UNICODE CHARACTER DATABASE IN XML

Version: Unicode 0.0.0 (draft)
Editor: Eric Muller (eric.muller@tele.net)
Date: 2017-03-14
This Version: http://www.unicode.org/reports/tr42/tr42-20-draft.html
Previous Version: http://www.unicode.org/reports/tr42/tr42-19.html
Latest Version: http://www.unicode.org/reports/tr42/
Latest Proposed Update: http://www.unicode.org/reports/tr42/proposed.html
Schema: http://www.unicode.org/reports/tr42/tr42-20-schema.xsd
Revision: 20

Summary
This annex describes an XML representation of the Unicode Character Database.

Status
This is a draft document which may be updated, replaced, or otherwise displaced 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 comments and other materials with the online reporting form (Feedback). Related information that is useful in understanding this annex is found in Unicode Standard Annex #14, "Common Data Forms 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 [Version]. For any errata which may apply to this annex, see [Errata].

Contents
1 Introduction
2 Overall schema
   2.1 General techniques
   2.2 Name space
   2.3 Databases
   2.4 Root element
   2.5 Common attributes
   2.6 Ordering of elements
3 Description
4 Repertoire
   4.1 Sets of code points
   4.2 Code point types
   4.3 Codes
   4.4 Properties
      4.4.1 Summary
      4.4.2 Name properties
      4.4.3 Other properties
      4.4.4 Robust
      4.4.5 General Category
      4.4.6 Combining properties
      4.4.7 Bidi property
      4.4.8 Decomposition properties
      4.4.9 Script Properties
      4.4.10 Joining properties
      4.4.11 Case properties
      4.4.12 East Asian Width properties
      4.4.13 East Asian Joining properties
      4.4.14 Eidn properties
      4.4.15 DDI Comment properties
      4.4.16 Derived properties
      4.4.17 Derived properties
      4.4.18 Derived properties
      4.4.19 Properties related to function and graphic characteristics
      4.4.20 Properties related to boundaries
      4.4.21 Properties related to decompositions
      4.4.22 Miscellaneous properties
      4.4.23 Unassigned properties
      4.4.24 Other properties
      4.4.25 Other properties
5 Blocks
6 Named Sequences
7 Normalization Corrections
8 Standardized Variants
9 CJK Radicals
10 Footnotes
11 The full schema
12 Examples
Acknowledgments
Modifications

1 Introduction
In working on Unicode implementations, it is often useful to access the full content of the Unicode Character Database (UCD). For example, in establishing mappings from characters to glyphs in fonts, it is convenient to see the character scalar value, the character name, the character East Asian Width, along with the shape and metrics of the proposed glyph to map to. Looking at all this data simultaneously helps in evaluating the mapping.

Directly accessing the data files that constitute the UCD is sometimes a daunting proposition. The data is dispersed in a number of files of various formats, and there are just enough peculiarities (all justified by the processing power available at the time the UCD representation was designed) to require a fairly intimate knowledge of the data format itself, in addition to the meaning of the data.
Many programming environments (for example, Java or C) do give access to the UCD. However, those environments tend to lag behind releases of the standard, or support only some of the UCD content.

Unibook is a wonderful tool to explore the UCD and in many cases is just the ticket; however, it is difficult to use when the task at hand has not been built-in, or when non-UCD data is to be displayed as well.

This annex presents an alternative representation of the UCD, which is meant to overcome these difficulties. We have chosen an XML representation, because parsing becomes a non-issue: there are a number of XML parsers freely available, and using them is often fairly easy. In addition, there are freely available tools that can perform powerful operations on XML data; for example, XPATH and XQUERY engines can be thought of as "grep" for XML data and XSLT can be thought of as "awk" for XML data.

It is important to note that we are interested in exploring the content of the UCD, rather than in using the UCD data to process character streams. Thus, we are not concerned so much by the speed of processing or the size of our representation.

Our representation supports the creation of documents that represent only parts of the UCD, either by not representing all the characters, or by not representing all the properties. This can be useful when only some of the data is needed.

This annex presents only the XML representation format of the UCD. The data itself is part of the Unicode Character Database.

2 Overall schema

2.1 General principles

Our schema can be used to create and validate documents which are intended to represent properties of Unicode code points, blocks, named sequences, normalization corrections, standardized variants, CJK radicals and emoji sources. A document may represent the values actually assigned in a given version of the UCD, or it may represent a draft version of the UCD, or a private agreement on Private Use characters. The validity of a XML document with respect to the schema defined in this annex does not assert anything about the correctness of the values.

Valid documents may provide values for only some of the the code points, or some of the Unicode properties, Furthermore, they may also incorporate non-Unicode properties.

Our schema is defined using English. However, a useful subset of the validity constraints can be captured using a schema language, thereby simplifying the task of validating documents. We have chosen Relax NG [RDF 1.2] in the compact syntax, as the schema language. It is important to stress that the schema which is defined in English imposes more constraints on the documents than can be validated with the Relax NG schema.

