UNICODE CHARACTER DATABASE IN XML

Version       | unicode (draft 1)
Editor        | Eric Muller (eric@adobe.com)
Date          | 2015-03-02
This Version   | http://www.unicode.org/reports/tr42/tr42-16.html
Previous Version | http://www.unicode.org/reports/tr42/tr42-15.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/tr42-16.xml
Revision      | 2

Summary
This annex describes an XML representation of 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This is a draft document which may be updated, replaced, or superseded by other documents at any time.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 [Online]. For a list of current Unicode Technical Reports see [Reports]. For more information about versions of the Unicode Standard see [History]. For any errata which may apply to this annex, see [Draft].

Contents

1 Introduction
  2 Overall schema
    2.1 General principles
    2.2 Namespace
    2.3 Elements
    2.4 Attributes
    2.5 Character attributes
    2.6 Ordering of elements
3 Description
    4 Reference
      4.1 Bana code points
      4.2 Code point types
      4.3 Unicode
      4.4 Properties
        4.4.1 Use properties
        4.4.2 Name properties
        4.4.3 Name Notes
        4.4.4 Block
        4.4.5 Additional Category
        4.4.6 Combining Properties
        4.4.7 Isolation Property
        4.4.8 Decomposition Properties
        4.4.9 Numeric Properties
        4.4.10 Joining Properties
        4.4.11 Uniblock Properties
        4.4.12 East Asian Width Property
        4.4.13 East Asian Classification
        4.4.14 Script Properties
        4.4.15 SCS Constraint Properties
        4.4.16 Handling Properties
        4.4.17 Font Properties
        4.4.18 Similar and Pattern and Programming Language Properties
        4.4.19 Properties related to function and graphic characteristics
        4.4.20 Properties related to boundaries
        4.4.21 Properties related to typographic
        4.4.22 Miscellaneous properties
        4.4.23 Unihan properties
5 Blocks
6 Named Sequences
7 Normalization Corrections
8 Standardized Variants
9 Wikipedia
10 Appendix
11 The full schema
12 Examples
13 Acknowledgements
14 Bibliographies

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 ICU) 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 a "grep" for XML data.

12 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.

2.2 Namespace
2.3 Elements
2.4 Attributes
2.5 Character attributes
2.6 Ordering of elements
3 Description
4 Reference
4.1 Bana code points
4.2 Code point types
4.3 Unicode
4.4 Properties
4.4.1 Use properties
4.4.2 Name properties
4.4.3 Name Notes
4.4.4 Block
4.4.5 Additional Category
4.4.6 Combining Properties
4.4.7 Isolation Property
4.4.8 Decomposition Properties
4.4.9 Numeric Properties
4.4.10 Joining Properties
4.4.11 Uniblock Properties
4.4.12 East Asian Width Property
4.4.13 East Asian Classification
4.4.14 Script Properties
4.4.15 SCS Constraint Properties
4.4.16 Handling Properties
4.4.17 Font Properties
4.4.18 Similar and Pattern and Programming Language Properties
4.4.19 Properties related to function and graphic characteristics
4.4.20 Properties related to boundaries
4.4.21 Properties related to typographic
4.4.22 Miscellaneous properties
4.4.23 Unihan properties
5 Blocks
6 Named Sequences
7 Normalization Corrections
8 Standardized Variants
9 Wikipedia
10 Appendix
11 The full schema
12 Examples
13 Acknowledgements
14 Bibliographies
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 inheritance in CSS or in XML FO (see section 2.3 Datatypes).

