Developing CMap Resources for CID-Keyed Fonts

This document is a tutorial and "best practices" document for font developers and sophisticated end users who develop CMap resources intended to be used with CID-keyed fonts, or used for building OpenType/CFF fonts. Some familiarity with CID-keyed font technology is assumed. Tools, written in Perl, that simplify CMap resource development are also provided in this document, in the text itself or as an attachment. The techniques, principles, and tools that are presented in this document can be applied to the development of CMap resources for any character collection.

Because this document is designed to be a tutorial, it provides practical advice and helpful tips for developing CMap resources. A complete description and treatment of CMap resources, to include the full syntactic specification, is available in Adobe Technical Note #5014, entitled *Adobe CMap and CIDFont Files Specification*.* A complete description of the various CJK character collections and their associated CMap resources can be found in Adobe Technical Note #5094, entitled *Adobe CJKV Character Collections and CMaps for CID-Keyed Fonts*.† Adobe's public CMap resources are available as part of the *CMap Resources* open source project that is hosted at Open @ Adobe.‡

The software provided as part of this document, written in Perl, comes with no warranty, written or expressed. Adobe assumes no liability for data lost as a result of using them. Also note that Perl is required to execute these tools (Perl is freely available for a variety of platforms, and is bundled with some, such as Mac OS X).

1.1 Introduction

A CMap resource is used to specify, in a unidirectional manner, mappings from character codes, such as Unicode, to CIDs (*Character IDs*), and are designed to be used with CID-keyed fonts, such as CIDFont resources.

If you are unfamiliar with the structure and syntax of a CMap resource, this section provides a short example. CMap resource syntax is simply a specialized set of PostScript language commands. For information on PostScript language syntax, please refer to *PostScript Language Reference*, Third Edition (Addison-Wesley, 1999).§

The following is a minimal—but fully-functional—Unicode (UTF-32 encoding form) CMap resource named *UniJIS-UTF32-H*. It maps a single Unicode character, U+4E00, from its UTF-32 encoding form to its appropriate Adobe-Japan1-6 CID, specifically CID+1200.

```
%!PS-Adobe-3.0 Resource-CMap
%%DocumentNeededResources: ProcSet (CIDInit)
%%IncludeResource: ProcSet (CIDInit)
%%BeginResource: CMap (UniJIS-UTF32-H)
%%Title: (UniJIS-UTF32-H Adobe Japan1 6)
%%Version: 1.000
%%Copyright: ----------------------------------------------------------
%%Copyright: Copyright 1990-2012 Adobe Systems Incorporated.
%%Copyright: All rights reserved.
```

* http://www.adobe.com/devnet/font/pdfs/5014.CIDFont_Spec.pdf
† http://www.adobe.com/devnet/font/pdfs/5094.CJK_CID.pdf
‡ http://sourceforge.net/adobe/cmap/
§ http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/postscript/pdfs/PLRM.pdf

```
%%Copyright:
%%Copyright: Redistribution and use in source and binary forms, with or
%%Copyright: without modification, are permitted provided that the
%%Copyright: following conditions are met:
%%Copyright:
%%Copyright: Redistributions of source code must retain the above
%%Copyright: copyright notice, this list of conditions and the following
%%Copyright: disclaimer.
%%Copyright:
%%Copyright: Redistributions in binary form must reproduce the above
%%Copyright: copyright notice, this list of conditions and the following
%%Copyright: disclaimer in the documentation and/or other materials
%%Copyright: provided with the distribution.
%%Copyright:
%%Copyright: Neither the name of Adobe Systems Incorporated nor the names
%%Copyright: of its contributors may be used to endorse or promote
%%Copyright: products derived from this software without specific prior
%%Copyright: written permission.
%%Copyright:
%%Copyright: THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
%%Copyright: CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
%%Copyright: INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
%%Copyright: MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
%%Copyright: DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
%%Copyright: CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
%%Copyright: SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
%%Copyright: NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
%%Copyright: LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
%%Copyright: HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
%%Copyright: CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
%%Copyright: OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
%%Copyright: SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
%%Copyright: ----------------------------------------------------------
%%EndComments

/CIDInit /ProcSet findresource begin

12 dict begin

begincmap

/CIDSystemInfo 3 dict dup begin
  /Registry (Adobe) def
  /Ordering (Japan1) def
  /Supplement 6 def
end def

/CMapName /UniJIS-UTF32-H def
/CMapVersion 1.000 def
/CMapType 1 def

/XUID [1 10 25539] def

/WMode 0 def

1 begincodespacerange
  <00000000> <0010FFFF>
endcodespacerange

1 beginnotdefrange
<00000000> <0000001f> 1
endnotdefrange

100 begincidchar
<00004e00> 1200
endcidchar
```

```
endcmap
CMapName currentdict /CMap defineresource pop
end
end

%%EndResource
%%EOF
```