An important characteristic of Relax NG is that its schemas do not modify or augment the inofset of the documents. Therefore, it is possible to process our XML representation without using the schema. Also, the schema is relatively straightforward and can be converted mechanically to other schema languages.

While our XML representation is not intended to be used during processing of characters and strings, it is still a design principle for our schema to support the relatively efficient representation of the UCD. This is achieved by an inheritance mechanism, similar to property inheritance in CSS or in XSL-FO (see section 3.2.5.1).

Many invariants impose constraints on the values of the different properties for a given code point. For example, if the value of the Numeric Type property is None, then the value of the Numeric Value property should be the empty string; and if the value of the Other Alphabetic property is true, then the value of the Alphabetic property should be true. Those invariants are not captured in the schema.

2.2 Namespace

The namespace for our elements is "http://www.unicode.org/sr/2003/ucd/1.0/". Our attributes are in the empty namespace.

```xml
<namespace declaration, 1.0> =
default namespace xsd = "http://www.w3.org/2001/XMLSchema"
</namespace declaration>
```

In all our examples, we assume that this namespace is the default one.

2.3 Datatypes

We use a standard XML Schema datatypes:

```xml
<datatypes declaration, 1.0> =
<xs:annotation>
<xs:appinfo>
<xs:documentation>
</xs:appinfo>
</xs:annotation>
</datatypes declaration>
```

Characters are pervasive in the UCD, and will need to be represented. Representing characters directly by themselves would seem the most obvious choice, for example, we could express that the decomposition of U+006F is "a", that is has exactly two characters in (the inofset of) the XML document. However, the current XML specification limits the set of characters that can be part of a document.

Another problem is that the various tools (XML parser, XPATH engine, etc.) may equate U+0068 with U+0065 U+0063 U+0068; thus making it difficult to figure out which of the two sequences is contained in the database (or which is sometimes important for our purposes). Therefore, we choose instead to represent characters by their code points: we follow the usual convention of four to six hexadecimal digits (uppercases) and code points in a sequence separated by spaces; for example, the decomposition of U+0068 will be represented by the nine characters "0065 0063 0068"

```xml
<datatypes for code points, 1.0> =
single-code-point = xsd:string [ pattern = "([1-9A-F]([1-9A-F]([1-9A-F])|0)]+")
one-or-more-code-points = list [ single-code-point + ]
zero-or-more-code-points = list [ single-code-point + ]
two-code-points = list [ single-code-point, single-code-point ]
```

2.4 Root Element

The root element of valid documents is a ucd.

```xml
<schema start, 1.0> =
<element ucd [ ucd.content ]
```

2.5 Common attributes

A large number of properties are boolean. We uniformly use the values Y and N for those.

```xml
<boolean type, 1.0> =
boolean = "Y" | "N"
```

2.6 Ordering of elements

In elements that hold lists of child elements, such as repertoire, group, or standardized-variants, the schema does not require that the child elements be in any particular order.

3 Description

The root element may have a description child element, which in turn contains any string, which is meant to describe what the XML document purports to describe.

It is recommended that if the document purports to represent the UCD of some Unicode version, the description be selected in accord with the rules listed in version; and conversely, that documents which do not purport to represent the UCD be described as such.

```xml
<description, 1.0> =
ucd.content & element description [ text ]
```

4 Repertoire

The repertoire child element of the ucd element describes the code points and their properties. As we will see shortly, code points can be described individually or as part of a group.

```xml
<repertoire, 1.0> =
ucd.content &  
```
4.1 Sets of code points

It is often the case that successive code points have the same property values, for a given set of properties. The most striking example is that of an unbroken plane, where all but the last two code points are reserved and have the same property values. Another example is the URO (U+FE00 .. U+9FA5) where all the code points have the same property values if we ignore their name and their Unihan properties.

This observation suggests that it is profitable to represent sets of code points which share the same properties, rather than individual code points. To make the representation of the sets simple, we restrict them to be segments in the code point space, that is a set is defined by the first and last code point it contains. Those are captured by the attributes first-p and last-p. The attribute up is a shorthand notation for the case where the set has a single code point.

\{Set of code points, \text{B}\} = \text{set-of-code-points} \to <\text{set-of-code-points} >
\{ attribute first-p \to \text{single-code-point}>
\{ attribute last-p \to \text{single-code-point} >
\}

In the repertoire, there must be at least one code-point element for a given code point.

4.2 Code point types

When thinking about Unicode code points, it is useful to split them into four types:

- those assigned to abstract characters (PUA or not)
- the noncharacters
- the surrogate code points
- the reserved code points

This leads to four elements to describe sets of code points:

\{Code points, \text{B}\} = \text{code-point} \to <\text{code-point}>
\{ attribute reserved \to \text{reserved}>
\{ attribute noncharacter \to \text{noncharacter}>
\{ attribute surrogate \to \text{surrrogate}>
\{ attribute char \to \text{char}>
\}

4.3 Group

While we already recognized the situation where a set of code points have exactly the same set of property values, another common situation is that of code points which have almost all the same property values.

