Unicode Character Mapping Formats

UTC/1999-034

Draft, 10-25 MED

This is a proposal for an XML format for exchanging character encoding specifications. The goal is to supply complete information, so that solely on the basis of the information in the file an implementation could perform accurate mappings between Unicode and the given character encoding.

Background

Illegal and Unassigned Codes

Client software may need to distinguish the different types of mismatches that can occur when transcoding data to and from Unicode. These fall into the following categories:

1. The sequence is *unassigned* (aka undefined).
   For example,
   - 0xA3BF is unassigned in CP950
   - 0x0EDD is unassigned in Unicode, V3.0
2. The sequence is *incomplete* (and thereby illegal):
   For example,
   - 0xA3 is incomplete in CP950.
     - Unless followed by another byte of the right form, it is illegal.
   - 0xD800 is incomplete in Unicode.
     - Unless followed by another value of the right form, it is illegal.
   - 0xDC00 is incomplete in Unicode.
     - Unless preceded by another value of the right form, it is illegal.
3. The sequence is simply *illegal.*
   For example,
   - 0xFF is illegal in CP950

Unassigned characters are treated as a single code point: for example, 0xA3BF is treated as a single code point when mapping into Unicode from CP950. The actual conversion routines will typically handle an unassigned value in a variety of ways (depending on the parameters passed in), such as:

- stop or throw an exception
  - in particular, this is commonly used by higher level character encodings, such as ISO 2022 conversions, to know when to stop converting into one encoding and pick another to convert to.
- map it to a substitution character
  - such as the Unicode U+FFFD REPLACEMENT CHARACTER
- represent it by a hex escape sequence
  - for example, when mapping from U+1234 to other code pages, it can be represented by "%12%34" in URLs, "ሴ" in XML or HTML, "\u1234" in Java or C++, or "\x{1234}" in Perl.

Note: there is an important difference between the case where a sequence represents a real REPLACEMENT CHARACTER in a legacy encoding, as opposed to just being unassigned, and thereby sometimes being mapped to REPLACEMENT CHARACTER for that reason.

Illegal values represent some corruption of the data stream. Conversion routines may be directed to handle this in a different way than by replacement characters. For example, a routine might map unassigned characters to a substitution character, but throw an exception on illegal values.

In some cases, users have the option of using fallback characters, where an character that is not represented in the target code page is given a "best fit" mapping. For example, an encoding might not have curly quotes; the generic

quotes can be used as a fallback.

Completeness

It is important that a mapping file be a complete description. From the data in the file, it should be possible to tell for any sequence of bytes whether that sequence is assigned, unassigned, incomplete, or illegal. It should also be possible to tell if characters need to be rearranged to be in Unicode standard order (visual order, combining marks after base forms).

- Unless otherwise indicated in the data file, any sequences of bytes that are not mentioned are assumed to be unassigned.
- All control values (C0, C1) should be explicitly mapped.
- All private use (e.g. user defined) characters should be explicitly mapped, either to the private use zone in Unicode, or to the correct characters outside of that zone.
- Only a real replacement character should be mapped to REPLACEMENT CHAR; unassigned characters should not be mapped to it. Similarly, when mapping back from Unicode, only the REPLACEMENT CHAR should map to SUB or other legacy equivalent.
- Incomplete and illegal sequences should be indicated.
- All fallback mappings must be clearly indicated. This is especially important for modern software that guarantees round-tripping into and out of Unicode. When emitting XML, for example, fallbacks should not be used; instead, escapes are used that preserve data integrity.

XML Format

The file starts with the following lines. The encoding can be any valid encoding: only the ASCII repertoire of characters is required, but comments may be in other character encodings..