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/ns/2003/ucd/1.0". Our attributes are in the empty namespace.

```xml
<namespace declaration, 1>
  default namespace ucd = "http://www.unicode.org/ns/2003/ucd/1.0"
</namespace declaration, 1>
```

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, 2>
  <xs:attribute use="http://www.w3.org/2001/XMLSchema-datatypes"

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 "&amp;#x00E8;&amp;#x0030;", that is have exactly two characters (in the infoset 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+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 (upperpanes) and code points in a sequence separated by space; for example, the decomposition of U+00E8 will be represented by the nine characters "0065 0030" in the infoset.

```xml
<datatypes for code points, 3>
  <xs:attribute use="http://www.w3.org/2001/XMLSchema-datatypes"

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 unallocated plane, where all but the last two code points are reserved and those assigned to abstract characters (PUA or not)

```xml
<set-of-code-points, 8>
  <attribute cp { single-code-point } =

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

```xml
<set-of-code-points, 8>
  <attribute cp { single-code-point } =

In elements that hold lists of child elements, such as `repertoire`, we use the properties `first-cp`, `last-cp`, and `cp` for those:

```xml
<set-of-code-points, 8>
  <attribute cp { single-code-point } =

A large number of properties are boolean. We uniformly use the values `v` and `n` for those:

```xml
<set-of-code-points, 8>
  <attribute cp { single-code-point } =

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.

```xml
<description, 5>
  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, 7>
  repertoire { code-point | group } =

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 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.

```xml
[Set of code points, 8]
  set-of-code-points =

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:

```xml
[Code points, 9]
  code-point { code-point |

When thinking about Unicode code points, it is useful to split them into four types:
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+1752 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+1749?" easier, because that data is expressed right there. On the other hand, this leads to rather large data files, and it 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 element. For example, the fragment with explicit properties:

```xml
<char cp="1740" age="3.2" na="BUHID LETTER A" gc="Mn" sc="Buhd"/>
<char cp="1741" age="3.2" na="BUHID LETTER I" gc="Lo" sc="Buhd"/>
<char cp="1752" age="3.2" na="BUHID VOWEL SIGN U" gc="Me" sc="Buhd"/>
<char cp="1820" age="3.0" na="MONGOLIAN LETTER A" gc="Mong" sc="Mong"/>
```

is equivalent to this fragment which uses a group:

```xml
<group age="3.2" go="Lo" sc="Buhd">
  <char cp="1740" na="BUHID LETTER A"/>
  <char cp="1741" na="BUHID LETTER I"/>
  <char cp="1752" na="BUHID VOWEL SIGN U"/>
  <char cp="1820" na="MONGOLIAN LETTER A" sc="Mong"/>
</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.

```xml
[groups, 10] =
  group +
  element group {
    code-point-properties & attribute na1 { character-name }?
  }
```

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 databases (some properties are no longer present in version 6.1.0).

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.

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.

```xml
[age, 11] =
  code-point-properties & attribute age {
    "2.0" | "2.1" | "3.0" | "3.1" | "3.2" | "3.3" | "5.0" | "5.1" | "5.2" | "6.0" | "6.1" | "6.2" | "6.3" | "7.0" | "8.0"
  }
```

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 [na1].

```xml
[name pattern, 12] =
  code-point-properties & attribute na1 { character-name }
```

4.4.3 Name Aliases

The Name_Alias property is represented by zero or more name-alias child elements:

```xml
[name_alias property, 14] =
  code-point-properties & element name-alias {
    attribute alias { text ? } attribute alias { text ? }
    attribute type { "abbreviation" | "alternate" | "control" | "composition" | "figurate" ? }
  }
```

4.4.4 Block

The Block property is represented by the blk attribute:

```xml
[block property, 15] =
  code-point-properties &
  attribute blk { "Aegean_Numbers" | "Alchemical" | "Arabic" | "Archaic" | "Arabic Adages"... 
```
UAX #42: Unicode Character Database in XML
http://www.unicode.org/reports/tr42/tr42-16.html
4.4.5 General Category

The general category is represented by the gc attribute.

\[ gc\] = code-point-properties & 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" }?

4.4.6 Combining Properties

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

\[ ccc\] = code-point-properties & attribute ccc { xsd:integer { minInclusive="0" maxInclusive="254" } }?

4.4.7 Bidirectionality Properties

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

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

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

The `bidi_control` property is represented by the `bidi_c` attribute.

The `bpt` and `bpb` attributes are represented by the `bpt` and `bpb` attributes respectively.

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

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

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

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

The `jg` attribute is the joining group of the character.
The Join_Control property is represented by the `join_control` attribute.

4.4.11 Linebreak Properties

The linebreak property is represented by the `linebreak` attribute.

4.4.12 East Asian Width Property

The East Asian width property is represented by the `east_asian_width` 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 # NUMBER SIGN) as a shorthand notation; this enables those attributes to be captured in groups.

The simple case mappings are recorded in the `suc`, `slc`, and `stc` attributes.

The Simple_Case_Folding and Case_Folding properties are recorded in the `scf` and `cf` attributes respectively.

The Case_Ignorable, Cased, Changes_When_Casemapped, Changes_When_Casefolded, Changes_When_Lowercased, Changes_When_NFKC_Casefolded, Changes_When_Titlecased, Changes_When_Uppercased and NKFC_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 scx attributes respectively.