For example, the characters U+1740 BUHID LETTER A .. U+1753 BUHID VOWEL SIGN U all have the age "3.2", and all have the script "Buhid". On the one hand, it is convenient to support data files in which those properties are explicitly listed with every code point, at this makes answering questions like "what is the age of U+17497" easier, because data is expressed right there. On the other hand, this leads to rather large data files, and also tends to obscure the differences between similar characters.

Our representation accounts for this situation with the notion of groups. A group element is simply a container of code points that also holds default values for the properties. If a code point inside a group does not list explicitly a property but the group lists it, then the code point inherits that property from its group. For example, the fragment with explicit properties:

\text{char cp1=1740 age=3.2 na=Buhid LETTER A go=Lo sc=Buhid}>
\text{char cp2=1741 age=3.2 na=Buhid LETTER I go=Lo sc=Buhid}>
\text{char cp3=1752 age=3.2 na=Buhid VOWEL SIGN go=Me sc=Buhid}>
\text{char cp4=1820 age=3.2 na=MONGOLIAN LETTER A go=Lo sc=Mong}>

is equivalent to this fragment which uses a group:

\text{group age=3.2 na=Lo sc=Buhid}>
\text{char cp1=1740 age=BUHID LETTER A}>
\text{char cp2=1741 age=BUHID LETTER I}>
\text{char cp3=1752 age=BUHID VOWEL SIGN}>
\text{char cp4=1820 age=MONGOLIAN LETTER A}>

\text{GROUP}>

The element for U+1740 does not have the age attribute, and it therefore inherits it from its enclosing group element, that is "3.2". On the other hand, the element for U+1820 does have this attribute, so the value is "3.0".

As this example illustrates, the notion of group does not necessarily align with the notion of Unicode block. It is entirely defined and limited to our representation. In particular, the value of a property for a code point can always be determined from the XML document alone, assuming that this property and this code point are expressed at all. Of course, one may create an XML representation where the groups happen to coincide with the Unicode blocks.

Groups cannot be nested. The motivation for this limitation is to make the life of consumers easier: either a property is defined by the element for a code point, or it is defined by the immediately enclosing group element.

\text{[groups, 10]} = 
\text{group} \to <\text{group}>
\{ attribute group \to \text{group}>
\{ attribute code-point \to \text{code-point}>
\}

4.4 Properties

Each property, except for the Special Case Condition and Name Alias properties, is represented by an attribute. In an XML data file, the absence of an attribute (may be only on some code-points) means that the document does not express the value of the corresponding property. Conversely, the presence of an attribute is an expression of the corresponding property value; the implied null value is represented by the empty string.

The Name Alias property is represented by zero or more name-alias child elements. Unlike the situation for properties represented by attributes, it is not possible to determine whether all of the aliases have been represented in a data file by inspecting that data file.

The name of an attribute is the abbreviated name of the property as given in the file PropertyAliases.txt in version 6.1.0 of the UCD. For the Unihan properties, the name is that given in the various versions of the Unihan database (some properties are no longer present in version 6.1.0).

For catalog and enumerated properties, the values are those listed in the file PropertyValues.txt in version 6.1.0 of the UCD; if there is an abbreviated name, it is used; otherwise the long name is used.

Note that the set of possible values for a property captured in this schema may change from one version to the next.

4.4.1 Age property
The **name** attribute captures the version of Unicode in which a code point was assigned to an abstract character, or made a surrogate or non-character.

```
<name f="11" c="2.0"> 1.3.2</name>
```

### 4.4.2 Name properties

There are two name properties: the name given by the current version of the standard (\textit{na}), and possibly the name this character had in version 1.0 of the standard (\textit{na1}).

```
<name pattern="[A-Z0-9-:\(\)\;\(\)]" f="12" c=""/>
```

The majority of the characters in Unicode have a name which is of the form CJK UNIFIED IDEOGRAPH\textendash;\textless;code point\textgreater;. It also happens that character names cannot contain the character \texttt{U+0023 \# NUMBER SIGN}, so we adopted the following convention: If a code point has the attribute \textit{na} (either directly or by inference from an enclosing group), then occurrences of the character \texttt{#} in the name are to be interpreted as the value of the code point. For example:

```
<char cap="3400" na="CJK UNIFIED IDEOGRAPH-3400"/>
```

and

```
<char cap="3400" na="CJK UNIFIED IDEOGRAPH-#"/>
```

are equivalent. The \texttt{#} can be in any position in the value of the \textit{na} attribute. The convention also applies just as well to a set of multiple code points:

```
<char cap="3400" na="CJK UNIFIED IDEOGRAPH-3400"/>
```

```
<char cap="3401" na="CJK UNIFIED IDEOGRAPH-3401"/>
```

is equivalent to

```
<char cap="3400" na="CJK UNIFIED IDEOGRAPH-3400"/>
```

```
<char cap="3401" na="CJK UNIFIED IDEOGRAPH-3401"/>
```

which in turn is equivalent to:

```
<char first="3400" last="3401" na="CJK UNIFIED IDEOGRAPH-#"/>
```