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE characterMapping SYSTEM "CharacterMapping.dtd">
```

Header

A mapping file begins with a comment header. Here is an (artificial) example:

```
<characterMapping
 name="CP938"
 description="Sun variant of CP942 for Japanese"
 unicodeVersion="3.0"
 tableVersion="2"
 contact="mark@unicode.org"
 bidiOrder="logical"
 combiningOrder="after"
>
```

The **characterMapping** element is the root. It contains a number of required attributes:

name is the IANA name. If none exists, use "X-name". The **name** should be unique; if two mapping tables differ in any mapping, in specification of illegal characters, in their bidi ordering, in their combining character ordering, etc. then they must have a different name (or different version: see below).

description should be complete enough to describe the code page, and distinguish it form other similar code pages. It should include a rough characterization of the target locale, such as "for Japanese" or "for Western Europe", or "for Greek".

unicodeVersion is the earliest version of the Unicode standard that contains all of the characters mapped to. That is, most of the ISO 8859 series will use Unicode 2.0; the new ones with Euro will use Unicode 2.1.

tableVersion is the version of the data. Any time the data is modified, the value must be increased. If only additions are made, then the same name can be retained; if not, then a new name must be used. Additions change mappings from "unassigned" to "assigned". Any change in the validity of character sequences requires a new name.

contact is the person to contact in case errors are found in the data.

bidiOrder specifies whether the character encoding is to be interpreted in visual or logical order. Unicode is strictly logical order. Application of the Unicode Bidirectional Algorithm is required to map to a visual-order character encoding; application of a reverse bidirectional algorithm is required to map back to Unicode. The default value for this attribute is "logical". It is only relevant for character encodings for the Middle East (Arabic and Hebrew).

combiningOrder specifies the order of combining marks. Some character encodings, typically those for bibliographic use, store combining marks before base characters. Unicode stores them uniformly after base characters. The default value for this attribute is "after". This is only relevant for character encodings with combining marks.

History

```
<history>
 <modified version="2" date="1999-09-25">
  Added Euro.
 </modified>
 <modified version="1" date="1997-01-01">
  Made out of whole cloth for illustration.
 </modified>
</history>

<notes>
 No notes here.
</notes>
```

The **history** element provide information about the changes to the file, coordinated with the version. The latest version should be first. This is a required field. The subelements and attributes should be clear from their names.

The **notes** provide an opportunity to document special features of the mapping.

Naming Information

```
<aliases>
  <!--List of aliases, such as IANA names-->
  <n n="MS983"/>
  <n n="MacSunJIS"/>
 </aliases>
 <displayNames>
  <!--List of display names for this encoding in different langauges-->
  <d l="en" n="Sun Chinese"/>
  <d l="fr_fr" n="Chinois solar"/>
 </displayNames>
```

The **aliases** element provides a list of possible aliases for this code page. It is optional. The names may not be unique, because of the historic behind the development of names. The **n** attribute is used to supply the name.

The **displayNames** element is optional, but strongly recommended. It provide user-level names that can be presented in menus, such as in Netscape Navigator *View>Character Set* or the Microsoft Internet Explorer *View>Encoding*. The individual names are supplied with the **d** elements. The **l** attribute supplies the locale in the format "<ISO 2-letter language> ('_' <ISO 2-letter country> ('_' variant)*)?". The **n** attribute supplies the name. Both attributes are required.

> **Note:** short attribute and element names are used just to conserve space where there may be a large number of items, or for consistency across other elements that may have a large number of items.

Delta Mappings

```
<import source="ftp://ftp.unicode.org/Public/MAPPINGS/VENDORS/MICSFT/CP852.XML"/>
```

It is possible to supply just the differences between one table and a base table. This is done with the **import** element, which is optional. If this is used, then any further data simply overrides the data in the base table. The value of the **source** attribute is a valid URL pointing to a valid character encoding table.

Validity Specification

As discussed above, it is important to be able to distinguish when characters are unassigned vs. when they are invalid. Valid and invalid sequences are specified by the **validity** element.

```
<validity>
  <!--Validity specification for SJIS-->
  <illegal s="FD" e="FF"/>

