property is used in the derivation of the Alphabetic property.

Contributory properties are typically defined in PropList.txt and the corresponding derived property is then listed in DerivedCoreProperties.txt.

Jamo_Short_Name
is an unusual contributory property, both in terms of its name and how it is used. It is defined in its own property file, Jamo.txt, and is used to derive the Name property value for Hangul syllable characters, according to the rules spelled out in Section 3.12, "Conjoining Jamo Behavior" in [Unicode].

Contributory properties are incomplete by themselves and are not intended for independent use. For example, an API returning Unicode property values should implement the derived core properties such as Alphabetic or Default_Ignorable_Code_Point, rather than the corresponding contributory properties, Other_Alphabetic or Other_Default_Ignorable_Code_Point.

5.4 Case and Case Mapping

Case for bicameral scripts and case mapping of characters are complicated topics in the Unicode Standard—both because of their inherent algorithmic complexity and because of the number of characters and special edge cases involved.

This section provides a brief roadmap to discussions about these topics, and specifications and definitions in the standard, as well as explaining which case-related properties are defined in the UCD.

Section 3.13, "Default Case Algorithms" in [Unicode] provides formal definitions for case-related concepts (*cased*, *case-ignorable*, ...), for case conversion (*toUppercase(X)*, ...), and for case detection (*isUppercase(X)*, ...). It also provides the formal definition of caseless matching for the standard, taking normalization into account.

Section 4.2, "Case—Normative", in [Unicode] introduces case and case mapping properties. Table 4-1, "Sources for Case Mapping Information", describes the kind of case-related information that is available in various data files of the UCD. The table below lists those data files again, giving the explicit list of case-related properties defined in each. The link on each property leads its description in the Property Table above.

UCD Files and Case Properties

File Name	Case Properties
UnicodeData.txt	Simple_Uppercase_Mapping, Simple_Lowercase_Mapping, Simple_Titlecase_Mapping
SpecialCasing.txt	Uppercase_Mapping, Lowercase_Mapping, Titlecase_Mapping
CaseFolding.txt	Simple_Case_Folding, Case_Folding
DerivedCoreProperties.txt	Uppercase, Lowercase
PropList.txt	Soft_Dotted, Other_Uppercase, Other_Lowercase

For compatibility with existing parsers, UnicodeData.txt only contains case mappings for characters where they constitute one-to-one mappings; it also omits information about context-sensitive case mappings. Information about these special cases can be found in the separate data file, SpecialCasing.txt, expressed as separate properties.

Section 5.18, "Case Mappings", in [Unicode] discusses various implementation issues for handling case, including language-specific case mapping, as for Greek and for Turkish. That section also describes case folding in particular detail.

The special casing conditions associated with case mapping for Greek, Turkish, and Lithuanian are specified in an additional field in SpecialCasing.txt. For example, the lowercase mapping for sigma in Greek varies according to its position in a word. The condition list does not constitute a formal character property in the UCD, because it is a statement about the context of occurrence of casing behavior for a character or characters, rather than a semantic attribute of those characters. Note that versions of the UCD from Version 3.2.0 to Version 5.0.0 *did* list property aliases for Special_Case_Condition (scc), but this was determined to be an error when the UCD was analyzed for representation in XML; consequently, the Special_Case_Condition property aliases were removed as of Version 5.1.0.

Caseless matching is of particular concern for a number of text processing algorithms, so is also discussed at some length in UAX #31: Unicode Identifier and Pattern Syntax [UAX31] and in UTS #10: Unicode Collation Algorithm [UTS10].

Further information about locale-specific casing conventions can be found in the Unicode Common Locale Data Repository [CLDR].

5.5 Property Value Lists

The following subsections give summaries of property values for certain Enumeration properties. Other property values are documented in other, topically-specific annexes; for example, the Line_Break property values are documented in UAX #14: Unicode Line Breaking Algorithm [UAX14] and the various segmentation-related property values are documented in UAX #29: Unicode Text Segmentation [UAX29].

5.5.1 General Category Values

The General_Category property of a code point provides for the most general classification of that code point. It is usually determined based on the primary characteristic of the assigned character for that code point. For example, is the character a letter, a mark, a number, punctuation, or a symbol, and if so, of what type? Other General_Category values define the classification of code points which are not assigned to regular graphic characters, including such statuses as private-use, control, surrogate code point, and reserved unassigned.

Many characters have multiple uses, and not all such cases can be captured entirely by the General_Category value. For example, the General_Category value of Latin, Greek, or Hebrew letters does not attempt to cover (or preclude) the numerical use of such letters as Roman numerals or in other numerary systems. Conversely, the General_Category of ASCII digits 0..9 as Nd (decimal digit) neither attempts to cover (or preclude) the occasional use of these digits as letters in various orthographies. The General_Category is simply the first-order, most usual categorization of a character.

For more information about the General_Category property, see Chapter 4 in [Unicode].

The values in the General_Category field in UnicodeData.txt make use of the short, abbreviated property value aliases for General_Category. For convenience in reference, the General_Category Values table below lists all the abbreviated and long value aliases for General_Category values, reproduced from PropertyValueAliases.txt, along with a brief description of each category.

General_Category Values

Abbr	Long	Description
Lu	Uppercase_Letter	an uppercase letter
Ll	Lowercase_Letter	a lowercase letter
Lt	Titlecase_Letter	a digraphic character, with first part uppercase
Lm	Modifier_Letter	a modifier letter
Lo	Other_Letter	other letters, including syllables and ideographs
Mn	Nonspacing_Mark	a nonspacing combining mark (zero advance width)
Mc	Spacing_Mark	a spacing combining mark (positive advance width)
Me	Enclosing_Mark	an enclosing combining mark
Nd	Decimal_Number	a decimal digit
Nl	Letter_Number	a letterlike numeric character
No	Other_Number	a numeric character of other type
Pc	Connector_Punctuation	a connecting punctuation mark, like a tie
Pd	Dash_Punctuation	a dash or hyphen punctuation mark
Ps	Open_Punctuation	an opening punctuation mark (of a pair)
Pe	Close_Punctuation	a closing punctuation mark (of a pair)
Pi	Initial_Punctuation	an initial quotation mark
Pf	Final_Punctuation	a final quotation mark
Po	Other_Punctuation	a punctuation mark of other type
Sm	Math_Symbol	a symbol of primarily mathematical use
Sc	Currency_Symbol	a currency sign
Sk	Modifier_Symbol	a non-letterlike modifier symbol
So	Other_Symbol	a symbol of other type
Zs	Space_Separator	a space character (of various non-zero widths)
Zl	Line_Separator	U+2028 LINE SEPARATOR only
Zp	Paragraph_Separator	U+2029 PARAGRAPH SEPARATOR only
Cc	Control	a C0 or C1 control code
Cf	Format	a format control character
Cs	Surrogate	a surrogate code point
Co	Private_Use	a private-use character
Cn	Unassigned	a reserved unassigned code point or a noncharacter

Note that the value gc=Cn does not actually occur in UnicodeData.txt, because that data file does not list unassigned code points.

Characters with the quotation-related General_Category values Pi or Pf may behave like opening punctuation (gc=Ps) or closing punctuation (gc=Pe), depending on usage and quotation conventions.

The symbol "L&" is used to stand for any combination of uppercase, lowercase or titlecase letters (Lu, Ll, or Lt), in the first part of comments in the data files. The LC value for the General_Category property, as documented in PropertyValueAliases.txt also stands for uppercase, lowercase or titlecase letters.

The Unicode Standard does not assign non-default property values to control characters (gc=Cc), except for certain well-defined exceptions involving the Unicode Bidirectional Algorithm, the Unicode Line Breaking Algorithm, and Unicode Text Segmentation. Also, implementations will usually assign behavior to certain line breaking control characters—most notably U+000D and U+000A (CR and LF)—according to platform conventions. See Section 5.8 "Newline Guidelines" in [Unicode] for more information.

5.5.2 Bidirectional Class Values

The values in the Bidi_Class field in UnicodeData.txt make use of the short, abbreviated property value aliases for Bidi_Class. For convenience in reference, the Bidi_Class Values table below lists all the abbreviated and long value aliases for Bidi_Class values, reproduced from PropertyValueAliases.txt, along with a brief description of each category.

