UNICODE CHARACTER DATABASE IN XML

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 needed).

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

2 Overall schema

2.1 General principles

This annex presents only the XML representation format of the UCD. The data itself is part of the Unicode Character Database.
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 [ISO 19757], 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 infoset 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 invariance in CSS or in XSLT (see section 2.3.1). 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. Thus, if the property Other Alphabetic 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/ma2003/ucd/1.0. Our attributes are in the empty namespace.

[namespace declaration, 1] =
    default namespace ucd = "http://www.unicode.org/ma2003/ucd/1.0"

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

2.3 Datatypes

We use a standard XML Schema data-types:

[datatypes declaration, 2] =
 _schema start = http://www.w3.org/2001/XMLSchema
[datatypes, 3] =
  <xs:schema targetNamespace="http://www.unicode.org/ns/2003/ucd/1.0"
           xmlns:ucd = "http://www.unicode.org/ma2003/ucd/1.0"
           elementFormDefault="qualified"
           attributeFormDefault="lowercase">

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+00E8 is “U+0065 U+0300” (that is have exactly two characters in (the infoset of) the XML document. However, the current XML specificatiion 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+00E8 with U+0065 U+0300, thus making it difficult to figure out which of the two sequences is contained in the database (which is sometimes important for our purposes). Therefore, we chose instead to represent characters by their code points, we follow the usual convention of four to six hexadecimal digits (uppercase) and code points in a sequence separated by space; for example, the decomposition of U+00E8 will be represented by the nine characters “0065 0300” in the infoset.

[datatypes for code points, 3] =
  _single-code-point = _xstring { pattern = "([A-F]| [0-9]){4,6}"
  _one-or-more-code-points = list [ _single-code-point ]
  _zero-or-one-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 ucd.

[schema start, 4] =
  _start = ucd [ ucd.content ]

2.5 Common attributes

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

[boolean type, 5] =
  _booleans = "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 [Versions]; and conversely, that documents which do not purport to represent the UCD be described as such.

[description, 6] =
  _description =
    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:

[repertoire, 7] =
  _repertoire =
    ucd.content [ element _repertoire [ (code-point | group) + ]

4.1 Sets of code points

It is often the case that successive code points have the the same property values, for a given set of properties. The most striking example is that of an unallocated plane, where all but the last two code points are reserved and have the same property values. Another example is the URO (U+4E00 ... 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-cp and last-cp. The attribute cp is a shorthand notation for the case where the set has a single code point.

[Set of code points, 8] =
  _set-of-code-points =
    attribute cp [ _single-code-point ]
    attribute first-cp [ _single-code-point ]
    attribute last-cp [ _single-code-point ]

In the repertoire, there must be at most 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 (PLA 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, 9] =
  code-point =
    _element reserved |
      _set-of-code-points, _code-point-attributes
    code-point =
      _element noncharacter |
      _set-of-code-points, _code-point-attributes
    code-point =
      _element surrogate |
      _set-of-code-points, _code-point-attributes
For catalog and enumerated properties, the values are those listed in the file PropertyValueAliases.txt in version 6.1.0 of the UCD; if there is an abbreviated name, it is used, otherwise the long name is used.

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 data file by inspecting that data file.

### 4.4.1 Age property

```
<char cp="1741" age="3.2" na="BUHID LETTER I" gc="Lo" sc="Buhd"/>
```

is equivalent to this fragment which uses a group:

```
<group age="3.2" gc="Lo" sc="Buhd">
  <char cp="1740" na="BUHID LETTER A"/>
  <char cp="1741" na="BUHID LETTER I"/>
</group>
```

The Name_Alias property is represented by zero or more

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

which in turn is equivalent to:

```
<char cp="3401" na="CJK UNIFIED IDEOGRAPH-3401"/>
<char cp="3400" na="CJK UNIFIED IDEOGRAPH-3400"/>
```

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

### 4.4.2 Name properties

There are two name properties: the name given by the current version of the standard (\[name\]), and possibly the name this character had in version 1.0 of the standard (\[name, 1\]).

```
\[name, pattern, 12\] =
  \[character-name, 15\] =
```

### 4.4.3 Name Aliases

```
\[name alias property, 14\] =
```

```
\[name alias, 15\] =
```

### 4.4.4 Block

```
\[block property, 15\] =
```

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:

```
<group age="3.2" gc="Lo" sc="Buhd">
  <char cp="1740" na="BUHID LETTER A"/>
  <char cp="1741" na="BUHID LETTER I"/>
</group>
```

is equivalent to this group which uses a group:

```
<group age="3.2" gc="Lo" sc="Buhd">
  <char cp="1740" na="BUHID LETTER A"/>
  <char cp="1741" na="BUHID LETTER I"/>
</group>
```

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.