### 4.4.3 Name Aliases

The **name-alias** property is represented by zero or more \texttt{name-alias} child elements:

```
<name-alias f="14" c="">
```

### 4.4.4 Block

The **block** property is represented by the \texttt{bk} attribute:

```
<block f="15" c="">
```

http://www.unicode.org/reports/tr42/tr42-20.html
4.4.5 General Category

The general category is represented by the \texttt{gr} attribute.

\begin{verbatim}
{gc property, 16} =
  codepoint-attributes &
  attribute gr = [ "L" | "Lt" | "Lm" | "Ln" | "Lc"
                | "C" | "Cc" | "Cf" | "Cs" | "Cn"
                | "S" | "Sc" | "Kc" | "Kl" | "Z"
                | "H" | "Hc" | "Pi" | "Pd" | "Pn"
                | "M" | "Mo" | "No" ]
\end{verbatim}

4.4.6 Combining properties

The combining class is represented by the \texttt{ccc} attribute, which holds the decimal representation of the combining class.

Because the set of values that this property has taken across the various versions of the UCD is rather large, our schema does not restrict the possible values to those actually used.

\begin{verbatim}
{ccc property, 17} =
  codepoint-attributes &
  attribute ccc { mininteger = 0 maxinclusive = 255 }
\end{verbatim}

4.4.7 Bidirectionality properties

The bidirectional class is represented by the \texttt{b} attribute.

\begin{verbatim}
{bc property, 16} =
  codepoint-attributes &
  attribute b = [ "L" | "Lm" | "Lo" | "R" | "RM" 
              | "Rl" | "Rm" | "RN" | "L" | "LM" | "LK" | "LKl" | "LRO" | "LRE" | "LRI" | "LRL" | "LRC" | "N" ]
\end{verbatim}
The mirrored property is represented by the \texttt{mm} attribute, which takes a boolean value.

\begin{verbatim}
 Área | Área | | Área
| Área | | Área
| Área | | Área

\text{ Área }
\end{verbatim}

The \texttt{bgw} attribute is the code point of a character whose glyph is typically a mirrored image of the glyph for the current character.

\begin{verbatim}
 Área | Área | | Área
| Área | | Área
| Área | | Área

\text{ Área }
\end{verbatim}

The \texttt{Bidi Control} property is represented by the \texttt{bsi} attribute.

\begin{verbatim}
 Área | Área | | Área
| Área | | Área
| Área | | Área

\text{ Área }
\end{verbatim}

The bidi paired bracket type and bidi paired bracket properties are represented by the \texttt{bs} and \texttt{bbs} attributes respectively.

\begin{verbatim}
 Área | Área | | Área
| Área | | Área
| Área | | Área

\text{ Área }
\end{verbatim}

\subsection{4.4.8 Decomposition properties}

The decomposition type and decomposition mapping properties are represented by the \texttt{c} and \texttt{cmp} attributes.

Most characters have a decomposition mapping to themselves. This is similar to the situation we encountered with names, and we adopted a similar convention: if the value of a decomposition mapping is the character itself, we use the attribute value if (-0023 # NUMBER SIGN) as a shorthand notation; this enables those attributes to be captured in groups.

\begin{verbatim}
 Área | Área | | Área
| Área | | Área
| Área | | Área

\text{ Área }
\end{verbatim}

The properties \texttt{Composition Exclusion} and \texttt{Full Composition Exclusion} are represented by the attributes \texttt{ci} and \texttt{cmp_cx}.

\begin{verbatim}
 Área | Área | | Área
| Área | | Área
| Área | | Área

\text{ Área }
\end{verbatim}

\subsection{4.4.9 Numeric Properties}

The numeric type is represented by the \texttt{nt} attribute.

The numeric value is represented by the \texttt{nv} attribute, represented as a fraction.

\begin{verbatim}
 Área | Área | | Área
| Área | | Área
| Área | | Área

\text{ Área }
\end{verbatim}

\subsection{4.4.19 Joining properties}

The joining class of a character is represented by the \texttt{ja} attribute.

The \texttt{ja} attribute is the joining group of the character.

\begin{verbatim}
 Área | Área | | Área
| Área | | Área
| Área | | Área

\text{ Área }
\end{verbatim}
The Join_Control property is represented by the join_c attribute.

The linebreak property is represented by the ln attribute.

The East Asian Width property is represented by the ea attribute.

The Uppercase, Lowercase, Other_Uppercase and Other_Lowercase properties are represented by corresponding attributes.