Bidi_Class Values

Abbr	Long	Description
L	Left_To_Right	any strong left-to-right character
LRE	Left_To_Right_Embedding	U+202A: the LR embedding control
LRO	Left_To_Right_Override	U+202D: the LR override control
R	Right_To_Left	any strong right-to-left (non-Arabic-type) character
AL	Arabic_Letter	any strong right-to-left (Arabic-type) character
RLE	Right_To_Left_Embedding	U+202B: the RL embedding control
RLO	Right_To_Left_Override	U+202E: the RL override control
PDF	Pop_Directional_Format	U+202C: terminates an embedding or override control
EN	European_Number	any ASCII digit or Eastern Arabic-Indic digit
ES	European_Separator	plus and minus signs
ET	European_Terminator	a terminator in a numeric format context, includes currency signs
AN	Arabic_Number	any Arabic-Indic digit
CS	Common_Separator	commas, colons, and slashes
NSM	Nonspacing_Mark	any nonspacing mark
BN	Boundary_Neutral	most format characters, control codes, or noncharacters
B	Paragraph_Separator	various newline characters
S	Segment_Separator	various segment-related control codes
WS	White_Space	spaces
ON	Other_Neutral	most other symbols and punctuation marks

Please refer to UAX #9: The Unicode Bidirectional Algorithm [UAX9] for an an explanation of the significance of these values when formatting bidirectional text.

5.5.3 Character Decomposition Mapping

The value of the Decomposition_Mapping property for a character is provided in field 5 of UnicodeData.txt. This is a string property, consisting of a sequence of one or more Unicode code points. The default value of the Decomposition_Mapping property is the code point of the character itself. The use of the default value for a character is indicated by leaving field 5 empty in UnicodeData.txt. Informally, the value of the Decomposition_Mapping property for a character is known simply as its *decomposition mapping*. When a character's decomposition mapping is other than the default value, the decomposition mapping is printed out explicitly in the names list for the Unicode code charts.

The prefixed tags supplied with a subset of the decomposition mappings generally indicate formatting information. Where no such tag is given, the mapping is canonical. Conversely, the presence of a formatting tag also indicates that the mapping is a compatibility mapping and not a canonical mapping. In the absence of other formatting information in a compatibility mapping, the tag is used to distinguish it from canonical mappings.

In some instances a canonical mapping or a compatibility mapping may consist of a single character. For a canonical mapping, this indicates that the character is a canonical equivalent of another single character. For a compatibility mapping, this indicates that the character is a compatibility equivalent of another single character.

The compatibility formatting tags used in the UCD are listed in the table below:

Compatibility Formatting Tags

Tag	Description
<font>	Font variant (for example, a blackletter form)
<noBreak>	No-break version of a space or hyphen
<initial>	Initial presentation form (Arabic)
<medial>	Medial presentation form (Arabic)
<final>	Final presentation form (Arabic)
<isolated>	Isolated presentation form (Arabic)
<circle>	Encircled form
<super>	Superscript form
<sub>	Subscript form
<vertical>	Vertical layout presentation form
<wide>	Wide (or zenkaku) compatibility character
<narrow>	Narrow (or hankaku) compatibility character
<small>	Small variant form (CNS compatibility)
<square>	CJK squared font variant
<fraction>	Vulgar fraction form
<compat>	Otherwise unspecified compatibility character

Note:
There is a difference between decomposition and the Decomposition_Mapping property. The Decomposition_Mapping property is a string property whose values (mappings) are defined in UnicodeData.txt, while the decomposition (also termed "full decomposition") is defined in Section 3.7, "Decomposition" in [Unicode] to use those mappings *recursively.*

- The canonical decomposition is formed by recursively applying the canonical mappings, then applying the Canonical Ordering Algorithm.
- The compatibility decomposition is formed by recursively applying the canonical **and** compatibility mappings, then applying the Canonical Ordering Algorithm.

Starting from Unicode 2.1.9, the decomposition mappings in UnicodeData.txt can be used to derive the full decomposition of any single character in canonical order, without the need to separately apply the Canonical Ordering Algorithm. However, canonical ordering of combining character sequences ***must*** still be applied in decomposition when normalizing source text which contains any combining marks.

The normalization of Hangul conjoining jamos and of Hangul syllables depends on algorithmic mapping, as specified in Section 3.12, "Conjoining Jamo Behavior" in [Unicode]. That algorithm specifies the full decomposition of all precomposed Hangul syllables, but effectively it is equivalent to the recursive application of pairwise decomposition mappings, as for all other Unicode characters. Formally, the Decomposition_Mapping property value for a Hangul syllable is the pairwise decomposition and not the full decomposition.

Each character with the Hangul_Syllable_Type value LVT will have a Decomposition_Mapping consisting of a character with an LV value and a character with a T value. Thus for U+CE31 the Decomposition_Mapping is <U+CE20, U+11B8>, rather than <U+110E, U+1173, U+11B8>.

5.5.4 Canonical Combining Class Values

The values in the Canonical_Combining_Class field in UnicodeData.txt are numerical values used in the Canonical Ordering Algorithm. Some of those numerical values also have explicit symbolic labels as property value aliases, to make their intended application more understandable. For convenience in reference, the Canonical_Combining_Class Values table below lists all the long symbolic aliases for Canonical_Combining_Class values, reproduced from PropertyValueAliases.txt, along with a brief description of each category.

Canonical_Combining_Class Values

Value	Long	Description
0	Not_Reordered	Spacing and enclosing marks; also many vowel and consonant signs, even if nonspacing
1	Overlay	Marks which overlay a base letter or symbol
7	Nukta	Diacritic nukta marks in Brahmi-derived scripts
8	Kana_Voicing	Hiragana/Katakana voicing marks
9	Virama	Viramas
10		Start of fixed position classes
199		End of fixed position classes
200	Attached_Below_Left	Marks attached at the bottom left
202	Attached_Below	Marks attached directly below
204		Marks attached at the top right
208		Marks attached to the left
210		Marks attached to the right
212		Marks attached at the top left
214		Marks attached directly above
216	Attached_Above_Right	Marks attached at the top right
218	Below_Left	Distinct marks at the bottom left
220	Below	Distinct marks directly below
222	Below_Right	Distinct marks at the bottom right
224	Left	Distinct marks to the left
226	Right	Distinct marks to the right
228	Above_Left	Distinct marks at the top left
230	Above	Distinct marks directly above
232	Above_Right	Distinct marks at the top right
233	Double_Below	Distinct marks subtending two bases
234	Double_Above	Distinct marks extending above two bases
240	Iota_Subscript	Greek iota subscript only

Some of the Canonical_Combining_Class values in the table are not currently used for any characters but are specified here for completeness. Some values do not have long symbolic aliases, but these two sets are not congruent. Do not assume that absence of a long symbolic alias implies non-use of a particular Canonical_Combining_Class. See DerivedCombiningClass.txt
for a complete listing of the use of Canonical_Combining_Class values for any particular version of the UCD.

Combining marks with ccc=224 (Left) follow their base character in storage, as for all combining marks, but are rendered visually on the left side of them. Note that for all past versions of the UCD and continuing with this version of the UCD, only two tone marks used in certain notations for Hangul syllables have ccc=224. Those marks are actually rendered visually on the left side of the preceding *grapheme cluster*, in the case of Hangul syllables resulting from sequences of conjoining jamos.

Those few instances of combining marks with ccc=Left should be distinguished from the far more numerous examples of left-side vowel signs and vowel letters in Brahmi-derived scripts. The Canonical_Combining_Class value is zero (Not_Reordered) for both ordinary, left-side (reordrant) vowel signs such as U+093F DEVANAGARI VOWEL SIGN I and for Thai-style left-side (Logical_Order_Exception=Yes) vowel letters such as U+0E40 THAI CHARACTER SARA E. The "Not_Reordered" of ccc=Not_Reordered refers to the behavior of the character in terms of the Canonical Ordering Algorithm as part of the definition of Unicode Normalization; it does *not* refer to any issues of visual reordering of glyphs involved in display and rendering. See Section 3.11, "Canonical Ordering Behavior" in [Unicode].

5.5.5 Decompositions and Normalization

Decomposition is specified in Chapter 3, Conformance of [Unicode]. UAX #15, Unicode Normalization Forms [UAX15] specifies the interaction between decomposition and normalization. That annex specifies how the decompositions defined in UnicodeData.txt are used to derive normalized forms of Unicode text.

A number of derived properties related to Unicode normalization are called the "Quick_Check" properties. These are defined to enable various optimizations for implementations of normalization, as explained in Section 14, "Detecting Normalization Forms", in UAX #15, Unicode Normalization Forms [UAX15]. The values for the four Quick_Check properties for all code points are listed in DerivedNormalizationProps.txt. The interpretations of the possible property values are summarized in the table below:

Quick_Check Property Values

Property	Value	Description
NFC_QC, NFKC_QC, NFD_QC, NFKD_QC	No	Characters that cannot ever occur in the respective normalization form.
NFC_QC, NFKC_QC	Maybe	Characters that may occur in the respective normalization, depending on the context.
NFC_QC, NFKC_QC, NFD_QC, NFKD_QC	Yes	All other characters. This is the default value for Quick_Check properties.