### 4.4.4 Block

```
\[block property, 15\] =
```

The \[age\] attribute captures the version of Unicode in which a code point was assigned to an abstract character, or made a surrogate or non-character.

```
\[age, 1\] =
```

```
\[name pattern, 12\] =
```

```
```

### 4.4.5 Name property

The \[age\] attribute is an expression of the corresponding property; 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.

For catalog and enumerated properties, the values are those listed in the file PropertyValueAliases.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).

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 \[age\] attribute captures the version of Unicode in which a code point was assigned to an abstract character, or made a surrogate or non-character.

```
\[age, 1\] =
```

```
```

```
```

### 4.4.2 Name properties

There are two name properties: the name given by the current version of the standard \[na\], and possibly the name this character had in version 1.0 of the standard \[na, 1\].

```
\[name pattern, 12\] =
```

```
```

### 4.4.3 Name Aliases

```
```

```
```

4.4.4 Block

```
```

The Name_Alias property is represented by zero or more \[name alias\] child elements.

```
\[name alias property, 14\] =
```

```
```

```
```
4.4.5 General Category

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

\begin{verbatim}
[gc property, 16] =
  code-point-attributes &=
   attribute gc { "Lu" | "Ll" | "Lt" | "Lm" | "Lo"
   | "Mn" | "Mc" | "Me"
   | "Nd" | "Nl" | "No"
   | "Pc" | "Pd" | "Ps" | "Pe" | "Pi" | "Pf" | "Po"
   | "Sm" | "Sc" | "Sk" | "So"
   | "Zs" | "Zl" | "Zp"
   | "Cc" | "Cf" | "Cs" | "Co" | "Cn"}
\end{verbatim}

4.4.6 Combining properties
The combining class is represented by the `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.

```xml
<ccc property, 17> =
code-point-attributes =
attribute ccc { xsd:integer { minInclusive="0" maxInclusive="254" }}?
</ccc>
```

### 4.4.7 Bidirectionality properties

The bidirectional class is represented by the `bc` attribute.

```xml
<bc property, 18> =
code-point-attributes =
attribute bc { "AL" | "AN" | "B " | "BN" | "CS" | "EN" | "ES" | "ET" |
FSI" | "L" | "LRE" | "LRI" | "LRO" | "NSM" | "ON" |
PDF | "PDI" | "R" | "RLE" | "RLI" | "RLO" | "S" | "WS" }?
</bc>
```

The mirrored property is represented by the `Bidi_M` attribute, which takes a boolean value.

```xml
<Bidi_M property, 19> =
code-point-attributes =
attribute Bidi_M { boolean }?
</Bidi_M>
```

The `bmg` attribute is the code point of a character whose glyph is typically a mirrored image of the glyph for the current character.

```xml
<strong>4.4.8 Decomposition properties</strong>

The decomposition type and decomposition mapping properties are represented by the `dt` and `dm` attributes.

Most characters have a decomposition mapping to themselves. This is very 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 U+0023 # NUMBER SIGN as a shorthand notation; this enables those attributes to be captured in groups.

