# A Specification for CDL

## *Character Description Language*

Source: Tom Bishop <tbishop@wenlin.com> and Richard Cook <rscook@unicode.org>

Status: Expert Contribution

Date: 2003-10-31 (slightly corrected 2003-12-21)

Action: For consideration by UTC and IRG

Contents

## INTRODUCTION

Character Description Language (CDL) is for accurately describing and displaying the forms of all Han (CJKV) characters. This document, which is the first public specification of CDL, presents the key features and syntax of the language, and discusses some of its applications, especially to character encoding standards work. We propose adoption of CDL as a data management tool for ensuring accuracy and long-term stability in the public character encoding process.

The acute need for CDL is predicated upon the fact that the set of Han characters is truly open-ended, rather like the set of English words. Historical and idiosyncratic spelling differences present a vast quantity of data, and a large number of forms not easily related to currently encoded forms. Witness the tens of thousands of characters being evaluated by the IRG for inclusion in CJK Unified Ideographs Extension C1.

CDL is based on Unicode, XML, and a few well-known characteristics of Han characters:

- Most characters are formed by combining two or more simpler characters or components and fitting them into a square.
- Basic characters or components are composed of strokes, which are classified into distinct types in accordance with modern orthographic conventions.
- Identification of stroke types underlies consistent counting of strokes.
- Stroke types, stroke counts, and component analysis are essential to the learning process, character recognition, indexing, and comparison of variant forms.

A set of less than fifty stroke types is sufficient for the construction of practically all characters in a modern printed style, as demonstrated by the existence of CDL descriptions for over 40,000 characters, including all BMP Han characters and over 12,000 in Extension B.

## THE CDL FONT DATABASE

A CDL description of a character encodes an analysis of the character into its constituent components and/or strokes, and simultaneously provides instructions for displaying the character. A collection of CDL descriptions can therefore serve as both a database and a font.

When CDL is used as a font format, a software interpreter converts the descriptions into glyphs in real time. For Han characters, CDL has some advantages over conventional font formats. It is much smaller — only about 12 bytes per character, on average, when compressed. It is a kind of “meta-font” in the sense that it has variable parameters so that the same descriptions can produce different styles of glyphs.[1] New glyphs can be added to the font relatively quickly and easily. Consistency between the forms of related characters is easier to ensure as a consequence of the sharing of components.

As a database language, CDL encodes essential information for categorizing, indexing, learning, and recognizing Chinese characters. This information includes stroke count, stroke types, stroke order, component analysis, radicals and residual strokes, and coordinates of strokes and components. While some of this information is, or could be, stored in an ordinary database, CDL is better for enforcing consistency. For example, the stroke count of a character is calculated algorithmically from actual CDL instructions for writing the character stroke-by-stroke; it is not merely a personal impression, or gathered from one of various dictionaries that may not be mutually consistent (or even individually self-consistent) in counting the strokes of a particular component.

## EXAMPLES

Here is a description for 行, as a combination of the components 彳 and 亍:

```
<cdl char="行">
     <comp char="彳" points="0,0 40,128" />
     <comp char="亍" points="60,12 128,128" />
</cdl>
```

Positions are given as points with two-dimensional coordinates. The square enclosing the entire character has (x, y) coordinates ranging from (0, 0) for the top left corner, to (128, 128) for the bottom right corner.[2] The numbers after 彳 describe its bounding rectangle on the left side of 行: (0, 0) is its top left corner, and (40, 128) is its bottom right corner.[3] Similarly, a rectangle is given for 亍 on the right side of 行.

In order for the above CDL description to be carried out as a set of instructions (e.g., for displaying the character or counting its strokes), it is necessary for the interpreter to refer to the separate descriptions of the components, 彳 and 亍, as sequences of particular stroke types with specific coordinates. Here is a description[4] for 彳:

```
<cdl char="彳">
     <stroke type="p" points="107,0 10,46" />
     <stroke type="p" points="128,38 0,83" />
     <stroke type="s" points="86,70 86,128" />
</cdl>
```

There are three strokes in 彳. The first two (from top to bottom) are both type ‘p’, which stands for 撇 *piě*, a curved stroke falling to the left. The third stroke is type ‘s’, which stands for 竖 *shù*, a vertical falling stroke. For each of these simple stroke types, only two points are needed. For example, the first stroke starts at (107, 0) and ends at (10, 46).

Some descriptions combine components and strokes. Here, the character 太 is described as a combination of the component 大 (which itself is a character, and should have its own description), and a stroke of type ‘d’ (点 *diǎn*, dot):

```
<cdl char="太">
    <comp char="大" points="0,0 128,128" />
    <stroke type="d" points="45,104 66,128" />
</cdl>
```

## LANGUAGE DETAILS

CDL is an XML application, which means that it conforms to a widely-used standard syntax (usage of angle brackets < >, et cetera). We have already introduced most of the elements of the language: each description is contained in a `cdl` element, which can contain any number of `comp` (component) and/or `stroke` elements. There is another element, `cdl-list`, for enclosing a list (or file, font, or database) of descriptions. The only CDL elements currently defined are these four: `cdl-list`, `cdl`, `comp`, and `stroke`.