5.6 Property and Property Value Aliases

Both Unicode character properties themselves and their values are given symbolic aliases. The formal lists of aliases are provided so that well-defined symbolic values are available for XML formats of the UCD data, for regular expression property tests, and for other programmatic textual descriptions of Unicode data. The aliases for properties are defined in PropertyAliases.txt. The aliases for property values are defined in PropertyValueAliases.txt.

Alias Files in the UCD

File Name	Status	Description
PropertyAliases.txt	N	Names and abbreviations for properties
PropertyValueAliases.txt	N	Names and abbreviations for property values

Aliases are defined as ASCII-compatible identifiers, using only uppercase or lowercase A-Z, digits, and underscore "_". Case is not significant when comparing aliases, but the preferred form used in the data files for longer aliases is to titlecase them.

Aliases may be translated in appropriate environments, and additional aliases may be useful in certain contexts. There is no requirement that only the aliases defined in the alias files of the UCD be used when referring to Unicode character properties or their values; however, their use is recommended for interoperability in data formats or in programmatic contexts.

5.6.1 Property Aliases

In PropertyAliases.txt, the first field specifies an abbreviated symbolic name for the property, and the second field specifies the long symbolic name for the property. These are the preferred aliases. Additional aliases for a few properties are specified in the third or subsequent fields.

The long symbolic name alias is self-descriptive, and is treated as the official name of a Unicode character property. For clarity it is used whenever possible when referring to that property in this annex and elsewhere in the Unicode Standard. For example: "The Line_Break property is discussed in UAX #14, Unicode Line Breaking Algorithm [UAX14]."

The abbreviated symbolic name alias is short and less mnemonic, but is useful for expressions such as "lb=BA" in data or in other contexts where the meaning is clear.

The property aliases specified in PropertyAliases.txt constitute a unique name space. When using these symbolic values, no alias for one property will match an alias for another property.

5.6.2 Property Value Aliases

In PropertyValueAliases.txt, the first field contains the abbreviated alias for a Unicode property, the second field specifies an abbreviated symbolic name for a value of that property, and the third field specifies the long symbolic name for that value of that property. These are the preferred aliases. Additional aliases for some property values may be specified in the fourth or subsequent fields. For example, for binary properties, the abbreviated alias for the True value is "Y", and the long alias is "Yes", but each entry

also specifies "T" and "True" as additional aliases for that value, as shown in the table below:

Binary Property Value Aliases

Long	Abbreviated	Other Aliases
Yes	Y	True, T
No	N	False, F

Not every property value has an associated alias. Property value aliases are typically supplied for catalog and enumeration properties, which have well-defined, enumerated values. It does not make sense to specify property value aliases, for example, for the Numeric_Value property, whose value could be any number, or for a string property such as Simple_Lowercase_Mapping, whose values are mappings from one code point to another.

The Canonical_Combining_Class property requires special handling in PropertyValueAliases.txt. The values of this property are numeric, but they comprise a closed, enumerated set of values. The more important of those values are given symbolic name aliases. In PropertyValueAliases.txt, the second field provides the numeric value, while the third field contains the abbreviated symbolic name alias and the fourth field contains the long symbolic name alias for that numeric value. For example:

```
ccc; 230; A    ; Above
ccc; 232; AR   ; Above_Right
```

Taken by themselves, property value aliases do not constitute a unique name space. The abbreviated aliases, in particular, are often re-used as aliases for values for different properties. All of the binary property value aliases, for example, make use of the same "Y", "Yes", "T", "True" symbols. Property value aliases may also overlap the symbols used for property aliases. For example, "Sc" is the abbreviated alias for the "Currency_Symbol" value of the General_Category value, but it is also the abbreviated alias for the Script property. However, the aliases for values for any single property are always unique within the context of that property. What that means is that expressions that combine a property alias and a property value alias, such as "lb=BA" or "gc=Sc" *always*
refer unambiguously just to one value of one given property, and will not match any other value of any other property.

The property value alias entries for three properties, Age, Block, and Joining_Group, make use of a special metavalue "n/a" in the field for the abbreviated alias. This should be understood as meaning that no abbreviated alias is defined for that value for that property, rather than as an alias per se.

In a few cases, because of longstanding legacy practice in referring to values of a property by short identifiers, the abbreviated alias and the long alias are the same. This can be seen, for example, in some property value aliases for the Line_Break property and the Grapheme_Cluster_Break property.

5.7 Matching Rules

When matching Unicode character property names and values, it is strongly recommended that all Property and Property Value Aliases be recognized. For best results in matching, rather than using exact binary comparisons, the following loose matching rules should be observed.

Numeric Property Values

For all numeric properties, and for properties such as Unicode_Radical_Stroke which are constructed from combinations of numeric values, use loose matching rule UAX44-LM1 when comparing property values.

UAX44-LM1. Apply numeric equivalences.

- "01.00" is equivalent to "1".
- "1.666667" in the UCD is a repeating fraction, and equivalent to "10/6" or "5/3".

Character Names

Unicode character names constitute a special case. Formally, they are values of the Name property. While each Unicode character name for an assigned character is guaranteed to be unique, names are assigned in such a way that the presence or absence of spaces cannot be used to distinguish them. Furthermore, implementations sometimes create identifiers from Unicode character names by inserting underscores for spaces. For best results in comparing Unicode character names, use loose matching rule UAX44-LM2.

UAX44-LM2.
Ignore case, whitespace, underscore ('_'), and all medial hyphens except the hyphen in U+1180 HANGUL JUNGSEONG O-E.

- "zero-width space" is equivalent to "ZERO WIDTH SPACE" or "zerowidthspace"
- "character -a" is *not* equivalent to "character a"

Symbolic Values

Property aliases and property value aliases are symbolic values. When comparing them, use loose matching rule UAX44-LM3.

UAX44-LM3. Ignore case, whitespace, underscore ('_'), and hyphens.

- "linebreak" is equivalent to "Line_Break" or "Line-break"
- "lb=BA" is equivalent to "lb=ba" or "LB=BA"

Loose matching is generally appropriate for the property values of Catalog, Enumeration, and Binary properties, which have symbolic aliases defined for their values. Loose matching should not be done for the property values of String properties, which do not have symbolic aliases defined for their values; exact matching for String property values is important, as case distinctions or other distinctions in those values may be significant.

5.8 Invariants

Property values in the UCD may be subject to correction in subsequent versions of the standard, as errors are found. Also, some multi-valued properties such as Line_Break or

Word_Break may have additional values defined for them. However, some property values and some aspects of the file formats are considered invariant. This section documents such invariants.

5.8.1 Character Property Invariants

All formally guaranteed invariants for properties or property values are described in the Unicode Character Encoding Stability Policy [Stability]. That policy and the list of invariants it enumerates are maintained outside the context of the Unicode Standard per se. They are not part of the standard, but rather are constraints on what can and cannot change in the standard between versions, and on what decisions the Unicode Technical Committee can and cannot take regarding the standard.

In addition to the formally guaranteed invariants described in the Unicode Character Encoding Stability Policy, this section notes a few additional points regarding character property invariants in the UCD.

Some character properties are simply considered *immutable*: once assigned, they are never changed. For example, a character's name is immutable, because of its importance in exact identification of the character. The Canonical_Combining_Class and Decomposition_Mapping of a character are immutable, because of their important to the stability of the Unicode Normalization Algorithm [UAX15].

The list of immutable character properties is shown in the table below:

Immutable Properties

Property Name	Abbr Name
Name	na
Jamo_Short_Name	jsn
Canonical_Combining_Class	ccc
Decomposition_Mapping	dm
Pattern_Syntax	Pat_Syn
Pattern_White_Space	Pat_WS

In some cases, a property is not immutable, but the list of possible values that it can have is considered invariant. For example, while at least some General_Category values are subject to change and correction, the enumerated set of possible values that the General_Category property can have is fixed and cannot be added to in the future.

All characters other than those of General_Category M* are guaranteed to have Canonical_Combining_Class=0. Currently it is also true that all characters other than those of General_Category Mn have Canonical_Combining_Class=0. However, the more constrained statement is not a guaranteed invariant; it is possible that some new character of General_Category Me or Mc could be given a non-zero value for Canonical_Combining_Class in the future.

In Unicode 4.0 and thereafter, the General_Category value *Decimal_Number* (Nd), and the Numeric_Type value *Decimal*
(de) are defined to be co-extensive; that is, the set of characters having General_Category=Nd will always be the same as the set of characters having NumericType=de.