Most characters have a case mapping and case folding properties that simply map or fold to themselves. This is very similar to the situation we encountered with names, and we adopted a similar convention: if the value of a case mapping or case folding property for the character itself, we use the attribute value "U+0023 # NUMBER SIGN" as a shorthand notation; this enables those attributes to be captured in groups.

The simple case mappings are recorded in the uc, lc, etc attributes.

The non-simple casing are recorded in the uc, lc and cc attributes.
The Simple_Case_Folding and Case_Folding properties are recorded in the sc and cf attributes respectively.

The Case_Ignorable, Cased, Changes_When_Casefolded, Changes_When_Casemapped, Changes_When_Lowercased, Changes_When_NFKC_Casemapped, Changes_When_Titlecased, Changes_When_Uppercased and NFKC_Casefold properties are recorded in these attributes.

Note that the UCD records more information about case folding than is expressed in the properties, specifically the entries in CaseFolding.txt with status T.

4.4.14 Script properties

The script and script extension properties are represented by the sc and ssc attributes respectively.

4.4.15 ISO Comment properties

The ISO 10646 comment field is represented by the isc attribute.

4.4.16 Hangul properties

The property Hangul_Syllable_Type is represented by the hst attribute.

4.4.17 Indic properties

The property Indic_Syllable_Category is represented by the isc attribute.
The property Indic_Matra_Category is represented by the `iwm` attribute:

```
<![IFC property, 41]>
<code-point-attributes &
  attribute iwm ["Right"]
   ["Left"]
   ["Visual_Order_left"]
   ["Left_and_right"]
   ["Top"]
   ["Bottom"]
   ["Top_and_bottom"]
   ["Top_and_right"]
   ["Top_and_left"]
   ["Top_and_left_and_right"]
   ["Bottom_and_right"]
   ["Bottom_and_left"]
   ["Bottom_and_left_and_right"]
   ["Overset"]
   ["Invisible"]
   ["H"]
>]
```

The property Indic_Positional_Category is represented by the `ipc` attribute:

```
<![IFC property, 42]>
<code-point-attributes &
  attribute ipc ["Bottom"]
   ["Bottom_and_right"]
   ["Left"]
   ["Left_and_right"]
   ["Right"]
   ["Top"]
   ["Top_and_bottom"]
   ["Top_and_right"]
   ["Top_and_left"]
   ["Top_and_left_and_right"]
   ["Overset"]
   ["Invisible"]
   ["H"]
>]
```

### 4.4.18 Identifier and Pattern and programming language properties

The properties ID_Start, Other_ID_Start, XID_Start, ID_Continue, Other_ID_Continue, and XID_Continue are represented by corresponding attributes:

```
<![Identifier properties, 43]>
<code-point-attributes &
  attribute ID # (boolean )>
<code-point-attributes &
  attribute IDG # (boolean )>
<code-point-attributes &
  attribute IDG # (boolean )>
<code-point-attributes &
  attribute IDV # (boolean )>
<code-point-attributes &
  attribute IDV # (boolean )>
<code-point-attributes &
  attribute IDV # (boolean )>
<code-point-attributes &
  attribute IDG # (boolean )>
<code-point-attributes &
  attribute IDG # (boolean )>
```

The properties Pattern_Syntax and Pattern_White_Space are represented by corresponding attributes:

```
<![Pattern properties, 44]>
<code-point-attributes &
  attribute Pat_Syntax # (boolean )>
<code-point-attributes &
  attribute Pat_White # (boolean )>
```

### 4.4.19 Properties related to function and graphic characteristics

The properties Dash, Hyphen, Quotation_Mark, Terminal_Punctuation, Sentence_Terminal, Diaiomatic, Extender, Soft_Dotted, Alphabetic, Other_Alphabetic, Math, Other_Math, Hex_Digit, ASCII_Hex_Digit, Default_Ignorable_Code_Point, Other_Default_Ignorable_Code_Point, Logic_Alternate_UnionException, Preceding_Combination_Mark, White_Space, Vertical_Punctuation_and_Nongraphic_Punctuations describe the function or graphic characteristics of a character, and have each a corresponding attribute.

```
<![Properties related to function and graphic characteristics, 45]>
<code-point-attributes &
  attribute Dash # (boolean )>
<code-point-attributes &
  attribute Hyphen # (boolean )>
<code-point-attributes &
  attribute Querc # (boolean )>
<code-point-attributes &
  attribute Tere # (boolean )>
<code-point-attributes &
  attribute Stem # (boolean )>
<code-point-attributes &
  attribute Dis # (boolean )>
```

http://www.unicode.org/reports/tr42/tr42-20.html
4.4.20 Properties related to boundaries

The properties Grapheme_Base, Grapheme_Extend, Other_Grapheme_Extend, Grapheme_Link, Grapheme_Cluster_Break, Word_Break and Sentence_Break each have a corresponding attribute:

{properties related to boundaries, 46} =
\{code-point\-attributes &
  \attribute Grapheme_Extend ( boolean )?
\}