Both the `cdl` and `comp` elements have `char` (character) attributes. The value of the `char` attribute is simply a character: typically a Han character, which might be encoded with UTF-8 or any other XML-supported encoding. Any character can, in principle, be used as a component.[5]

The `stroke` element has a `type` attribute, whose value is one of less than fifty names of stroke types that are defined for CDL. This article has already introduced ‘p’ for 撇 *piě* and a few others. One of the most complex stroke types is ‘hzzzg’, which stands for 横折折折钩 *héng-zhé-zhé-zhé-gōu*, and is exemplified by the character 乃. It has six reference points, including four points of inflection between the starting and ending points. The essential features of each stroke type are: its name; the number of reference points it uses; and the directions and curvatures between the reference points. The complete set of CDL stroke types is documented in another article.[6, 7]

There is a form of recursion implied by CDL. For example, a description of 龍 may refer (with a `comp` tag) to a description of 立, which in turn may refer (with another `comp` tag) to a description of 亠, which describes two individual strokes. A CDL interpreter will therefore typically process components within components within components, using recursive algorithms (and scaling coordinates according to bounding rectangles). Recursion stops when `stroke` elements are reached.[8]

Any CDL description that uses `comp` elements can be transformed automatically into a description that uses only `stroke` elements. For example, 行 is described as a sequence of two components 彳 and 亍, each of which is in turn described as a sequence of three strokes. Alternatively, 行 could be described directly as a sequence of six strokes. A straightforward recursive algorithm can transform the component description into the “strokes-only” description. The reverse transformation might be more difficult. Component descriptions are more generally useful as well as more concise.[9]

## EXTENDING THE PRECISION AND SCOPE OF CHARACTER SETS

Compared with even the largest standard character set, CDL provides more precision: the ability to distinguish between unified variants. It also provides wider scope: a potentially infinite number of Han characters.

CDL can describe and display particular variants of characters that are "unified" (treated as equivalent) in standard character sets. For example, in Unicode the forms 者 (eight strokes) and 者 (nine strokes) are both `U+8005`, but can be made distinct using CDL.[10]

CDL can also be used for describing and displaying characters that are not in any standard character set. Some such characters might simply not have been encoded yet; some might be new; some might have extremely limited and special usages, and therefore might not even be suitable for inclusion in a standard character set.

The CDL instructions for displaying a character can be composed whenever the need arises (preferably using a graphical user interface), and included directly in a document using XML syntax. Of course, the program displaying the text needs to have the capability of interpreting the language, possibly by means of a "plug-in" or "helper" application; people reading the text simply see the resulting image of the character, not the CDL tags.

## MANAGING DATA FOR CHARACTER SET STANDARDIZATION

By simultaneously producing both a (meta-)font and a database, CDL can enable standards organizations to publish representative glyphs and stroke counts (etc.), that are consistent with each other. Furthermore, the language can facilitate systematic treatment of the complex and difficult problems of unification and variation. Currently, such systematic treatment is held back by the absence of an intermediate representation of character forms, between abstract "characters" and concrete "images" (or particular written/printed instances) of characters. Each Unicode codepoint represents an abstract character, which corresponds to a potentially infinite number of graphic images. Graphic images are useful as examples of characters, but it is practically impossible, in general, for an algorithm to determine the stroke count of an image, or to measure the degree of similarity between two images according to the principles of Han unification. Consequently, with over 70,000 Han characters already encoded, it has become difficult to determine whether a given glyph corresponds to any of the characters that have already been encoded. Really there are two difficulties: first, to find all the likely candidates for codepoints that might correspond to the glyph in question; second, to decide for each of those codepoints whether the glyph belongs to that codepoint's implicit equivalence class according to the unification principles.