5.8.2 UCD File Format Invariants

There are also some constraints on allowable change in the file formats for UCD files. In general, the file format conventions are changed as little as possible, to minimize the impact on implementations which parse the machine-readable data files. However, some of the constraints on allowable file format change go beyond conservatism in format and instead have the status of invariants. These guarantees apply in particular to UnicodeData.txt, the very first data file associated with the UCD.

The number and order of the fields in UnicodeData.txt is fixed. Any additional information about character properties to be added to the UCD in the future will appear in separate data files, rather than being added as an additional field to UnicodeData.txt or by reinterpretation of any of the existing fields.

5.8.3 Invariants in Implementations

Applications may wish to take the various character property and file format invariants into account when choosing how to implement character properties.

The Canonical_Combining_Class offers a good example. The character property invariants regarding Canonical_Combining_Class guarantee that values, once assigned, will never change, and that all values used will be in the range 0..255. This means that the Canonical_Combining_Class can be safely implemented in an unsigned byte and that any value stored in a table for an existing character will not need to be updated dynamically for a later version.

In practice, for Canonical_Combining_Class far fewer than 256 values are used. Unicode 3.0 used 53 values; Unicode 3.1 through Unicode 4.1 used 54 values; and Unicode 5.0 through Unicode 5.1 used 55 values. New, non-zero Canonical_Combining_Class values are seldom added to the standard. (For details about this history, see DerivedCombiningClass.txt.) Implementations may take advantage of this fact for compression, because only the ordering of the non-zero values, and not their absolute values, matters for the Canonical Ordering Algorithm. In principle, it would be possible for up to 256 values to be used in the future, but the chances of the actual number of values exceeding 128 are remote at this point. There are implementation advantages in restricting the number of internal class values to 128—for example, the ability to use signed bytes without implicit widening to ints in Java.

5.9 Validation

This section still needs more work. The table should be restructured. Two possibilities are to have a complete list of all the properties, each with an explicit syntax specified for them (along the lines now done in UAX #38 for the Unihan properties), or to refactor the discussion as follows: First provide a section classifying the value domains for properties into pattern types, providing a table which lists all the properties associated with each pattern type, and then give another table that shows the regex used for each pattern type. That would be much easier both to understand and and to validate.

The table below appears to have errors and omissions in it still. Age is underspecified, and should be done more tightly. The expression for Unicode_1_Name is probably too tight, by contrast. The ISO_Comment field is incorrect. And the regex for the Block and the Script properties should not be the same. There may be other problems, as well.

The property values for many of the Unicode character properties have a regular syntax that makes it possible to validate the values in the UCD data files by means of regular expressions. Regular expressions for a number of the Catalog, String and Miscellaneous type properties in the UCD are provided in the table below. These expressions use Perl syntax, but may be of course be converted to other formal conventions for use with other regular expression engines.

Regular Expressions for Property Values

Abbr	Name	Regex for Allowable Values	
age	Age	/([0-9]+\.[0-9]|unassigned)/	
nv	Numeric_Value	/-?[0-9]+\.[0-9]+/	Field 2
		/-?[0-9]+(\[0-9]+)?/	Field 3
blk	Block	/[a-zA-Z0-9]+([_\ ][a-zA-Z0-9]+)*/	
sc	Script		
dm	Decomposition_Mapping	/[\x{0}-\x{10FFFF}]+/	
FC_NFKC	FC_NFKC_Closure		
cf	Case_Folding	/[\x{0}-\x{10FFFF}]+/	
lc	Lowercase_Mapping		
tc	Titlecase_Mapping		
uc	Uppercase_Mapping		
sfc	Simple_Case_Folding	/[\x{0}-\x{10FFFF}]/	
slc	Simple_Lowercase_Mapping		
stc	Simple_Titlecase_Mapping		
suc	Simple_Uppercase_Mapping		
bmg	Bidi_Mirroring_Glyph	/[\x{0}-\x{10FFFF}]?/	
isc	ISO_Comment	/([A-Z0-9]+(([-\ ]|\ -|-\ )[A-Z0-9]+)*|\<CONTROL\>)?/	
na1	Unicode_1_Name	/([A-Z0-9]+(([-\ ]|\ -|-\ )[A-Z0-9]+)*(\ \((CR|FF|LF|NEL)\))?)?/	
na	Name	/([A-Z0-9]+(([-\ ]|\ -|-\ )[A-Z0-9]+)*|\<CONTROL\>)?/	

5.10 Deprecation

In the Unicode Standard, the term *deprecation* is used somewhat differently than it is in some other standards. Deprecation is used to mean that a character or other feature is strongly discouraged from use. This should not, however, be taken as indicating that anything has been removed from the standard, nor that anything is *planned* for removal from the standard. Any such change is constrained by the Unicode Consortium Stability Policies [Stability].

For the Unicode Character Database, there are two important types of deprecation to be

noted. First, an *encoded character* may be deprecated. Second, a *character property* may be deprecated.

When an encoded character is strongly discouraged from use, it is given the property value Deprecated=True. The Deprecated property is a binary property defined specifically to carry this information about Unicode characters. Note that very few characters are ever formally deprecated this way; it is not enough that a character be uncommon, obsolete, disliked, or not preferred. Only those few characters which have been determined by the UTC to have serious architectural defects or which have been determined to cause significant implementation problems are ever deprecated. Note that even in the most severe cases, such as the deprecated format control characters (U+206A..U+206F), an encoded character is *never* removed from the standard. Furthermore, although deprecated characters are strongly discouraged from use, and should be avoided in favor of other, more appropriate mechanisms, they *may* occur in data. Conformant implementations of Unicode processes such a Unicode normalization *must* handle even deprecated characters correctly.

In the Unicode Character Database, a character property itself may also become strongly discouraged—usually because it no longer serves the purpose it was originally defined for. In such cases, the property is labelled "deprecated" in the Property Table. For example, see the Grapheme_Link property.

6 Test Files

The UCD contains a number of test data files. Those provide data in standard formats which can be used to test implementations of Unicode algorithms. The test data files distributed with this version of the UCD are listed in the table below.

Unicode Algorithm Test Data Files

File Name	Specification	Status	Unicode Algorithm
NormalizationTest.txt	[UAX15]	N	Unicode Normalization Algorithm
LineBreakTest.txt	[UAX14]	N	Unicode Line Breaking Algorithm
GraphemeBreakTest.txt	[UAX29]	N	Grapheme Cluster Boundary Determination
WordBreakTest.txt	[UAX29]	N	Word Boundary Determination
SentenceBreakTest.txt	[UAX29]	N	Sentence Boundary Determination

The normative status of these test files reflects their use to determine the correctness of implementations claiming conformance to the respective algorithms listed in the table. There is no requirement that any particular Unicode implementation also implement the Unicode Line Breaking Algorithm, for example, but *if* it implements that algorithm correctly, it should be able to replicate the test case results specified in the data entries in LineBreakTest.txt.

6.1 NormalizationTest.txt

This file contains data which can be used to test an implementation of the Unicode Normalization Algorithm. (See [UAX15].)

The data file has a Unicode string in the first field (which may consist of just a single code point). The next four fields then specify the expected output results of converting that

string to Unicode Normalization Forms NFC, NFD, NFKC, and NFKD, respectively. There are many tricky edge cases included in the input data, to ensure that implementations have correctly implemented some of the more complex subtleties of the Unicode Normalization Algorithm.

The header section of NormalizationTest.txt provides additional information regarding the normalization invariant relations that any conformant implementation should be able to replicate.

The Unicode Normalization Algorithm is not tailorable. Conformant implementations should be expected to produce results as specified in NormalizationTest.txt and should not deviate from those results.

6.2 Segmentation Test Files and Documentation

LineBreakTest.txt, located in the auxiliary directory of the UCD, contains data which can be used to test an implementation of the Unicode Line Breaking Algorithm. (See [UAX14].) The header of that file specifies the data format and the use of the test data to specify line break opportunities. Note that non-ASCII characters are used in this test data as field delimiters.

There is an associated documentation file, LineBreakTest.html, which displays the results of the Line Breaking Algorithm in an interactive chart form, with a documented listing of the rules.

The Unicode text segmentation test data files are also located in the auxiliary directory of the UCD. They contain data which can be used to test an implementation of the segmentation algorithms specified in [UAX29]. The headers of those file specify the data format and the use of the test data to specify text segmentation opportunities. Note that non-ASCII characters are used in this test data as field delimiters.

There are also associated documentation files, which display the results of the segmentation algorithms in an interactive chart form, with a documented listing of the rules:

- GraphemeBreakTest.html
- SentenceBreakTest.html
- WordBreakTest.html

Unlike the Unicode Normalization Algorithm, the Unicode Line Breaking Algorithm and the various text segmentation algorithms *are* tailorable, and there is every expectation that implementations will tailor these algorithms to produce results as needed. The test data files only test the *default*
behavior of the algorithms. Testing of tailored implementations will need to modify and/or extend the test cases as appropriate to match any documented tailoring.