You will discover that much of a CMap resource's structure is static in that it does not change from CMap resource to CMap resource.

The sections that follow step you through each section of a CMap resource, using the above minimal CMap resource as the example. Special attention is given to those sections that specify code space ranges and the actual character code to CID mappings.

1.2 The CMap Resource Header

The CMap resource header is simply a series of PostScript comments, but their content is important, because it is parsed by some software.

```
%!PS-Adobe-3.0 Resource-CMap
%%DocumentNeededResources: ProcSet (CIDInit)
%%IncludeResource: ProcSet (CIDInit)
%%BeginResource: CMap (UniJIS-UTF32-H)
%%Title: (UniJIS-UTF32-H Adobe Japan1 6)
%%Version: 1.000
%%EndComments
```

The first three lines are static. The first is the typical comment line ('percent bang') that indicates a PostScript resource. The next two lines indicate that the CID procedure set (called *ProcSet*) resource is required, and is in a resource called *CIDInit*.

```
%!PS-Adobe-3.0 Resource-CMap
%%DocumentNeededResources: ProcSet (CIDInit)
%%IncludeResource: ProcSet (CIDInit)
```

The next line specifies the CMap resource name in parentheses:

```
%%BeginResource: CMap (UniJIS-UTF32-H)
```

The following line specifies the CMap resource name and the /Registry, /Ordering, and /Supplement of the character collection on which it is based, again in parentheses:

```
%%Title: (UniJIS-UTF32-H Adobe Japan1 6)
```

The following line specifies the version number of the CMap resource, indicated by a real number:

```
%%Version: 1.000
```

Note the lack of parentheses, because only PostScript-language strings are delimited by parentheses.

CMap resources developed by Adobe also include an open-source copyright notice in the header as follows:

```
%%Copyright: ----------------------------------------------------------
%%Copyright: Copyright 1990-2012 Adobe Systems Incorporated.
%%Copyright: All rights reserved.
%%Copyright:
%%Copyright: Redistribution and use in source and binary forms, with or
%%Copyright: without modification, are permitted provided that the
%%Copyright: following conditions are met:
%%Copyright:
%%Copyright: Redistributions of source code must retain the above
%%Copyright: copyright notice, this list of conditions and the following
%%Copyright: disclaimer.
```

```
%%Copyright:
%%Copyright: Redistributions in binary form must reproduce the above
%%Copyright: copyright notice, this list of conditions and the following
%%Copyright: disclaimer in the documentation and/or other materials
%%Copyright: provided with the distribution.
%%Copyright:
%%Copyright: Neither the name of Adobe Systems Incorporated nor the names
%%Copyright: of its contributors may be used to endorse or promote
%%Copyright: products derived from this software without specific prior
%%Copyright: written permission.
%%Copyright:
%%Copyright: THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
%%Copyright: CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
%%Copyright: INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
%%Copyright: MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
%%Copyright: DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
%%Copyright: CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
%%Copyright: SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
%%Copyright: NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
%%Copyright: LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
%%Copyright: HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
%%Copyright: CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
%%Copyright: OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
%%Copyright: SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
%%Copyright: ----------------------------------------------------------
```

Whether to include similar copyright information, or any copyright information at all, is at the discretion of the developer of the CMap resource.