4.4.21 Properties related to ideographs

The properties Ideographic, Unified_Ideograph, IDS_Binary_Operator, IDS_Ternary_Operator and Radical have corresponding attributes:

{properties related to ideographs, 47} =
\{code-point\-attributes &
  \attribute Ideographic ( boolean )?
\}

4.4.22 Miscellaneous properties

The properties Depreciated, Variation_Selector, and Noncharacter_Code_Point have corresponding attributes:

{miscellaneous properties, 48} =
\{code-point\-attributes &
  \attribute Depreciated ( boolean )?
\}
4.4.23 Unihan properties

The Unihan properties (from the Unihan database) are represented as attributes.

```xml
<property name="U+9A4E" value="\u56fe\u9664"/>
```

http://www.unicode.org/reports/tr42/tr42-20.html
4.4.24 Tangut data

The Tangut data are represented as attributes.

{Tangut data, 50}
4.4.25 Nushu data

The Nushu data are represented as attributes.

Nushu data, 55:

```xml
<code-point-attributes & attribute kScn, Nushu> debacle <xs:string pattern='[0-9]+'>1595</xs:string></code-point-attributes & attribute kScn, Nushu>
```

5 Blocks

The blocks child of the ucd describes the blocks. It has one child block element per block, with attributes to describe the extent and name of the block.

```xml
<blocks, 52> =
  ucd.content &
  element blocks {
    element block {
      attribute first-op { single-code-point },
      attribute last-op { single-code-point },
      attribute name { text } } +
  }
```

6 Named Sequences

The named-sequences child of the ucd describes the named sequences. It has one child named-sequence element per named sequence, with attributes to describe the name and sequence.

```xml
[named sequences, 53] =
  ucd.content &
  element named-sequences {
    element named-sequence {
      attribute cpos { one-or-more-code-points },
      attribute name { text } } +
  }
```

7 Normalization Corrections

The normalization-corrections child of the ucd describes the normalization corrections. It has one child normalization-correction element per correction, with attributes to describe the code point affected, its old normalization, its new normalization and the version of Unicode in which the correction was made.

```xml
[normalization corrections, 54] =
  ucd.content &
  element normalization-corrections {
    element normalization-correction {
      attribute cpos { single-code-point },
      attribute old { one-or-more-code-points },
      attribute new { one-or-more-code-points },
      attribute version { text } 
    } +
  }
```

8 Standardized Variants

The standardized-variants child of the ucd describes the standardized variant. It has one child element standardized-variant per variant. The attributes on that last element capture the variation sequence, the description of the desired appearance, and the shaping environment under which the appearance is different.

```xml
[standardized variants, 55] =
  ucd.content &
  element standardized-variants {
    element standardized-variant {
      attribute cpos { two-code-points },
      attribute desc { text },
      attribute when { text } 
    } +
  }
```

9 CJK Radicals

The cjk-radicals child of the ucd describes the CJK radicals. It has one child element cjk-radical per radical. The attributes on that last element capture the radical number, the corresponding CJK radical character, and the corresponding CJK unified ideograph.

```xml
[cjk radicals, 56] =
  ucd.content &
  element cjk-radicals {
    element cjk-radical {
      attribute number { existing [pattern='[0-9](\d)(\d)\w']},
      attribute radical { single-code-point },
      attribute ideograph { single-code-point } +
    }
  }
```

10 Emoji sources

The emoji-sources child of the ucd describes the emoji sources.

```xml
[emoji sources, 58] =
  ucd.content &
  element emoji-sources {
    element emoji-source {
      attribute unicode { one-or-more-code-points },
      attribute idnum { jis-code-point? },
      attribute emoji { jis-code-point? },
      attribute shorthand { jis-code-point? } +
    }
  }
```

11 The full schema

Our schema is just the accumulation of the pieces we have described so far:

```xml
[UCD RelaxNG schema, 59] =
  [namespace declaration: 7] [datatypes: 2, 3, 15, 317] [schema start: 5] [boolean type: 3] [description: 6] [instructions: 7 8 9 102]`
An expanded version is linked from the top of this document.

12 Examples

Here is a fragment of the UCD for a few representative characters (only some of the properties are represented):