```xml
<decomposition properties, 23> =
code-point-attributes =
attribute dt { "can" | "com" | "enc" | "fin" | "font" | "fra" |
"init" | "iso" | "med" | "nar" | "nb" | "sml" |
sqr" | "sub" | "sup" | "vert" | "wide" | "none" }?
code-point-attributes =
attribute dm { "#" | zero-or-more-code-points }?
</decomposition properties>
```

The properties Composition_Exclusion and Full_Composition_Exclusion are represented by the attributes `ce` and `comp_xc`:

```xml
<composition properties, 24> =
code-point-attributes =
attribute ce { boolean }?
code-point-attributes =
attribute comp_xc { boolean }?
</composition properties>
```

The properties NFC_Quick_Check, NFD_Quick_Check, NFKC_Quick_Check, NKFD_Quick_Check, Expands_On_NFC, Expands_On_NFD, Expands_On_NFKC, Expands_On_NKFD, FC_NFKC_Closure have corresponding attributes.

```xml
<quick check properties, 25> =
code-point-attributes =
attribute NFC_QC { "Y" | "N" | "M" }?
code-point-attributes =
attribute NFD_QC { "Y" | "N" }?
code-point-attributes =
attribute NFKC_QC { "Y" | "N" | "M" }?
code-point-attributes =
attribute NFKD_QC { "Y" | "N" }?
code-point-attributes =
attribute XO_NFC { boolean }?
code-point-attributes =
attribute XO_NFD { boolean }?
code-point-attributes =
attribute XO_NFKC { boolean }?
code-point-attributes =
attribute XO_NKFD { boolean }?
code-point-attributes =
attribute FC_NFKC { "#" | one-or-more-code-points }?
</quick check properties>
```

### 4.4.9 Numeric Properties

The numeric type is represented by the `nt` attribute.

The numeric value is represented by the `nv` attribute, represented as a fraction.

```xml
<numeric properties, 26> =
code-point-attributes =
attribute nt { "None" | "De" | "Di" | "Nu" }?
code-point-attributes =
attribute nv { "NaN" | xsd:string { pattern = "-?[0-9]+/\[0-9\]+" } }?
</numeric properties>
```

### 4.4.10 Joining properties

The joining class of a character is represented by the `jt` attribute.

The `jg` attribute is the joining group of the character.