7 UCD Change History

This section summarizes the changes to the UCD—including its documentation files—and is organized by Unicode versions. The summary includes changes extending all the way back to Unicode 2.0.0, taken from the obsoleted UCD.html documentation file, which predates the creation of this annex. The intent is for this first consolidated version of the annex to preserve that complete prior history from UCD.html. Subsequent versions

of the annex will provide only an abbreviated UCD change history section containing only the delta change information from each preceding version.

Starting from Unicode 4.0.1, references in the change history are often made to a Public Review Issue (PRI). See http://www.unicode.org/review/resolved-pri.html for more information about each of those cases.

Changes documented prior to Unicode 4.0 only covered UnicodeData.txt. From Unicode 4.0 onward, the documentation of changes includes modifications of other files as well.

Unicode 5.2.0

General:

TBD

Common file changes:

TBD

Changes in specific files:

TBD

Unicode 5.1.0

General:

- Added UCD in XML to the release, in a new subdirectory "ucdxml".

UCD.html:

- Added clarification regarding the Decomposition_Mapping for Hangul syllables.
- Added specific documentation about First/Last convention for ranges in UnicodeData.txt.
- Improved introduction to General Category Values.
- Added reference to UTR #23 and updated other references.
- Added note regarding abbreviation of Quick Check property names.
- Added notes regarding omissions of foldings where the value is the same as the code point itself.
- Applied correction for erratum about derivation of Default_Ignorable_Code_Point.
- Added the section on Validation of property values, with string property validation, default values, and boolean values.
- Removed Special_Case_Condition. (The property values were never defined clearly enough to be applied.)
- Corrected typos for PropList and Composition_Exclusion.
- Updated property type for Jamo_Short_Name to Miscellaneous (M).
- Added clarification of property type for Canonical_Combining_Class.
- Updated listing of default values for UnicodeData fields.
- Moved documentation of Grapheme_Link from PropList.txt to DerivedCoreProperties.txt section.

- Updated references to Unihan.html, to refer to UAX #38, instead. Removed invalid bookmarks on Unihan property tags.

Changes in specific files:

Appropriate data files were updated to include the 1,624 new characters added in Unicode 5.1.

- UnicodeData.txt
  - The 5 Arabic characters that surround numeral sequences (U+0600..U+0603, U+06DD) were changed from Bidirectional_Class=AL to AN. This has the effect of putting the surrounding sign and the numeral sequence in the same directional run, making them easier to implement correctly.
  - 11 directional quotation marks (U+2018..U+201F, U+301D..U+301F) were changed to Bidi_Mirrored=N. This constituted a partial reversion of the change for Version 5.0 related to PRI #91.
  - U+05BE was changed from gc=Po to gc=Pd.
  - U+02EC and U+0374 were changed from gc=Sk to gc=Lm.
  - U+A802 was changed from gc=Mc to gc=Mn.
  - 10 compatibility ideographs were given numeric values.
- Unihan.txt
  - Two existing unified ideographs, U+6F06 and U+9621, were given numeric values.
  - One new provisional property was added. Corrections and additions to other properties were made. See [UAX38] for the modification history.
- ArabicShaping.txt
  - A new joining group, BURUSHASKI YEH BARREE, was added.
- BidiMirroring.txt
  - Removed glyph mappings for the 11 characters that were changed to Bidi_Mirrored=N.
  - Updated glyph mappings for U+2278 and U+2279 to [BEST FIT].
- Blocks.txt
  - Added 17 new block definitions.
- DerivedNumericValues.txt
  - A third field was added to this file, expressing the extracted numeric value as a whole integer, if possible, or as a rational fraction, for example, 1/6.
- LineBreak.txt
  - There were numerous updates to linebreaking properties. See the Modification History in UAX #14 for details. Also see PRI #105.
- NamedSequences.txt
  - Lithuanian named sequences were approved and moved to this file from NamedSequencesProv.txt.
- NamedSequencesProv.txt
  - A new, complete set of named sequences for Tamil consonants and syllables were added to this file.
- PropertyAliases.txt
  - Added entry for Jamo_Short_Name.

    - Added corrected alias for Simple_Case_Folding.
    - Removed entry for Special_Case_Condition.
- PropertyValueAliases.txt
    - Appropriate aliases were added for new Block and Script values.
    - For Block aliases, new values "ASCII", "Latin_1", and "Greek" were added for common use.
    - Appropriate aliases were added for new Word_Break and Sentence_Break values.
    - Explicit Y/N, T/F aliases were added for all binary properties.
    - Additional aliases using underscores were added for aliases that used hyphen-minus.
    - Some titlecased aliases were added for consistency.
- PropList.txt
    - The middle dots (U+00B7, U+0387) were added to identifiers by changing them to Other_ID_Continue=Y. See PRI #100.
    - For consistency, the halfwidth Katakana sound marks (U+FF9E, U+FF9F) were added to Grapheme_Extend by making them Other_Grapheme_Extend=Y.
    - The tag characters (U+E0001, U+E0020..U+E007F) were changed to Deprecated=Y.
    - Other_Math values were adjusted for a number of mathematical symbols.
    - U+05BE was changed to Dash=Y, consistent with the change in its General Category.
- Scripts.txt
    - 11 new Script values were added: Sundanese, Lepcha, Ol_Chiki, Vai, Saurashtra, Kayah_Li, Rejang, Lycian, Carian, Lydian, and Cham.
    - U+0374 and U+0385 were changed from Greek to Common, because of canonical equivalence issues.
    - U+0CF1 and U+0CF2 were changed from Kannada to Common, because of their use in Vedic texts.
    - Roman numeral compatibility characters, U+2160..U+2183, were changed from Common to Latin.
    - A circled Hangul character, U+327E, was changed from Common to Hangul.
    - Squared Katakana compatibility characters, U+32D0..U+32FE and U+3300..U+3357, were changed from Common to Katakana.
- SpecialCasing.txt
    - Clarified the use of language tags for specification of casing contexts.
- StandardizedVariants.txt
    - Updated documentation to note the existence of ideographic variation sequences and the Ideographic Variation Database (IVD).
- GraphemeBreakProperty.txt
    - Added Prepend class (for Logical_Order_Exception=Y).
    - Added SpacingMark class (for most gc=Mc).
- SentenceBreakProperty.txt
    - Added Extend and SContinue classes.
    - Split U+0009 and U+000A off from Sep class into CR and LF classes.

    - Removed U+00A0 from OLetter class.
- WordBreakProperty.txt
    - Added CR, LF, Newline, Extend, and MidNumLet classes.
    - Moved U+0027 and U+2019 from MidLetter class to MidNumLet class.
    - Moved U+002E from MidNum class to MidNumLet class.
    - Added U+060C and U+066C to MidNum class.
- Text Boundary Test Files
    - The existing test files, GraphemeBreakTest.txt, SentenceBreakTest.txt, and WordBreakTest.txt were substantially extended.
    - A new test file, LineBreakTest.txt, was added, with test cases for UAX #14.

Unicode 5.0.0

UCD.html:

- Added new properties.
- Updated property invariants for combining classes.
- Reorganized order of sections in the document for clarity.

Common file changes:

In many data files an explicit default property assignment range was added (in a machine-readable comment line), to assist implementations in assigning values for code points not otherwise listed in the data file.

Changes in specific files:

Appropriate data files were updated to include the 1,369 new characters added in Unicode 5.0.

Two new data files, NameAliases.txt and NamedSequencesProv.txt, were added to the UCD.

- UnicodeData.txt

  Note that except for the changes involving U+0294 LATIN LETTER GLOTTAL STOP, changes made to General_Category and Bidirectional_Class impacted primarily a handful of archaic letters.

    - U+10341 GOTHIC LETTER NINETY was changed from gc=Lo to gc=Nl. This change also impacted a numeric field, for consistency.
    - U+103D0 OLD PERSIAN WORD DIVIDER was changed from gc=So to gc=Po, and from bc=ON to bc=L.
    - U+103D1..U+103D5 were changed from bc=ON to bc=L.
    - U+23B4..U+23B6 were changed from various punctuation assignments to gc=So.
    - U+2132 TURNED CAPITAL F was changed from gc=So to gc=Lu, and from bc=ON to bc=L.
    - U+2183 ROMAN NUMERAL REVERSED ONE HUNDRED was changed from gc=Nl to gc=Lu.