```xml
<reportData>
<cnChar cp="0019" age="1,1" name="UNLIT SEPARATOR" go="C" bcs="B" bcl="CM"/>
<cnChar cp="0020" age="1,1" name="SPACE" gc="Zs" bcs="WS" eaa="Na" bcl="BP"/>
<cnChar cp="0026" age="1,1" name="AMPERSAND" gc="Pi" bcs="ON" eaa="Na"/>
<cnChar cp="0028" age="1,1" name="LEFT PARENTHESES" gc="Ps" bcs="ON" bcl="Mn" bclm="0029" eaa="Na" bcl="OP"/>
<cnChar cp="0041" age="1,1" name="LATIN CAPITAL LETTER A" gc="Lu" bcl="0061" eaa="Na" sc="Loca"/>
<cnChar cp="AC00" age="2,3" name="HANGUL SYLLABLE GA" gc="Lo" dcl="can" dnm="1190 1161" eaa="W" bcl="ID" sc="Hang">$
<cnChar cp="30094" age="3,1" name="CJK UNIFIED IDEOGRAPH-2094" gc="Lo" eaa="W" bcl="ID" sc="Hang"/>
</reportData>
</ucd>
```

Acknowledgments

Thanks to Markus Schecker and Mark Davis for their help developing this XML representation. Thanks to the reviewers: Julie Allen, Ernest van den Bougard, Daniel Banitz, John Cowan, Asmus Freytag, Felix Sasaki, Andrew West.

Modifications

This section indicates the changes introduced by each revision.

Revision 20

- (draft 3) Modified patterns for the `kernSource` attributes.
- (draft 3) New values for the `bidi` attribute: `knx`, `kndr`, `kmsr`, `kndw`, `knsh`, `kndm`, `kndg`, `kndhh`, `knbd`, `kndb`, `kndk`, `kniv`, `kndi`, `knrm`, `knsw`, `kndt`, `kndh`, `knjd`, `kndj`, `knne`, `kndn`, `kngr`, `kndr`, `knss`. These new values allow for kerning to be specified for individual characters, as well as for character pairs.
- (draft 3) New code point attributes for Nusho data: `knx`, `knsh`, `knsw`, and `knsw`. These attributes allow for kerning to be applied to specific code points in the Nusho script.
- (draft 3) New values for the `bidi` attribute: `knx`, `kndr`, `kmsr`, `kndw`, `knsh`, `kndm`, `kndg`, `kndhh`, `knbd`, `kndb`, `kndk`, `kniv`, `kndi`, `knrm`, `knsw`, `kndt`, `kndh`, `knjd`, `kndj`, `knne`, `kndn`, `kngr`, `kndr`, `knss`. These new values allow for kerning to be specified for individual characters, as well as for character pairs.
- (draft 3) New code point attributes for Nusho data: `knx`, `knsh`, `knsw`, and `knsw`. These attributes allow for kerning to be applied to specific code points in the Nusho script.

Revision 19

- New value for the `ase` attribute: `kix`, `kixt`.
- New values for the `sc` attribute: `ada`, `bin`, `marc`, `new`, `다가`, `tang`.
- New values for the `bidi` attribute: `knx`, `kndr`, `kmsr`, `kndw`, `knsh`, `kndm`, `kndg`, `kndhh`, `knbd`, `kndb`, `kndk`, `kniv`, `kndi`, `knrm`, `knsw`, `kndt`, `kndh`, `knjd`, `kndj`, `knne`, `kndn`, `kngr`, `kndr`, `knss`. These new values allow for kerning to be specified for individual characters, as well as for character pairs.
- New code point attributes for Nusho data: `knx`, `knsh`, `knsw`, and `knsw`. These attributes allow for kerning to be applied to specific code points in the Nusho script.
- Modified patterns for the `skxsi` and `skxstr` attributes.

Revision 18 being a proposed update, only changes between revisions 17 and 19 are noted here.

Revision 17

- New value for the `ade` attribute: `kix`, `kixt`.
- New values for the `sc` attribute: `knd`, `kndr`, `kmsr`, `kndw`, `knsh`, `kndm`, `kndg`, `kndhh`, `knbd`, `kndb`, `kndk`, `kniv`, `kndi`, `knrm`, `knsw`, `kndt`, `kndh`, `knjd`, `kndj`, `knne`, `kndn`, `kngr`, `kndr`, `knss`. These new values allow for kerning to be specified for individual characters, as well as for character pairs.
- New code point attributes for Nusho data: `knx`, `knsh`, `knsw`, and `knsw`. These attributes allow for kerning to be applied to specific code points in the Nusho script.
- Modified patterns for the `skxsi` and `skxstr` attributes.
Switched the reference to ISO 19757 from 2003 and 2003 Amd1 to 2008.

Revision 15
- New value for the aae attribute: 7.e.
- New values for the s5 attribute.
- New values for the sc attribute.
- New values for the s1 attribute.
- New values for the scc attribute.
- New values for the scscc attribute.
- New values for the xinhsource attribute.
- New values for the xinh_source attribute.

Revision 14 being a proposed update, only changes between revisions 13 and 15 are noted here.

Revision 13
- New value for the aae attribute: 7.e.
- New values sq, ml, sl for the w1 attribute (for Unicode 6.3).
- New code point attributes sre and sre for (for Unicode 6.3).
- New values for the q attribute: 7, 11, 15, 17, 21, 25, 29 (for Unicode 6.3).
- Updated the patterns for xinhSource and xinhSource for (for Unicode 6.3).
- Updated the patterns for xinhSource and xinhSource for (for Unicode 6.2).
- Clarified that the child elements list-like elements are in no particular order.

Revision 12 being a proposed update, only changes between revisions 11 and 13 are noted here.

Revision 11
- New value for the aae attribute: 7.e.