```xml
<joining properties, 27> =
```
code-point-attributes be
  attribute 35 [ ; | ; | ; | ; | ; ]
code-point-attributes be
  attribute 39 [ ; | ; ]

| \* \* | \* \* | \* \* | \* \* | \* \* | \* \* |
| \* \* | \* \* | \* \* | \* \* | \* \* | \* \* |
| \* \* | \* \* | \* \* | \* \* | \* \* | \* \* |
| \* \* | \* \* | \* \* | \* \* | \* \* | \* \* |
| \* \* | \* \* | \* \* | \* \* | \* \* | \* \* |

4.4.11 Linebreak properties
The linebreak property is represented by the \* attribute.

4.4.12 East Asian Width property
The East Asian width property is represented by the \* attribute.

4.4.13 Case properties
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 is the character itself, we use the attribute value \*U+0023 as a shorthand notation; this enables those attributes to be captured in groups.

The Simple_Case_Folding and Case_Folding properties are recorded in the \* and \* attributes.

The non-simple case are recorded in the \* and \* attributes.

The Simple_Case_Folding and Case_Folding properties are recorded in the \* and \* attributes respectively.
The Case_Ignorable, Cased, Changes_When_Casefolded, Changes_When_Casemapped, Changes_When_Lowercased, Changes_When_Uppercased and NKFC_Casefold properties are recorded in these attributes:

4.4.14 Script properties

The script and script extension properties are represented by the sc and scx 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.

The property Jamo_Short_Name is represented by the JSN attribute:

4.4.17 Indic properties

The property Indic_Syllabic_Category is represented by the InSC attribute.

4.4.18 Indic properties

The property Indic_Syllabic_Category is represented by the InSC attribute.
The property `Indic_Matra_Category` is represented by the `InMC` attribute:

\[
\text{[InMC property, 41]} = \text{code-point-attributes} \land \text{attribute InMC} \{ \text{"Right"}, \text{"Left"}, \text{"Visual_Order_Left"}, \text{"Left_And_Right"}, \text{"Top"}, \text{"Bottom"}, \text{"Top_And_Right"}, \text{"Top_And_Bottom"}, \text{"Top_And_Bottom_And_Right"}, \text{"Top_And_Left"}, \text{"Top_And_Left_And_Right"}, \text{"Top_And_Right"}, \text{"Top_And_Bottom_And_Right"}, \text{"Overstruck"}, \text{"Invisible"}, \text{"NA"} \}
\]

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

\[
\text{[InPC property, 42]} = \text{code-point-attributes} \land \text{attribute InPC} \{ \text{"Bottom"}, \text{"Bottom_And_Right"}, \text{"Left"}, \text{"Left_And_Right"}, \text{"NA"}, \text{"Overstruck"}, \text{"Right"}, \text{"Top"}, \text{"Top_And_Bottom"}, \text{"Top_And_Bottom_And_Right"}, \text{"Top_And_Left"}, \text{"Top_And_Left_And_Right"}, \text{"Top_And_Right"}, \text{"Visual_Order_Left"} \}
\]

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:

\[
\text{[identifier properties, 43]} = \text{code-point-attributes} \land \text{attribute IDS} \{ \text{boolean} \} \land \text{attribute OIDS} \{ \text{boolean} \} \land \text{attribute XIDS} \{ \text{boolean} \} \land \text{attribute IDC} \{ \text{boolean} \} \land \text{attribute OIDC} \{ \text{boolean} \} \land \text{attribute XIDC} \{ \text{boolean} \}
\]

The properties `Pattern_Syntax` and `Pattern_White_Space` are represented by corresponding attributes:

\[
\text{[pattern properties, 44]} = \text{code-point-attributes} \land \text{attribute Pat_Syn} \{ \text{boolean} \} \land \text{attribute Pat_WS} \{ \text{boolean} \}
\]

4.4.19 Properties related to function and graphic characteristics

The properties `Dash`, `Hyphen`, `Quotation_Mark`, `Terminal_Punctuation`, `Sentence_Terminal`, `Diacritic`, `Extender`, `Soft_Dotted`, `Alphabetic`, `Other_Alphabetic`, `Math`, `Other_Math`, `Hex_Digit`, `ASCII_Hex_Digit`, `Default_Ignorable_Code_Point`, `Other_Default_Ignorable_Code_Point`, `Logical_Order_Exception`, `Dependent_Collaboration`, `Max` and `White_Space` describe the function or graphic characteristic of a character, and have each a corresponding attribute.

\[
\text{[properties related to function and graphic characteristics, 45]} = \text{code-point-attributes} \land \text{attribute Dash} \{ \text{boolean} \} \land \text{attribute Hyphen} \{ \text{boolean} \} \land \text{attribute QMark} \{ \text{boolean} \} \land \text{attribute Term} \{ \text{boolean} \} \land \text{attribute STerm} \{ \text{boolean} \} \land \text{attribute Dia} \{ \text{boolean} \} \land \text{attribute Ext} \{ \text{boolean} \} \land \text{attribute PCM} \{ \text{boolean} \} \land \text{attribute SD} \{ \text{boolean} \} \land \text{attribute Alpha} \{ \text{boolean} \} \land \text{attribute OAlpha} \{ \text{boolean} \} \land \text{attribute Math} \{ \text{boolean} \} \land \text{attribute OMath} \{ \text{boolean} \} \land \text{attribute Hex} \{ \text{boolean} \} \land \text{attribute OHex} \{ \text{boolean} \} \land \text{attribute Default_Ignorable_Code_Point} \{ \text{boolean} \} \land \text{attribute Other_Default_Ignorable_Code_Point} \{ \text{boolean} \} \land \text{attribute Logical_Order_Exception} \{ \text{boolean} \} \land \text{attribute Dependent_Collaboration} \{ \text{int} \} \land \text{attribute Max} \{ \text{int} \} \land \text{attribute White_Space} \{ \text{boolean} \}
\]
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:

```xml
<attribute>
  <name>Gr_Base</name>
</attribute>
```

```xml
<attribute>
  <name>Gr_Ext</name>
</attribute>
```

```xml
<attribute>
  <name>OGr_Ext</name>
</attribute>
```

```xml
<attribute>
  <name>Gr_Link</name>
</attribute>
```

4.4.21 Properties related to ideographs

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