    - U+0294 LATIN LETTER GLOTTAL STOP was changed from gc=Ll to gc=Lo.
    - Casing assignments were added for several characters for new case pairs.
    - Case mappings were removed for U+0294 LATIN LETTER GLOTTAL STOP and updated for U+0241 LATIN CAPITAL LETTER GLOTTAL STOP.
    - 30 characters were changed to Bidi_Mirrored=Y. These consisted of compatibility paired punctuation and some quotation marks. See PRI #80 and PRI #91.
- Unihan.txt
    - 4 new provisional properties were added, and extensive corrections and additions to other properties were made. See Unihan.html for the modification history.
- ArabicShaping.txt
    - New joining classes were added for N'Ko.
- BidiMirroring.txt
    - 30 entries were added, to give glyph mappings for characters changed to Bidi_Mirrored=Y. See PRI #80 and PRI #91.
- Blocks.txt
    - Added 9 new block definitions.
- DerivedCoreProperties.txt
    - The deprecated derived property, Grapheme_Link, was added to this file.
- LineBreak.txt
    - There were numerous updates to linebreaking properties. See the Modification History in UAX #14 for details. Also see PRI #88.
- NamedSequences.txt
    - 6 named sequences for Gurmukhi and one for Latin were removed.
- PropertyValueAliases.txt
    - Appropriate aliases were added for new Block and Script values.
- PropList.txt
    - The Grapheme_Link property was deprecated and moved to DerivedCoreProperties.txt as derivable. U+034F COMBINING GRAPHEME JOINER was removed from the derivation.
    - U+1D6A4 MATHEMATICAL ITALIC SMALL DOTLESS I and U+1D6A5 MATHEMATICAL ITALIC SMALL DOTLESS J were added to Other_Math.
    - U+1039F UGARITIC WORD DIVIDER and U+103D0 OLD PERSIAN WORD DIVIDER were added to Terminal_Punctuation.
- Scripts.txt
    - 5 new Script values were added: Balinese, Cuneiform, Phoenician, Phags-pa, and Nko.
    - A new Script value Unknown was added and made the default for unassigned characters. See PRI #87.
    - 3 Mongolian punctuation characters used by Phags-pa were changed to Script=Common.
    - U+1DBF MODIFIER LETTER SMALL THETA was changed from Script=Latin to Script=Greek.
    - U+2132 TURNED CAPITAL F was changed from Script=Common to Script=Latin.

- StandardizedVariants.txt
  - 6 standardized variation sequences were added for Phags-pa.
- WordBreakProperty.txt
  - U+2132 TURNED CAPITAL F was added to ALetter.
  - 220 characters from the Myanmar, Khmer, Tai Le, and New Tai Lue scripts were removed from ALetter, because those scripts do not cusomarily use spaces between words and require special handling.

Unicode 4.1.0

General:

- Added a new subdirectory "auxiliary". In Version 4.1.0 it contains data files for properties associated with UAX #29: Text Boundaries [UAX29].

UCD.html:

- Added description of new directory and release structure, including files in the auxiliary subdirectory.
- Removed exception for field numbering in LineBreak.txt and EastAsianWidth.txt.
- Added new properties, and changed some of the documentation of the identifier properties.
- Removed the material that is now to be in Unihan.html.
- Removed the listing of default Bidi_Class values, referring now to DerivedBidiClass.txt.
- Replaced direct links to UAXes with links to references section.

Common file changes:

All remaining files not corrected for Unicode 4.0.1 have had their headers updated to explicitly point to Terms of Use. The headers have also been synchronized somewhat to share a more common format for file version, date, and pointers to documentation. The major exception is UnicodeData.txt, which for legacy reasons, has no header.

Changes in specific files:

Appropriate data files were updated to include the 1,273 new characters added in Unicode 4.1.0.

The description of the Unihan properties was separated out from UCD.html, extensively revised, and moved into a new documentation file, Unihan.html.

- UnicodeData.txt
  - The Bidi_Class value of U+202F was changed from bc=WS to bc=CS. See PRI #45.
  - The Bidi_Class value of U+FF0F was changed from bc=ES to bc=CS. See PRI #44.
  - The Bidi_Class value of U+2212 MINUS SIGN and 9 other characters similar to either a minus sign or a plus sign were changed to bc=ES. See PRI #57.
  - U+30FB KATAKANA MIDDLE DOT and U+FF65 HALFWIDTH KATAKANA MIDDLE DOT were changed from gc=Pc to gc=Po. See PRI #55.

    - Case mappings were added for Georgian capitals (Asomtavruli) to map them to the newly added Nuskhuri alphabet.
    - U+A015 YI SYLLABLE WU was changed from gc=Lo to gc=Lm.
    - 9 Ethiopic digits were changed from gc=Nd to gc=No.
    - The Numeric_Type of U+1034A GOTHIC LETTER NINE HUNDRED was changed from nt=None to nt=Nu, and it was given a Numeric_Value of 900.
    - Uppercase and titlecase mappings were added for U+019A LATIN SMALL LETTER L WITH BAR and U+0294 LATIN LETTER GLOTTAL STOP to map them to newly added capital letters.
- Unihan.txt
    - Extensive additions and corrections were made for this data file. See Unihan.html for the modification history.
- ArabicShaping.txt
    - The Joining_Group of U+06C2 ARABIC LETTER HEH GOAL WITH HAMZA ABOVE was changed to jg=Heh_Goal.
- BidiMirroring.txt
    - The Bidi_Mirroring_Glyph value for U+2A2D was corrected.
- Blocks.txt
    - Added 20 new block definitions.
- LineBreak.txt
    - The Line_Break property of all conjoining jamos was updated from lb=ID to make use of Hangul-specific Line_Break property values, aligned with the Hangul_Syllable_Type property.
    - Many other corrections were made to the Line_Break property of characters, particularly for punctuation marks specific to Runic, Mongolian, Tibetan and various Indic scripts. See the Modification History in UAX #14 for details.
- PropertyAliases.txt
    - Properties and aliases were added for UAX #29, Text Boundaries: Grapheme_Cluster_Break, Word_Break, and Sentence_Break.
    - Properties and aliases were added for: Other_ID_Continue, Pattern_White_Space, and Pattern_Syntax.
    - An alias was added for White_Space: "space", for compatibility with POSIX.
- PropertyValueAliases.txt
    - Property value aliases were added for all new properties, and for new values added to existing catalog properties (blocks and scripts).
    - Property value aliases were added for compatibility with POSIX: "cntrl", "digit", and "punct".
- PropList.txt
    - 3 new properties were added: Other_ID_Continue, Pattern_White_Space, and Pattern_Syntax.
    - U+30A0 KATAKANA-HIRAGANA DOUBLE HYPHEN was given the Dash property.
    - U+A015 YI SYLLABLE WU was given the Extender property.
    - Golden number runes (U+16EE..U+16F0), Roman numerals (U+2160..U+2183), and U+1034A GOTHIC LETTER NINE HUNDRED were removed from Other_Alphabetic.
    - Circled Latin letters (U+24B6..U+24E9) were added to Other_Alphabetic.

These changes to Other_Alphabetic were to better align Alphabetic and casing properties. The derived property Alphabetic is now a superset of the derived properties Lowercase and Uppercase, for compatibility with POSIX-style character classes.

- 3 musical symbol combining flags (U+1D170..U+1D172) were added to Other_Grapheme_Extend to fix an inconsistency in the data.
- U+200B ZERO WIDTH SPACE was removed from Other_Default_Ignorable_Code_Point.

- Scripts.txt
  - 8 new Script values were added: Buginese, Coptic, New_Tai_Lue, Glagolitic, Tifinagh, Syloti_Nagri, Old_Persian, and Kharoshthi.
  - The Script value Katakana_Or_Hiragana (Hrkt) was removed.
  - The Script for the 14 Coptic letters in the Greek and Coptic block were updated to sc=Copt.
  - 10 characters (punctuation and extenders) shared by Katakana and Hiragana were changed from sc=Hrkt to sc=Zyyy.
- SpecialCasing.txt
  - The case mapping contexts defined in this file were updated.
  - A number of clarifying changes were made to comments in the header of this data file.

Unicode 4.0.1

UCD.html:

- Added documentation for two new properties.
- Added the property types Catalog and Miscellaneous.
- Described loose matching of property names and values.
- Added to documentation of file format.

Common file changes:

Some property values have different casing (upper versus lower) for consistency between the data files and the PropertyValueAlias file. There are some additional changes in comments:

- Nearly all files changed headers to explicitly point to *Terms of Use*.
- Labels for code points without names now have a more uniform style, such as <*reserved-1234*>.
- Where characters with a default value are not listed, that information is indicated in the total code point counts.
- The full property name and property value name (for enumerated properties) is usually supplied in a comment.