CDL can help resolve both of the difficulties just mentioned. A CDL database could be built for all encoded Han characters. Each character could potentially have multiple CDL descriptions, corresponding to variants[11] that have been unified. Then, when confronted with a glyph, if one were uncertain whether it was already encoded, one could construct a CDL description for it, and run a program to compare that description with those already in the database, to find the closest matches. (Several comparison algorithms could be applied for the same character, some based on strokes, some based on components.) Of course, a perfect match would be unlikely, but trivial differences in coordinates or stroke order would easily be recognized as falling within the scope of unification. Less trivial differences would still require judgment by experts, but CDL would make it far easier for the experts to apply the unification rules consistently. For example, all the characters containing a given component could be examined to discover any precedent for unifying two variants of that component. If the decision were made to unify the new glyph with an already encoded character, in spite of some difference, then the CDL for the new glyph could be added to the database as a variant, thus providing a precedent, making the unification rules more explicit, and facilitating future usage of the database.[12, 13]

## ORIGIN AND CURRENT STATUS

CDL was originally designed and implemented (in the C programming language) by one of the authors, and is an integral part of *Wenlin Software for Learning Chinese,* published by Wenlin Institute, Inc. Its original application was Wenlin's *Stroking Box*, which illustrates for learners how to write a character stroke-by-stroke in slow motion. It turned out to be fast enough for use as a general-purpose scalable font. It also provides stroke-count and stroke-type information, and is even applied to handwriting recognition. However, the CDL language itself is hidden from the user, and only the resulting stroked characters are visible. Wenlin actually uses a compressed binary format, which is equivalent to the XML format, but very compact and fast for machine processing. Wenlin's CDL was used to create printed radical and stroke-order indexes for 9,638 characters in the *ABC Chinese-English Comprehensive Dictionary,* published in 2003 by University of Hawaii Press (ISBN 0-8248-2766-X).

Currently (October 2003) over 40,000 characters have CDL descriptions, including all the Han characters in Unicode 3.0 (with Extension A) and many more that are in Unicode 4.0 (Extension B). These descriptions were made by the authors.

## CONCLUSION

Experience has shown CDL to be a useful language for systematic treatment of Han characters. While it undoubtedly still has room for improvement, the authors have become convinced (with the encouragement of several members of the Unicode Technical Committee) that it should be made public for the benefit of the international community, especially standards organizations. Comments, questions, and suggestions are welcome.

## REFERENCES

The latest revision of this article, and other information about CDL (including the list of stroke types and a DTD[14]), may be found at http://www.wenlin.com/cdl.

The Unicode Consortium website is http://www.unicode.org. The International Standards Organization (ISO) website is http://www.iso.org. The Ideographic Rapporteur Group (IRG) website is http://www.cse.cuhk.edu.hk/~irg.

XML (Extensible Mark-up Language) is described at http://www.xml.org and http://www.w3.org/XML.

## NOTES

1. The concept of a "meta-font" originated with the METAFONT language (documented in The METAFONTbook by Donald Knuth, 1986, ISBN 0-201-13445-4). Although CDL is not closely related to METAFONT, there is a procedure for converting CDL into METAFONT, but currently only at a low-level in which the glyph outline is exactly specified. A similar procedure exists for converting CDL into the PostScript language (PostScript is a trademark of Adobe; see http://www.adobe.com).

2. All coordinates are decimal integers in the range 0 through 128. CDL could easily be extended to allow floating-point numbers and/or different ranges of coordinates. However, the use of small integers and a power of two like 128 leads to compact storage and fast rendering even on slow machines, and has been found to give plenty of precision. More sophisticated versions of the language should allow symbolic variable names, and even algebraic expressions, to stand for coordinates. It should be possible to convert automatically from such "higher level" versions of CDL into the basic "low-level" version of CDL that uses only numerical coordinates. For some purposes, it is likely to be convenient to describe component and stroke positions with less precision, with rough indications such as top, left, top-left, middle, etc.; there should be utilities to support conversion back and forth between such rough indications and precise coordinates.

3. A clarification is needed regarding coordinates and bounding rectangles. In general, the reference points for a stroke are inside the stroke, roughly at the center of the tip of an imaginary brush. For a thick stroke, the fat tip of the brush may extend the radius of "ink" a considerable distance in all directions from the reference point. The precise flow of ink depends on the particular font style, and the same CDL description could be displayed differently by different interpreters (or by the same interpreter, given a different set of preferences). What we mean by the bounding rectangle of a component is based only on the reference points; "extra ink" might extend about half the thickness of a stroke in any direction beyond that rectangle.

4. The description for 亻 has been simplified slightly to make it easier to understand. A better description might use the optional `points` attribute of the `cdl` tag. Rather than simply `<cdl char="亻">`, the opening tag might be `<cdl char="亻" points="24,0 104,128">`. This means that when 亻 is displayed by itself, it does not take up the entire square, but instead has some space on both sides, making it relatively tall and narrow. When 亻 (or any character) is used as a component, however, this `points` attribute is ignored, since the `comp` tag has its own `points` attribute. The stroke points should always make a component touch all four edges of its grid, so that its bounding rectangle is `0,0 128,128` before any scaling is applied. The `points` attribute is even more important for 口 ("mouth"), which has

a large amount of space on all four sides; characters like 因 look best with a smaller amount of space on all four sides. In general, any character with a stroke running along an outside edge tends to look better (especially in juxtaposition with other characters) with some space on that edge; so, 相 might have `points="0,0 124,128"`.