```xml
<attribute>
  <name>Ideo</name>
</attribute>
```

```xml
<attribute>
  <name>UIdeo</name>
</attribute>
```

```xml
<attribute>
  <name>IDSB</name>
</attribute>
```

```xml
<attribute>
  <name>IDST</name>
</attribute>
```

4.4.22 Miscellaneous properties

The properties Deprecated, Variation_Selector and Noncharacter_Code_Point have corresponding attributes:

```xml
<attribute>
  <name>Dep</name>
</attribute>
```

```xml
<attribute>
  <name>VS</name>
</attribute>
```

```xml
<attribute>
  <name>NChar</name>
</attribute>
```

4.4.23 Unihan properties

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

```xml
<attribute>
  <name>kAccountingNumeric</name>
</attribute>
```

```xml
<attribute>
  <name>kAlternateHanYu</name>
</attribute>
```

```xml
<attribute>
  <name>kAlternateJEF</name>
</attribute>
```

```xml
<attribute>
  <name>kAlternateKangXi</name>
</attribute>
```

```xml
<attribute>
  <name>kAlternateMorohashi</name>
</attribute>
```

```xml
<attribute>
  <name>kBigFive</name>
</attribute>
```

```xml
<attribute>
  <name>kCCCII</name>
</attribute>
```

```xml
<attribute>
  <name>kCNS1986</name>
</attribute>
```

```xml
<attribute>
  <name>kCNS1992</name>
</attribute>
```

```xml
<attribute>
  <name>kCangjie</name>
</attribute>
```

```xml
<attribute>
  <name>kCantonese</name>
</attribute>
```

```xml
<attribute>
  <name>kCheungBauer</name>
</attribute>
```

```xml
<attribute>
  <name>kCheungBauerIndex</name>
</attribute>
```

```xml
<attribute>
  <name>kCihaiT</name>
</attribute>
```

```xml
<attribute>
  <name>kCompatibilityVariant</name>
</attribute>
```
The Tangut data are represented as attributes.

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.

[block, 51] =
  ucd

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.

[named sequences, 52] =
  ucd
  | element named-sequences [ element named-sequence [ attribute cps : 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.

[normalization corrections, 53] =
  ucd
  | element normalization-corrections [ element normalization-correction [ attribute name : text , attribute new : one-or-more-code-points , attribute old : one-or-more-code-points , attribute cps : two-code-points ] ] + ]

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.

[standardized variants, 54] =
  ucd
  | element standardized-variants [ element standardized-variant [ attribute name : text , attribute cps : two-code-points , attribute new : one-or-more-code-points , attribute old : one-or-more-code-points ] ] + ]

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.

[cjk radicals, 55] =
  ucd
  | element cjk-radicals [ element cjk-radical [ attribute name : text , attribute cps : two-code-points , attribute new : one-or-more-code-points , attribute old : one-or-more-code-points , attribute ideograph : single-code-point ] ] + ]

10 Emoji sources
The emoji-sources child of the ucd describes the emoji sources.
11 The schema

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

```
<Data type for code points, 56> =
    [jis-code-point | xml-string { pattern = "[0-9A-F]{4}" } ]

[emoji sources, 57] =
    <content type> { content } + { content }?
```

11 Examples

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