Changes in specific files:

- UnicodeData.txt
  - Changed the General_Category value of Zero Width Space (U+200B) from Zs to Cf. For background information, see PRI #21.
  - Bidi Conformance was made much clearer and more rigorous, also resulting

in a number of property changes. In particular, the Bidi_Class changes impact number and date formatting with the following characters: +, -, /
    - A review of Bidi_Class=BN and Default_Ignorable_Code_Point characters resulted in a number of changes; for details, see PRI #28.
    - Some other Bidi_Class tweaks were made for consistency.
    - Braille symbols were changed to being strong Left-to-right, to reflect usage.
    - The Bidi_Class and other property values of the Join_Control characters were not changed, but their role in combining characters sequences was. For more information, see http://www.unicode.org/versions/Unicode4.0.1/.
    - Removed an extraneous space at the end of the name field for two characters.
- Unihan.txt
    - There was a major revision of the Unihan data file, to bring it up-to-date for Unicode 4.0. (It was not released in Version 4.0.0, because of the time required to complete and check corrections to the data file.) This update rolls in fixes for nearly all known errors in the prior version of the file and adds a very large amount of other informative data. For details, see the header of that file.
    - Added three new tags: kHanyuPinlu, kGSR, and kIRG_USource.
    - Completed data for kCihaiT, kCowles, kGradeLevel, and kLau.
    - The kMandarin field has been corrected and its order restored to a "frequency" order.
- ArabicShaping.txt
    - Moved one entry into code point order.
- Blocks.txt
    - Corrected name of the Cyrillic Supplement block.
- DerivedCoreProperties.txt
    - ZWNJ/ZWJ (U+200C..U+200D) now have the Grapheme_Extend property.
- DerivedNormalizationProps.txt
    - The particular values associated with the Quick Check properties for characters were not changed, but a revision was made in how the Quick Check properties are expressed in the file, to bring it more into line with the model for other properties. This resulted in a significant change in the format of the data file and the explicit separation of Yes, No, and Maybe values. In addition, the actual aliases for the property values changed in the data file.
- Index.txt
    - Updated to correspond to the character index published as part of the Unicode Standard, Version 4.0.
- LineBreak.txt
    - Many changes for consistency and to better match best practice in existing line break implementations. See the Modification History in UAX #14 for details.
- PropertyAliases.txt
    - Addition of some property categories, with the order of property aliases adjusted for clarity.
    - Addition of alias entries for the new STerm and Variation_Selector properties.
- PropertyValueAliases.txt

    - Addition of specific values and aliases for age.
    - Addition of second alias for the Cyrillic Supplement block.
    - Addition of second alias for the Inseparable value of the Line Break property.
    - Revision of the all the Normalization Quick Check properties, to replace the pseudo-property "qc" with actual specific properties with explicit enumerated value aliases.
    - Addition of Katakana_Or_Hiragana script alias.
    - Fixed None, so it is used uniformly in first aliases instead of being the only n/a.
- PropList.txt
    - Major revision of the Other_Math property to align the derived Math property with the explanation given in UTR #25.
    - Extension of the list of characters with the Soft_Dotted property.
    - Significant update of the list of characters with the Terminal_Punctuation property.
    - Addition of a new STerm property, to simplify the description used in UAX #29.
    - Addition of the Variation_Selector property.
    - Reassignment of the list of characters with the Other_Default_Ignorable_Code_Point property, to enable simpler derivation.
    - Addition of ZWNJ/ZWJ (U+200C..U+200D) to Other_Grapheme_Extend.
- Scripts.txt
    - Significant revision of script assignments, to assign specific script values to many characters that previously had the Common script value.
    - Addition of the Katakana_Or_Hiragana script value, with list of characters for it.
    - The Script=Common values are now listed explicitly.
- SpecialCasing.txt
    - Correction of typo in comments.

Unicode 4.0.0

General:

For details on changes made to the UCD for Unicode 4.0.0, see Section D.4, "Changes from Unicode Version 3.2 to Version 4.0" in Appendix D of *The Unicode Standard, Version 4.0.*

- The Hyphen property is now Stabilized.
- Two Khmer characters were deprecated and four others were strongly discouraged.

Common file changes:

Default property values were more precisely defined, for code points not explicitly listed in the data files.

Changes in specific files:

- UnicodeData.txt

    - Numeric_Type=Decimal was aligned with General_Category=Nd.
    - The General_Category value of the modifier letters U+02B9..U+02BA, U+02C6..U+02CF was changed to Lm.
- Unihan.txt
    - CJK numeric values were added.
    - A new Unicode_Radical_Stroke property was defined.
- ArabicShaping.txt
    - U+06DD ARABIC END OF AYAH was changed to Join_Type=Non_Joining.
- Blocks.txt
    - Added new block definitions.
- DerivedCoreProperties.txt
    - Modifed the derivation of Grapheme_Extend to remove halfwidth katakana marks and most gc=Mc (except as needed to preserve canonical equivalences).
- HangulSyllableType.txt
    - A new data file, defining the Hangul_Syllable_Type property.
- LineBreak.txt
    - Modified to add lb=NL and lb=WJ values.
- PropList.txt
    - Other_Default_Ignorable_Code_Point was extended to include Hangul filler characters, soft hyphen (U+00AD), the combining grapheme joiner (U+034F), and zero width space (U+200B).
    - U+06DD ARABIC END OF AYAH and U+070F SYRIAC ABBREVIATION MARK were removed from Other_Default_Ignorable_Code_Point.
- PropertyValueAliases.txt
    - Added property value aliases for new blocks and scripts.
    - Added property value aliases for other new properties.
    - Added property value aliases NL and WJ for Line_Break.
- Scripts.txt
    - Added Script property values for newly encoded scripts.
    - Added a Script property value for Braille.
- SpecialCasing.txt
    - Fixes were made for Turkish and Lithuanian.

Unicode 3.2.0

General:

For details on changes made to the UCD for Unicode 3.2.0, see Section D.3, "Changes from Unicode Version 3.1 to Version 3.2" in Appendix D of *The Unicode Standard, Version 4.0.*

- Added a new subdirectory "extracted". It contains the data files for Derived Extracted Properties.

Changes in specific files:

Appropriate data files were updated to include the 1,016 new characters added in

Unicode 3.2.0.

- UnicodeData.txt
  - Updated ISO 6429 names for control functions to match the currently published version of that standard.
  - Changed the General_Category value for Mongolian free variation selectors (U+180B..U+180D) from Cf to Mn.
  - Changed the General_Category value for U+0B83 TAMIL SIGN VISARGA (aytham) from Mc to Lo.
  - Changed the General_Category value for U+06DD ARABIC END OF AYAH from Me to Cf.
  - Changed the General_Category value for U+17D7 KHMER SIGN LEK TOO from Po to Lm.
  - Changed the General_Category value for U+17DC KHMER SIGN AVAKRAHASANYA from Po to Lo.
  - Changed canonical decomposition for U+F951 from 96FB to 964B (see *Corrigendum #3: U+F951 Normalization*).
- PropertyAliases.txt
  - A new data file, defining aliases for properties.
- PropertyValueAliases.txt
  - A new data file, defining aliases for property values.

Unicode 3.1.1

Changes in specific files:

- UnicodeData.txt
  - Modification of the ISO 10646 annotation (in the ISO_Comment field) regarding Greek tonos, affecting entries for U+0301 and U+030D.

Unicode 3.1.0

General:

For details on changes made to the UCD for Unicode 3.1.0, see Section D.2, "Changes from Unicode Version 3.0 to Version 3.1" in Appendix D of *The Unicode Standard, Version 4.0.*

Changes in specific files:

Appropriate data files were updated to include the 2,237 new entries, to cover new individual characters and the new ranges of Unified CJK Ideographs encoded in Unicode 3.1.0.

- UnicodeData.txt
  - Changed the General_Category value of U+16EE..U+16F0 (Runic golden numbers) from No to Nl.

Unicode 3.0.1

General:

- Added 5- and 6-digit representation of code points past U+FFFF.

Changes in specific files:

- UnicodeData.txt
  - Added Private Use range definitions for Planes 15 and 16.
  - Minor additions for the 10646 annotations (ISO_Comment field).