5. Instead of, or in addition to, the char attribute, CDL supports a `uni` attribute, whose value is a hexadecimal Unicode scalar value (USV). For example, `uni="592A"` has the same meaning as `char="太"`. Simultaneous use of `char` and `uni` attributes is redundant but sometimes convenient. If both are used, they should be consistent. An optional `variant` attribute can be used in addition to either `char` or `uni`, to associate identification strings for distinguishing multiple descriptions for the same USV.

6. There is a widely-used system, commonly known as the 札 *zhá* system, which puts all strokes into only five stroke categories: 一 *héng*, 丨 *shù*, 丿 *piě*, 丶 *diǎn*, and 乛 *zhé*. Each of the less than fifty CDL stroke types belongs to one of the five 札 *zhá* stroke categories. Thus, 札 *zhá* classification can easily be obtained from a CDL description.

7. There are `head` and `tail` attributes for stroke elements, which describe minor changes to beginning and end points of strokes, respectively. Such changes are important for some typeface styles, especially where strokes join; however, they can be ignored for some simple styles, and for many applications of CDL. They are documented in another article, along with the list of stroke types.

8. A more explicit form of recursion could be supported, with one `cdl` tag allowed to occur inside of another, acting as an anonymous component. This would be one solution to the problem of unencoded components. Another solution is to assign private-use codes to unencoded components, give them separate descriptions in the same database, and use `comp` tags. The latter solution has the advantage that the same component can be used in more than one character without duplicating its description. Ideally, however, there should be standard (not private-use) codes for many components that are useful in CDL.

9. There are a few more optional attributes (such as a `radical` attribute for specifying which strokes in a character are considered to be its radical), which are beyond the scope of this article.

10. Actually, Unicode includes two compatibility characters, related to `U+8005` 者, namely `U+FA5B` and `U+2F97A`. The difference between them seems to involve a slight difference in the position of the extra dot.

11. In this context, we only distinguish “variants” if they have nontrivial differences in their CDL descriptions. (Slight coordinate differences can be regarded as trivial.) If there are two or more distinct CDL descriptions of a unified character, we call them all “variants” of each other, without any implication about relative correctness or deviance, since in general those qualities depend on the locale, the context, and/or the eye of the beholder.

12. While CDL can’t solve all the difficulties of Han unification and variation, it can go a long way toward making the principles and procedures more rational. Just the ability to

produce self-consistent radical and stroke-count indexes of the currently encoded Han characters will be an advance. It would be a mistake to assume that stroke count will always be fuzzy and ill-defined, and that when looking up a character, people will always have to be prepared to add or subtract one or two from the stroke count when their first guess fails. On the contrary, within particular locales, such as the PRC, a tremendous amount of careful work has been done, and official publications such as 现代汉语通用字笔顺规范 (ISBN 7-80126-201-8) have standardized not only the stroke counts but also the stroke orders and stroke categories for thousands of characters. The stroke count of one character is generally related to the stroke counts of other characters. Most characters are built from components, and as long as the stroke counts of those components are defined, there is rarely any difficulty in adding them together to obtain the combined stroke count. Therefore, if a standard defines the strokes of a few thousand characters, it implicitly defines the strokes of many thousands of additional characters.

13. There are conflicting conventions (in different countries, or even in the same country) for the strokes of some characters that are nevertheless unified in Unicode. Using a `variant` attribute in addition to a `char` (or `uni`) attribute, a standard CDL database can include several variants of a unified character, possibly with different strokes or components. Some kind of "variant selectors" could in this way be given very precise meanings. Whether to associate certain variants with certain locales is another question, perhaps best decided separately by implementations for particular locales; an international standard would simply specify which variant selectors correspond to which CDL descriptions.

14. Here is a minimal DTD (Document Type Definition); it omits a few optional or experimental attributes that were not mentioned in this document:

```
<?xml  encoding="UTF-8"?>
<!ELEMENT  cdl-list  (cdl)+>
<!ELEMENT  cdl  (comp|stroke)+>
<!ELEMENT  comp  EMPTY>
<!ELEMENT  stroke  EMPTY>
<!ATTLIST  cdl
     char           CDATA  #IMPLIED
     uni            CDATA  #IMPLIED
     variant        CDATA  #IMPLIED
     points         CDATA  #IMPLIED
>
<!ATTLIST  comp
     char           CDATA  #IMPLIED
     uni            CDATA  #IMPLIED
     variant        CDATA  #IMPLIED
     points         CDATA  #IMPLIED
>
<!ATTLIST  stroke
     type           CDATA          #IMPLIED
     points         CDATA          #IMPLIED
     head  (cut|corner|vertical)   #IMPLIED
     tail  (cut|long)              #IMPLIED
>
```