```
<sc property, 36> =
  code-point-properties &=
    attribute sc { script }?
<scx property, 37> =
  code-point-properties &=
    attribute scx { list { script + } }?
```

4.4.15 ISO Comment Properties

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

```
<isc property, 37> =
  code-point-properties &=
    attribute isc { text }?
```

4.4.16 Hangul Properties

The property Hangul_Syllable_Type is represented by the hst attribute.

```
<hst property, 38> =
  code-point-properties &=
    attribute hst { "L" | "LV" | "LVT" | "T" | "V" | "NA" }?
```

The property Jamo_Short_Name is represented by the JSN attribute:

```
<jsn property, 39> =
  code-point-properties &=
    attribute JSN { xsd:string { pattern="[A-Z]{0,3}" } }?
```

4.4.17 Indic Properties

The property Indic_Syllabic_Category is represented by the InSC attribute.

```

4 of 15 4/30/2015 3:04 PM

<inSC property, 40> =
  code-point-properties &=
    attribute InSC { "Avagraha" | "Bindu" | "Brahmi_Joining_Number" | "Cantillation_Mark" | "Consonant" | "Consonant_Dead" | "Consonant_Final" | "Consonant_Head_Letter" | "Consonant_Killer" | "Consonant_Medial" | "Consonant_Placeholder" | "Consonant_Preceding_Repha" | "Consonant_Repha" | "Consonant_Succeeding_Repha" | "Gemination_Mark" | "Invisible_Stacker" | "Joiner" | "Modifying_Letter" | "Non_Joiner" | "Nukta" | "Number" | "Number_Joiner" | "Other" | "Pure_Killer" | "Register_Shifter" | "Syllable_Modifier" | "Tone_Letter" | "Tone_Mark" | "Virama" | "Visarga" | "Vowel" | "Vowel_Dependent" | "Vowel_Independent" }?
```

The property Indic_Matra_Category is represented by the InMC attribute:

```
<inMC property, 41> =
  code-point-properties &=
    attribute InMC { "Right" | "Left" | "Visual_Order_Left" | "Left_And_Right" }?
```
The property Indic_Positional_Category is represented by the `InPC` attribute:

```
<InPC property, 42> =
code-point-properties &=
attribute InPC { "Bottom"
| "Bottom_And_Right"
| "Left"
| "Left_And_Right"
| "NA"
| "Overstruck"
| "Right"
| "Top"
| "Top_And_Bottom"
| "Top_And_Bottom_And_Right"
| "Top_And_Left"
| "Top_And_Left_And_Right"
| "Top_And_Right"
}
```

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-properties &=
attribute IDS { boolean }?
```

The properties Pattern_Syntax and Pattern_White_Space are represented by corresponding attributes:

```
[pattern properties, 44] =
code-point-properties &=
attribute Pat_Syn { boolean }?
```

4.4.19 Properties related to function and graphic characteristics

The properties Dash, Hyphen, Quotation_Mark, Terminal_Punctuation, STerm, 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 and White_Space describe the function or graphic characteristic of a character, and have each a corresponding attribute.

```
[properties related to function and graphic characteristics, 45] =
code-point-properties &=
attribute Dash { 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:

```
[properties related to boundaries, 46] =
code-point-properties &=
attribute Gr_Base { boolean }?
```
4.4.21 Properties related to ideographs

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

[4.4.21 Properties related to ideographs, 47] =

code-point-properties &=
  attribute Ideo { boolean }?
code-point-properties &=
  attribute UIdeo { boolean }?
code-point-properties &=
  attribute IDSB { boolean }?
code-point-properties &=
  attribute IDST { boolean }?
code-point-properties &=
  attribute Radical { boolean }?

4.4.22 Miscellaneous properties

The properties Deprecated, Variation_Selector and Noncharacter_Code_Point have corresponding attributes:

[miscellaneous properties, 48] =

code-point-properties &=
  attribute Dep { boolean }?
code-point-properties &=
  attribute VS { boolean }?
code-point-properties &=
  attribute NChar { boolean }?

4.4.23 Unihan properties

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

[Unihan properties, 49] =

code-point-properties &= attribute kAccountingNumeric
  { xsd:string {pattern="[0-9]+"} }?
code-point-properties &= attribute kAlternateHanYu
  { text }?  #old
code-point-properties &= attribute kAlternateJEF
  { text }?  #old
code-point-properties &= attribute kAlternateKangXi
  { text }?
code-point-properties &= attribute kAlternateMorohashi
  { text }?
code-point-properties &= attribute kBigFive
  { xsd:string {pattern="[0-9A-F]{4}"} }?
code-point-properties &= attribute kCCCII
  { xsd:string {pattern="[0-9A-F]{6}"} }?
code-point-properties &= attribute kCNS1986
  { xsd:string {pattern="[12E]-[0-9A-F]{4}"} }?
code-point-properties &= attribute kCNS1992
  { xsd:string {pattern="[123]-[0-9A-F]{4}"} }?
code-point-properties &= attribute kCangjie
  { xsd:string {pattern="[A-Z]{3}"} }?
code-point-properties &= attribute kCantonese
  { list { xsd:string {pattern="[a-z]+[1-6]"} +} }
code-point-properties &= attribute kCheungBauer
  { text }?
code-point-properties &= attribute kCheungBauerIndex
  { list { xsd:string {pattern="[0-9]{3}.[0-9]{2}"} +} }
code-point-properties &= attribute kCihaiT
  { list { xsd:string {pattern="[1-9][0-9]{0,3}.[0-9]{3}"} +} }
code-point-properties &= attribute kCompatibilityVariant
  { "" | xsd:string {pattern="U+2?[0-9A-F]{4}"} }?
code-point-properties &= attribute kCowles
  { list { xsd:string {pattern="[0-9]{1,4}(.[0-9]{1,2})?"} +} }
code-point-properties &= attribute kDaeJaweon
  { xsd:string {pattern="[0-9]{4}.[0-9]{2}[0158]"} }?
code-point-properties &= attribute kDefinition
  { text }?
code-point-properties &= attribute kEACC
  { xsd:string {pattern="[0-9A-F]{6}"} }?
code-point-properties &= attribute kFenn
  { list { xsd:string {pattern="[0-9]+a?[A-KP*]"} +} }
code-point-properties &= attribute kFennIndex
  { list { xsd:string {pattern="[1-9][0-9]{0,2}.[01][0-9]"} +} }
code-point-properties &= attribute kFourCornerCode
  { list { xsd:string {pattern="[0-9]{4}(.[0-9])?"} +} }
code-point-properties &= attribute kFrequency
  { xsd:string {pattern="[1-5]"} }?
code-point-properties &= attribute kGB0
  { xsd:string {pattern="[0-9A-F]{4}"} }?
code-point-properties &= attribute kGB1
  { xsd:string {pattern="[0-9A-F]{4}"} }?
code-point-properties &= attribute kGB3
  { xsd:string {pattern="[0-9A-F]{4}"} }?
code-point-properties &= attribute kGB5
  { xsd:string {pattern="[0-9A-F]{4}"} }?
code-point-properties &= attribute kGB7
  { xsd:string {pattern="[0-9A-F]{4}"} }?
code-point-properties &= attribute kGB8
  { xsd:string {pattern="[0-9]{4}"} }?
code-point-properties &= attribute kGradeLevel
  { text }?
5 Blocks

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

Similarly, the provisional-named-sequences child of the ucd describes the provisional named sequences.

[named sequences, \[51\]]=
   sub content to
       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 normalization, and the version of Unicode in which the correction was made.

[normalization corrections, \[52\]]=
   sub content to
       element normalization-corrections {
           element normalization-correction {