Unicode 3.0.0

Modifications made for Version 3.0.0 of UnicodeData.txt include many new characters and a number of property changes. These are summarized in Appendix D of *The Unicode Standard, Version 3.0.*

Unicode 2.1.9

Modifications made for Version 2.1.9 of UnicodeData.txt include:

- Corrected combining class for U+05AE HEBREW ACCENT ZINOR.
- Corrected combining class for U+20E1 COMBINING LEFT RIGHT ARROW ABOVE.
- Corrected combining class for U+0F35 and U+0F37 to 220.
- Corrected combining class for U+0F71 to 129.
- Added a decomposition for U+0F0C TIBETAN MARK DELIMITER TSHEG BSTAR.
- Added decompositions for several Greek symbol letters: U+03D0..U+03D2, U+03D5, U+03D6, U+03F0..U+03F2.
- Removed decompositions from the conjoining jamo block: U+1100..U+11F8.
- Changes to decomposition mappings for some Tibetan vowels for consistency in normalization. (U+0F71, U+0F73, U+0F77, U+0F79, U+0F81).
- Updated the decomposition mappings for several Vietnamese characters with two diacritics (U+1EAC, U+1EAD, U+1EB6, U+1EB7, U+1EC6, U+1EC7, U+1ED8, U+1ED9), so that the recursive decomposition can be generated directly in canonically reordered form (not a normative change).
- Updated the decomposition mappings for several Arabic compatibility characters involving shadda (U+FC5E..U+FC62, U+FCF2..U+FCF4), and two Latin characters (U+1E1C, U+1E1D), so that the decompositions are generated directly in canonically reordered form (not a normative change).
- Changed Bidi_Class values for: U+00A0 NO-BREAK SPACE, U+2007 FIGURE SPACE, U+2028 LINE SEPARATOR.
- Changed Bidi_Class values for extenders of General_Category Lm: U+3005, U+3021..U+3035, U+FF9E, U+FF9F.
- Changed the General_Category and Bidi_Class values for the Greek numeral signs: U+0374, U+0375.
- Corrected the General_Category value for U+FFE8 HALFWIDTH FORMS LIGHT VERTICAL.
- Added Unicode 1.0 names for many Tibetan characters (informative).

Unicode 2.1.8

Modifications made for Version 2.1.8 of UnicodeData.txt include:

- Added combining class 240 for U+0345 COMBINING GREEK YPOGEGRAMMENI so that decompositions involving iota subscript are derivable directly in canonically reordered form; this also has a bearing on simplification of casing of polytonic Greek.
- Changes were made in decompositions related to Greek tonos. These result from the clarification that monotonic Greek "tonos" should be equated with U+0301 COMBINING ACUTE, rather than with U+030D COMBINING VERTICAL LINE ABOVE. (All Greek characters in the Greek block involving "tonos"; some Greek characters in the polytonic Greek in the 1FXX block.)
- Changed decompositions involving dialytika tonos. (U+0390, U+03B0)
- Changed ternary decompositions to binary. (U+0CCB, U+FB2C, U+FB2D) These changes simplify normalization.
- Removed canonical decomposition for the generic candrabindu (U+0310).
- Corrected error in canonical decomposition for U+1FF4.
- Added compatibility decompositions to clarify collation tables. (U+2100, U+2101, U+2105, U+2106, U+1E9A)
- A series of General_Category changes to assist the convergence of the Unicode definition of identifier with ISO TR 10176:
  - So > Lo: U+0950, U+0AD0, U+0F00, U+0F88..U+0F8B
  - Po > Lo: U+0E2F, U+0EAF, U+3006
  - Lm > Sk: U+309B, U+309C
  - Po > Pc: U+30FB, U+FF65
  - Ps/Pe > Mn: U+0F3E, U+0F3F
- A series of Bidi_Class changes for consistency:
  - L > ET: U+09F2, U+09F3
  - ON > L: U+3007
  - L > ON: U+0F3A..U+0F3D, U+037E, U+0387
- Add case mapping: U+01A6 ↔ U+0280
- Updated symmetric swapping value for guillemets: U+00AB, U+00BB, U+2039, U+203A.
- Changes to combining class values. Most Indic fixed position class nonspacing marks were changed to combining class 0. This fixes some inconsistencies in how canonical reordering would apply to Indic scripts, including Tibetan. Indic interacting top/bottom fixed position classes were merged into single (non-zero) classes as part of this change. Tibetan subjoined consonants are changed from combining class 6 to combining class 0. Thai pinthu (U+0E3A) moved to combining class 9. Moved two Devanagari stress marks into generic above and below combining classes (U+0951, U+0952).
- Corrected the placement of semicolon near the symmetric swapping field. This affected U+FA0E and other scattered positions to U+FA29.

Version 2.1.7

This version was for internal change tracking only, and never publicly released.

Version 2.1.6

This version was for internal change tracking only, and never publicly released.

Unicode 2.1.5

Modifications made for Version 2.1.5 of UnicodeData.txt include:

- Changed decomposition for U+FF9E and U+FF9F so that correct collation weighting will automatically result from the canonical equivalences.
- Removed canonical decompositions for U+04D4, U+04D5, U+04D8, U+04D9, U+04E0, U+04E1, U+04E8, U+04E9 (the implication being that no canonical equivalence is claimed between these 8 characters and similar Latin letters), and updated 4 canonical decompositions for U+04DB, U+04DC, U+04EA, U+04EB to reflect the implied difference in the base character.
- Added Pi and Pf as General_Category values and assigned these values to the relevant quotation marks, based on the Unicode Technical Corrigendum on Quotation Characters.
- Updated many Bidi_Class values, following the advice of the ad hoc committee on bidi, to make the Bidi_Class values of compatibility characters more consistent.
- Changed General_Category values of several Tibetan characters: U+0F3E, U+0F3F, U+0F88..U+0F8B to make them non-combining, reflecting the combined opinion of Tibetan experts.
- Added case mapping for U+03F2.
- Corrected case mapping for U+0275.
- Added titlecase mappings for U+03D0, U+03D1, U+03D5, U+03D6, U+03F0.. U+03F2.
- Corrected compatibility label for U+2121.
- Add specific entries for all the CJK compatibility ideographs, U+F900..U+FA2D, so the canonical decomposition for each can be carried in the database.

Version 2.1.4

This version was for internal change tracking only, and never publicly released.

Version 2.1.3

This version was for internal change tracking only, and never publicly released.

Unicode 2.1.2

Modifications made in updating UnicodeData.txt to Version 2.1.2 for the Unicode Standard, Version 2.1 (from Version 2.0) include:

- Added two characters (U+20AC and U+FFFC).
- Amended Bidi_Class values for U+0026, U+002E, U+0040, U+2007.
- Corrected case mappings for U+018E, U+019F, U+01DD, U+0258, U+0275, U+03C2, U+1E9B.
- Changed combining order class for U+0F71.
- Corrected canonical decompositions for U+0F73, U+1FBE.
- Changed decomposition for U+FB1F from compatibility to canonical.
- Added compatibility decompositions for U+FBE8, U+FBE9, U+FBF9..U+FBFB.
- Corrected compatibility decompositions for U+2469, U+246A, U+3358.

Version 2.1.1

This version was for internal change tracking only, and never publicly released.

Unicode 2.0.0

The modifications made in updating UnicodeData.txt for the Unicode Standard, Version 2.0 include:

- Changed decompositions for Greek characters with *tonos* to use U+030D.
- Removed entries for the Unicode 1.1 Hangul Syllables block (U+3400..U+4DFF); mapping to the characters in the new Hangul Syllables block are in a separate table.
- Marked compatibility decompositions with additional tags.
- Changed old tag names for clarity.
- Revision of decompositions to use first-level decomposition, instead of maximal decomposition.
- Correction of all known errors in decompositions from earlier versions.
- Added control code names (as comments in the Unicode_1_Name field).
- Added Hangul Jamo decompositions.
- Added Number category to match properties list in book.
- Fixed General_Category values of Koranic Arabic marks.
- Fixed General_Category values of precomposed characters to match decomposition where possible.
- Added Hebrew cantillation marks and the Tibetan script.
- Added place holders for ranges such as CJK Ideographic Area and the Private Use Area.
- Added General_Category values Me, Sk, Pc, Nl, Cs, Cf, and rectified a number of mistakes in the database.

Acknowledgments

Mark Davis and Ken Whistler are the authors of the initial version and have added to and maintained the text of this annex. Julie Allen provided editorial suggestions for improvement of the text. Over the years, many members of the UTC have participated in the review of the UCD and its documentation.

References

For references for this annex, see Unicode Standard Annex #41, "Common References for Unicode Standard Annexes."

Modifications

The following summarizes modifications from previous revisions of this annex.

Revision 3

- Proposed update for Unicode 5.2.0.
- Completely reorganized and rewritten, to include all the content from the obsoleted

UCD.html.

- [Temporary] Added review note at top of text, and modified review notes to use css "reviewnote" class. Added "changedspan" to modifications section.
- Added Section 5.10 re deprecation.

Revision 2

- Initial approved version for Unicode 5.1.0.

Revision 1

- Initial draft.