- New value for the gen, w1 and t1 attributes: 17 (for Unicode 6.3).
- Updated the patterns for xinhSource and xinhSource for (for Unicode 6.2).

Revision 10 being a proposed update, only changes between revisions 9 and 11 are noted here.

Revision 9
- Clarified the default values.
- Indicate that property values may change from one release to the next.
- Introduced the new attributes, for the blue property.
- Introduced the sre attribute, for the ScriptExtensions property.
- Introduced the vname-alias element, for the VNameAlias property.
- New value for the aae attribute: 6.1.
- New values for the script attribute: Cnax, Mebo, Mero, Pind, Shrd, Sora, Tavt.
- New values for the sre attribute: 61.
- New value for the s5 attribute: kient, yeh.
- The value of the rycrcs attribute must now be either # of one-or-more-code-points.
- For the n value, the absence of a numeric value is now represented by \sre rather than by the empty string.
- The values of the sre are now restricted to 0..254, instead of 0..255.
- Updated the patterns for xseinvariant, xspecializesXaisvariant, xinhSource and xinhSource.

Revision 8 being a proposed update, only changes between revisions 7 and 9 are noted here.

Revision 7
- New value for the aae attribute: 6.8.
- New value for the s5 attribute: yen, werbts, goals.
- New values for the sre attribute: ban, krah, man.
- Updated the patterns for xinhSource, xinhSource, xinhSource, xinhSource, xinhSource, xinhSource.
- Added the scs and sce elements.
- Added the source-sources element.

Revision 6 being a proposed update, only changes between revisions 5 and 7 are noted here.

Revision 5
- Changed the type of block@first, block@last, and normalization-corrections@ep from text to single-code-point.
- Changed the type of named-sequence@aps, provision-aligned-sequence@aps, normalization-correction@ep and normalization-correction@ep from text to one-or-more-code-points.
- Changed the type of standardized-variants@ps from text to two-code-points.
- New values for the s5 attribute: farsil, yen.
- New value for the aae attribute: 5.2.
- New values for the sc attribute: tava, sv, tawa, tawa, java, mit, amr, sarh, phk, phii, phii.
- New value for the s1 attribute: cfr.
- New value for the sc attribute: taw.
- New code point attributes ci, cavi, cawi, cawi, cawi, cawi, cawi, cawi, cawi, cawi, cawi, cawi, cawi, cawi.
- New attributes kierntajo and xinhSource.
- New element ci=radicals.
Revision 4 being a proposed update, only changes between revisions 3 and 5 are noted here.

Revision 3

- First approved version, for Unicode 5.1.0.
- For optional elements which acts as collections, such as *repeated* and *named-sequences*, impose that there be at least one elements in the collection.
- Remove the constraint that the value of `is` is limited when `is` has certain values; similarly for `isSensitive` and `isNotSensitive`.
- Value `w` added to the `www` attribute (for Unicode 5.1).
- Value `w` added to the `s` attribute (for Unicode 5.1).
- Corrected the `val` script value to `sval`.
- Removed the discussion of elements or attributes in different namespace.
- Removed the `code-point` element.

Revision 2

- Promoted to Draft UAX.
- Changed the title from "An XML representation of the UCD".
- Value `s` added to the `age` attribute (for Unicode 5.1).
- Value `w` added to the `www` attribute (for Unicode 5.1).
- Values `c`, `extend`, `de`, `re` added to the `w` attribute (for Unicode 5.1).
- Values `c`, `e`, `d`, `r` added to the `w` attribute (for Unicode 5.1).
- Value `user-friendly` added to the `w` attribute (for Unicode 5.1).
- Value `isxn.childNodes` added to the `w` attribute (for Unicode 5.1).
- Value `isxn.parentNode` added to the `w` attribute (for Unicode 5.1).
- Values `crlf`, `crlfLine`, `text`, `crlfLine`, `tick`, `king`, `sour`, `surf` and `val` added to the `s` attribute (for Unicode 5.0).
- `isxn.attribute` renamed to `isxn`
- `src` attribute renamed to `isxn`
- Attribute `keywords` added (for Unicode 5.1.0).
- Pattern for attribute `kmlSource` extended (for Unicode 5.1.0).
- Element `provisional-name-sequences` added (for Unicode 5.0)

Revision 1

- First working draft.

© 2017 Unicode, Inc. All Rights Reserved. The Unicode Consortium makes no expressed or implied warranty of any kind, and assumes no liability for errors or omissions. No liability is assumed for incidental or consequential damage in connection with or arising out of the use of the information or programs contained or accompanying this technical report. The Unicode properties data files may not be used to generate or support such a database.

Unicode and the Unicode logo are trademarks of Unicode, Inc, and are registered in some jurisdictions.