               attribute cp [ 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-variant 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 variant sequence, the description of the desired appearance, and the shaping environment under which the appearance is different.

[standardized variants, \[53\]]=
   sub content to
       element standardized-variants {
           element standardized-variant {
               attribute cps [ two-code-points ],
               attribute name [ text ],
               attribute when [ text ] + |
           }
       }

9 CJK Radicals

The cjk-radical 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, \[54\]]=
   sub content to
       element cjk-radicals {
           element cjk-radical {
               attribute number [ xsd:string {pattern="[0-9]{1,3}'?"}],
               attribute when [ text ],
               attribute desc [ text ],
               attribute cps [ two-code-points ],
               attribute version [ text ] + |
           }
       }

10 Emoji sources

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

[emoji sources, \[55\]]=
   jie-code-point = xsd:string { pattern = "[0-9A-F]{4}" },

[emoji sources, \[56\]]=
   sub content to
       element emoji-sources {
           element emoji-source {
               attribute ideograph [ single-code-point ],
               attribute radical [ single-code-point ],
               attribute number [ xsd:string { pattern = "[0-9]{1,3}'?" } ],
               attribute when [ text ],
               attribute desc [ text ],
               attribute cps [ two-code-points ],
               attribute version [ text ] + |
           }
       }

11 The full schema

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

[UCD RelaxNG schema, \[57\]]=
   namespace declaration: 1
   [datatypes: 2, 3, 15, 55]
   [schema start: 4]
   [boolean type: 5]
   [elements: 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]

12 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/ns/2003/ucd/1.0>
   <element cp="001F" age="1.1" na="&lt;control&gt;" bc="ON" dm="1100 1161" ea="Na" lb="SP" sc="Hang" kRSUnicode="4.3" kIRGKangXi="4.9" kIRG_TSource="5-214E"/>
   <element cp="0020" age="1.1" na="SPACE" bc="WS" dm="1100 1161" ea="Na" lb="SP" sc="Hang" kRSUnicode="4.3" kIRGKangXi="4.9" kIRG_TSource="5-214E"/>
</ucd>
<char cp="1752" na="BUHID VOWEL SIGN I" gc="Mn"/>
<char cp="1820" age="3.0" na="MONGOLIAN LETTER A" sc="Mong"/>
<group>
</group>

Acknowledgments

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

Modifications

This section indicates the changes introduced by each revision.

Revision 16

- New value for the <pattern> attribute: x.
- New values for the <character> attributes: A, B, C, D.
- New values for the <script> attributes: Alf, Alb, Sml, Pml, Mrg.
- New value for the <name> attribute: consonant_killer, syllable_modifier.
- New code point attributes: csp, ksa.
- New patterns for the <kro_resource> attribute: sqc, mcs, jg.

Revision 15

- New value for the <script> attribute: x.
- New values for the <name> attributes.
- New values for the <script> attribute.
- New values for the <name> attribute.
- New values for the <kro_resource> attribute.
- New values for the <kro_resource> attribute.

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

Revision 13

- New value for the <script> attribute: x.
- New values for the <name> attributes.
- New values for the <script> attribute.
- New values for the <name> attribute.
- New values for the <kro_resource> attribute.
- New patterns for the <kro_resource> attribute.
- New patterns for the <kro_resource> attribute.
- Updated the patterns for <kro_resource> and <kro_resource> (for Unicode 6.3).
- 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 <script> attribute: x.
- New value for the <name> attributes.
- New value for the <script> attributes.
- Updated the patterns for <kro_resource> and <kro_resource> (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 <script> attributes, for the Block property.
- Introduced the <script> attribute, for the ScriptExtensions property.
- Introduced the <script> element, for the Name_Alias property.