  <legal s="81" e="9F" next="second" />
  <legal s="E0" e="FC" next="second" />

  <legal type="second" s="40" e="7E"/>
  <legal type="second" s="80" e="FC"/>

 </validity>
<validity>
```

The subelement are **legal** or **illegal**. Their attributes are:

- **type** the type of the given bytes; the default = **"start"**.
- **s** the start of the byte range
- **e** the end of the byte range; there is no default.
    - If it is missing, it is interpreted as being the same as **s** (thus a range with one single value).
- **next** the type that the following bytes are required to be in; the default = **"none"**, which indicates completion

If we look at the above table, the first line tells us that the single bytes FD through FF are illegal. The next two lines say that the bytes in the ranges 81 through 9F and E0 through FC are legal, if they are followed by a byte of **type="second"**. More detailed samples for a complex validity specifications are given in Samples.

If any bytes are not explicitly set for **type="start"**, then they are assumed to be legal with **next="done"**. Thus most single-byte encodings do not need validity elements. Any string can be used for the value of **type** or **next**, as long as it is not subject to an error condition.

Error Conditions

- Two lines conflict if they assign the same byte to a different **type**, or the same byte to a different **next**, or if one line makes the byte legal and the other makes it illegal.
- In the case of conflicts, if one of the conflicting lines is from an **import** then the byte ranges are adjusted to exclude the conflicting bytes (possibly generating multiple lines). *Otherwise the file is invalid.*
- If there is a **type** value with no matching **next** value in another line, the line is incomplete.
- If there is a **next** value with no matching **type** value in another line, the line is incomplete.
- If an incomplete line is from an import then it is disregarded. *Otherwise the file is invalid.*

Assignments

The last part of the table provides the assignments of byte sequences to Unicode characters. Here is an example:

```
<assignments>
```

```
<!--Unassignments-->
<ua b="AA"/>
<ua b="AB"/>

<!--Fallbacks-->
<f b="22" u="201C" n="LEFT DOUBLE QUOTATION MARK"/>
<f b="22" u="201D" n="RIGHT DOUBLE QUOTATION MARK"/>

<!--Main mappings-->

<!--Map ASCII to the same range-->
<ar s="00" e="7E" u="0000"/>

<!--Map user-defined area to private use-->
<ar s="F040" e="F07E" u="E000"/>

<!--Map other characters specifically-->
<a b="A1" u="FF61" n="HALFWIDTH IDEOGRAPHIC FULL STOP"/>
<a b="A2" u="FF62" n="HALFWIDTH LEFT CORNER BRACKET"/>
<a b="8156" u="3003" n="DITTO MARK"/>
<a b="8157" u="4EDD"/>
<a b="8158" u="3005" n="IDEOGRAPHIC ITERATION MARK"/>
<a b="8159" u="3006" n="IDEOGRAPHIC CLOSING MARK"/>
<a b="815A" u="3007" n="IDEOGRAPHIC NUMBER ZERO"/>
<a b="815B" u="30FC" n="KATAKANA-HIRAGANA PROLONGED SOUND MARK"/>
<a b="815C" u="2015" n="HORIZONTAL BAR"/>