```
<ucd xmlns="http://www.unicode.org/reports/2003/ucd1.0"> 
  <chapter>
    <char cp="0001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"WS"</bc>
      <ea>"Na"</ea>
      <lb>"SP"</lb>
    </char>
    <char cp="0028" age="1.1" na="LEFT PARENTHESIS" na1="OPENING PARENTHESIS">
      <gc>"Ps"</gc>
      <bc>"ON"</bc>
    </char>
    <char cp="0026" age="1.1" na="AMPERSAND" gc="Po" bc="ON">
      <ea>"Na"</ea>
    </char>
    <char cp="0032" age="1.1" na="NUMBER SIGN" gc="Zs" bc="WS">
      <lb>"CM"</lb>
    </char>
    <char cp="0020" age="1.1" na="SPACE" gc="Zs" bc="WS">
      <ea>"Na"</ea>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
    <char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR">
      <gc>"Cc"</gc>
      <bc>"S"</bc>
      <lb>"CM"</lb>
    </char>
  </chapter>
</ucd>
```

Acknowledgments

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

Modifications

This section indicates the changes introduced by each revision.

Revision 18

- New values for the s attribute: s.
- New values for the s attribute: Adlm, Bna, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrc, Mrk, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mrk, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mrk, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mrk, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mrk, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mrk, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mrk, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mrk, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mrk, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mrk, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mrk, Mro, Sgzn, Tang.
- Modified patterns for the AbetilnCode, Kebtiang, Kebtiang, Kebtiang, Kebtiang, Kebtiang, Kebtiang, Kebtiang, Kebtiang, Kebtiang attributes.

Revision 17

- New values for the s attribute: s.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.
- New values for the s attribute: Adlm, Mrk, Mrc, Mro, Sgzn, Tang.

Revision 16 being a proposed update, only changes between revisions 15 and 17 are noted here.

Revision 15

- New values for the s attribute: s.
- New values for the s attribute: s.
* New values for the **attribute.
* New values for the **attribute.
* New values for the **attribute.
* New values for the **attribute.
* New values for the **attribute.

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

Revision 13

* New value for the **attribute `c`.
* New values cp, ms, mp for the **attribute (for Unicode 6.3).
* New code point attributes ** and ** (for Unicode 6.3).
* New values for the **attribute **, **, **, ** (for Unicode 6.3).
* Updated the patterns for ** and ** (for Unicode 6.3).
* Updated the patterns for ** and ** (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 **attribute `c`.
* New value for the ** and ** attributes: ** (for Unicode 6.2).
* Updated the patterns for ** and ** (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 ** attributes, for the ** property.
* Introduced the ** attribute, for the ** property.
* New value for the **attribute `c`.
* New values for the **attribute `n` and `c`.
* New value for the **attribute: **.
* The value of the **attribute must now be either # or one-or-more-code-points.
* For the ** attribute, the absence of a numeric value is now represented by ** rather than by the empty string.
* The values of the ** are now restricted to **, instead of **.
* Updated the patterns for **, **, **, **; ** and **.

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

Revision 7

* New value for the **attribute `c`.
* New value for the **attribute: **.
* New value for the code attribute: **, **, **.
* Updated the patterns for ** and **; **.
* Updated the ** and ** elements.
* Added the ** element.
* Added the ** element.

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

Revision 5

* Changed the type of **; ** and ** from ** to **.
* Changed the type of **; **, **; **; **; ** from ** to **.
* Changed the type of **; ** from ** to **.
* New values for the ** attribute: **.
* New value for the ** attribute `c`.
* New value for the ** attribute: **.
* New value for the ** attribute: **.
* New code point attributes **.
* New attributes ** and **.
* New element **.
* Updated the patterns for **, **, **; **; **, **; **; **; **; **; **.
* Point out that Relax NG schemas do not modify or augment the infoset, and that it is possible to convert mechanically our schema to other schema languages.

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.
* For optional elements which acts as collections, such as ** and **, impose that there be at least one element in the collection.
* Remove the constraint that the value ** is limited when ** has certain values; similarly for ** and **.
* Value ** added to the ** attribute (for Unicode 5.1).
* Value ** added to the ** attribute (for Unicode 5.1).
- Corrected the Vai script value to Vaii.
- 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 1.1 added to the age attribute (for Unicode 5.1).
- Value cm, mm, cm added to the ce attribute (for Unicode 5.1).
- Values cm, mm, cm added to the ce attribute (for Unicode 5.1).
- Value Burushaski_Yeh_Barree added to the jg attribute (for Unicode 5.1).
- Value Alef_Maqsurah added to the jg attribute (for Unicode 2.x).
- Value CR, LF, MB, Extend added to the WB attribute (for Unicode 5.1).
- Value CR, EX, LF, SC added to the SB attribute (for Unicode 5.1).
- Value Burushaski_Yeh_Barree, Alef_Maqsurah added to the jg attribute (for Unicode 5.1).
- Attribute attribute renamed to xsn
- Attribute attribute renamed to xcf
- Pattern for attribute xsn extended (for Unicode 5.1).
- Element provisional-named-sequences added (for Unicode 5.0)