- New value for the <script> attribute: x.
- New values for the <script> attributes: x.
- New values for the <script> attributes: x.
- New values for the <script> attributes: x.
- New values for the <script> attributes: x.
- New values for the <script> attributes: x.
- The value of the <script> attribute must now be either # or one-or-more-code-points.
- For the <script> attribute, the absence of a numeric value is now represented by #x rather than by the empty string.
- The values of the <script> are now restricted to 0..254, instead of 0..255.
- Updated the patterns for <script> and <script> (for Unicode 6.3).

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

Revision 7

- New value for the <script> attribute: x.
- New value for the <script> attribute: x.
- New value for the <script> attributes: x.
- New value for the <script> attributes.
- Updated the patterns for <kro_resource> and <kro_resource>.
- Added the <script> and <script> elements.

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

Revision 5

- Changed the type of block/@first-cp, block/@last-cp and normalization_corrections/@cp from text to single-code-point.
* Changed the type of `named-sequence/@cps`, `provisional-named-sequences/@cps`, `normalization-correction/@old` and `normalization-correction/@new` from `text` to `one-or-more-code-points`.

* Changed the type of `standardized-variants/@cps` from `text` to `two-code-points`.

* New values for the `jg` attribute: Farsi_Yeh and Nya.

* New value for the `age` attribute: 5.2.

* New values for the `sc` attribute: Lana, Tavt, Avst, Egyp, Samr, Lisu, Bamu, Java, Mtei, Armi, Sarb, Prti, Phli, Kthi.

* New code point attributes: CI, Cased, CWCF, CWCM, CWL, CWKCF, CWT, CWU, NFKC_CF.

* New attributes: kHanyuPinyin and kIRG_MSource.

* New element: `cjk-radicals`.

* Updated the patterns for `kIRG_Gsource`, `kIRG_JSource`, `kIRG_KPSource`, `kIRG_KSource`, `kIRG_TSource`, `kIRG_VSource`, `kHanyuPinlu`, `kMandarin`, `kSemanticVariant`, `kSpecializedSemanticVariant`, `kVietnamese`, `kVariant`.

* 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 5.1.0.

* For optional elements which acts as collections, such as `repertoire` and `named-sequences`, impose that there be at least one elements in the collection.

* Remove the constraint that the value `jg` is limited when `jt` has certain values; similarly for `bmg`, `Bidi_M`, and `nv`, `nt`.

* Value `NL` added to the `WB` attribute (for Unicode 5.1).

* Value `PP` added to the `GCB` attribute (for Unicode 5.1).

* Values `CR`, `Extend`, `LF`, `MB` added to the `WB` attribute (for Unicode 5.1).

* Values `CR`, `EX`, `LF`, `SC` added to the `SB` 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).

* Values `Cari`, `Cham`, `Kali`, `Lepc`, `Lyci`, `Lydi`, `Olck`, `Rjng`, `Saur`, `Sund` and `Vai` added to the `sc` attribute (for Unicode 5.0).

* `jamo` attribute renamed to `JSN`.

* `sfc` attribute renamed to `scf`.

* Attribute `kXHC1983` added (for Unicode 5.1.0).

* Pattern for attribute `kIRG_USource` extended (for Unicode 5.1.0).

* Element `provisional-named-sequences` added (for Unicode 5.0).

**Revision 2**

* Promoted to Draft UAX.

* Changed the title from "An XML representation of the UCD"

* Value `1` added to the `age` attribute (for Unicode 5.1).

* Value `pp` added to the `gcb` attribute (for Unicode 5.1).

* Values `kn`, `kx`, `kr`, `kt` added to the `sc` attribute (for Unicode 5.1).

* Value `Kushashtri_Yeh_Vaishnavi` added to the `jg` attribute (for Unicode 5.1).

* Value `Kufi_Majuscula` added to the `jg` attribute (for Unicode 2.x).

* Values `Kufi`, `Chal`, `Kali`, `Kori`, `Bhar`, `Khar`, `Khoj`, `Kharo`, `Khar` and `vaishnavi` added to the `sc` attribute (for Unicode 5.0).

**Revision 1**

* First working draft.

---

Copyright © 2005-2015 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 and consequential damages in connection with or arising out of the use of the information or programs contained or accompanying this technical report. The Unicode Terms of Use apply.

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