</assignments>
```

In all of the elements, the attributes have the following meanings:

- **b** a sequence of bytes. Always 2 hex digits, unsigned. Multiple values should be separated by commas, but the commas can also be omitted.
- **u** a sequence of Unicode code points. One or more hex digits; unsigned. Multiple values must be separated by commas.
- **n** the Unicode code point name. Optional, but useful for reading the file and performing consistency checks. In the above example it is omitted for the CJK ideograph since the name adds little information.
- **x** alternative mapping. The value is a label on assignment lines used to add alternate mappings to the same file. Multiple labels can be attached to the same line, using commas between them. The choice of names is arbirtrary, except in so far as they might be used in an API to specify a variant of the character encoding. Examples are:
  - **"path"** indicates that the mapping is for pathnames.
    - For example, maps `5C` to `005C` "\" instead of `00A5` "¥".
  - **"graphics1"** indicates that control bytes in the source set are interpreted as graphic characters for old PC sets.
    - For example, maps `10` to `25BA BLACK RIGHT-POINTING POINTER`.
  - **"graphics2"** indicates that control bytes in the source set, except for CR and LF are interpreted as graphic characters for old PC sets.
    - For example, maps `10` to `25BA BLACK RIGHT-POINTING POINTER`.
  - **"cdra"** indicates that control bytes in the source set are interpreted according to IBM CDRA alternate mappings.
    - For example, maps `7F` to `1A SUB`.

The elements are:

- **ua** specifies that a byte sequence is unmapped. Normally this is not needed, since any sequence that is not assigned is assumed to be unassigned. The one exception is if there is an import statement, where it may be necessary to override specific mappings.
- **a** specifies an mapping from byte sequences to Unicode and back.
- **f** is the same as **a**, except that it specifies that a mapping is a fallback. This means that the mapping can be

used if fallbacks are turned on by API options.

- **ar** specifies that a range of byte sequences map to a range of Unicode values. This is a shorthand for a series of **a** mappings--especially useful for private use zone assignments. It uses attributes **s** for the start of the sequence; **e** for the end of the sequence, and **u** for the starting Unicode value. All byte sequences between **s** and **e** are mapped to the range from **u** to (**u+e-s**).
  - This element is present purely to reduce the number of lines in the file; in every respect (such as in error conditions), it should be treated as if it were simply an expanded series of **a** elements.

Error Conditions

- All sequences of bytes must be valid according to the validity specification. *Otherwise the file is invalid.*
- All sequences of bytes must map to legal Unicode code points. *Otherwise the file is invalid.*
  - The illegal code points are: out-of-range values (greater than `10FFFF`), surrogate values (`D800` to `DF00`), and excluded values (of the form `xxFFFF` or `xxFFFE`).
- A fallback line must have the same byte sequence as in some assignment line, *or* the same Unicode sequence as in some assignment line (not necessarily the same assignment line), *but* not both. *Otherwise the file is invalid.*
- Two assignment lines conflict if they have either the same byte sequence or the same Unicode sequence.
- An unassignment line conflicts if it has the same byte sequence as either an assignment line.
- In the case of conflicts, if one of the conflicting lines is from an **import** then it is disregarded. If the conflicting lines have different **x** (alternate) values, that does not cause a problem. *Otherwise the file is invalid.*

Samples

Full Sample

A full example is on CharacterMapping.xml. It is not a real one since it tries to show all of the features in one file, whereas in real life only a subset would be used. You can view it directly with Internet Explorer, which will interpret the XML—however, it doesn't appear to fetch the dtd properly, so you'll need to access them without using a link.

UTF-8 Sample

Here is a simple version of the UTF-8 validity specification, with the shortest-form bounds checking and exact limit bounds checking omitted. While in practice a mapping file is never required for UTF-8 since it is algorithmically derived, it is instructive to see the use of the validity element as a complicated example. As a reminder, first here are the valid ranges for UTF-8:

Unicode Code Points	UTF-8 Code Units
00	00
7F	7F
80	C2 80
7FF	DF BF
800	E0 A0 80
FFFF	EF BF BF
010000	F0 90 80 80
10FFFF	F4 8F BF BF

```
<illegal s="80" e = "BF"/>
<illegal s="F5" e = "FF"/>
```

```
<!-- 2 byte form -->
<legal s="C0" e="DF" next="final" />
<legal type="final" s="80" e="BF" />

<!-- 3 byte form -->
<legal s="DF" e="EF" next="prefinal" />
<legal type="prefinal" s="80" e="BF" next="final" />

<!-- 4 byte form -->
<legal s="F0" e="F4" next="preprefinal" />
<legal type="preprefinal" s="80" e="BF" next="prefinal" />
```

UTF-8 Full Sample

The following provides the full validity specification for UTF-8.

```
<validity>
 <!--Validity specification for SJIS-->
 <illegal s="80" e = "C1"/>
 <illegal s="F5" e = "FF"/>

 <!-- 2 byte form -->
 <legal s="C2" e="DF" next="final" />
 <legal type="final" s="80" e="BF"/>

 <!-- 3 byte form; Low range is special-->
 <legal s="E0"          next="prefinalLow" />
 <legal type="prefinalLow" s="A0" e="BF" next="final" />

 <!-- 3 byte form, Normal -->
 <legal s="E1" e="EF" next="prefinal"  />
 <legal type="prefinal"  s="80" e="BF" next="final" />

 <!-- 4 byte form, Low range is special -->
 <legal s="F0"          next="preprefinalLow" />
 <legal type="preprefinalLow" s="90" e="BF" next="prefinal"/>

 <!-- 4 byte form, Normal -->
 <legal s="F1" e="F3" next="preprefinal"    />
 <legal type="preprefinal" s="80" e="BF" next="prefinal" />

 <!-- 4 byte form, High range isspecial-->
 <legal s="F4"          next="preprefinalHigh" />
 <legal type="preprefinalHigh" s="80" e="8F" next="prefinal"/>

</validity>
```

DTD

A draft DTD is on CharacterMapping.dtd.