The final line of the header simply indicates the end of the CMap resource header.

```
%%EndComments
```

Vertical CMap resources use their horizontal counterpart for the majority—and sometimes all—of its mappings, and thus include extra entries in the header to indicate this external dependency. The following is what the header of the *UniJIS-UTF32-V* CMap resource looks like (copyright information is not shown for the sake of brevity):

```
%!PS-Adobe-3.0 Resource-CMap
%%DocumentNeededResources: ProcSet (CIDInit)
%%DocumentNeededResources: CMap (UniJIS-UTF32-H)
%%IncludeResource: ProcSet (CIDInit)
%%IncludeResource: CMap (UniJIS-UTF32-H)
%%BeginResource: CMap (UniJIS-UTF32-V)
%%Title: (UniJIS-UTF32-V Adobe Japan1 6)
%%Version: 1.000
%%EndComments
```

Note the two additional lines, specifically the third and fifth lines from the above CMap resource header:

```
%%DocumentNeededResources: CMap (UniJIS-UTF32-H)
%%IncludeResource: CMap (UniJIS-UTF32-H)
```

These lines indicate that this CMap resource depends on the presence and contents of the horizontal CMap resource, *UniJIS-UTF32-H*.

1.3 The CMap Resource Body

The first three lines of the CMap resource body are static for all CMap resources. The purpose of the first line is to check for the existence of the CID procedure set (*ProcSet*) as a resource named *CIDInit*:

```
/CIDInit /ProcSet findresource begin
```

If the result returned from the interpreter is true, then the device is enabled for CID-keyed fonts. This does not apply to ATM and other non-PostScript implementations.

The following line creates a new dictionary with 12 entries, the purpose of which is to store the dictionary entries that follow:

```
12 dict begin
```

Storing items in a PostScript dictionary makes them subsequently accessible.

The following line indicates the start of the CMap resource dictionary:

```
begincmap
```

The following dictionary entry, which uses the *usecmap* operator, generally appears only in vertical CMap resources, and indicates that all the mappings from the referenced CMap resource are inherited.

```
/UniJIS-UTF32-H usecmap
```

Our example CMap resource did not include this line because it is for horizontal use. The *UniJIS-UTF32-V* CMap resource, its vertical counterpart, includes this dictionary entry.

1.3.1 Compatibility Information

The three-item /CIDSystemInfo dictionary establishes compatibility information for the CMap resource, specifically the /Registry, /Ordering, and /Supplement values. (These three dictionary entries, sometimes abbreviated as *ROS*, can also be referred to using a single string that is a concatenation using a hyphen as a separator, such as *Adobe-Japan1-6*.) The same dictionary is also present in CIDFont resources. These entries must agree with the character collection with which the CMap resource is to function.

```
/CIDSystemInfo 3 dict dup begin
  /Registry (Adobe) def
  /Ordering (Japan1) def
  /Supplement 6 def
end def
```

NOTE: A matching /Supplement entry is not required for CIDFont and CMap resource compatibility.

The /Supplement value is not enclosed in parentheses, because it is an integer value, not a string. The example CMap resource is thus compatible with the Adobe-Japan1-6 character collection, but can be used with CIDFont resources that specify a /Supplement value other than 6.

The following line, the dictionary entry /CMapName, specifies the name of the CMap resource:

```
/CMapName /UniJIS-UTF32-H def
```

In this case, the name of the CMap resource is *UniJIS-UTF32-H*.

The following line, the dictionary entry /CMapVersion, indicates the version number of the CMap resource, expressed as a real number. This should agree with the value that is specified in the header section (the *%%Version:* line).

```
/CMapVersion 1.000 def
```

The following line indicates the type of CMap resource:

```
/CMapType 1 def
```

This obviously allows the possibility of different CMap resource formats. “ToUnicode” mapping resources, for example, specify 2 as the /CMapType value.* For the purpose of developing CMap resources as detailed in this document, please treat this entry as the standard way of specifying the CMap resource type.

* http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/5411.ToUnicode.pdf

1.3.2 /XUID & /UIDOffset Values

Next, there are the /XUID and /UIDOffset dictionary entries. These values are used to cache rendered bitmaps in VM (virtual memory) or on disk for faster imaging during subsequent rendering requests.

The /XUID array is used by PostScript Level 2 and PostScript 3 for caching purposes. The first element must be the Font XUID identifier of the developer of the CMap resource, and in the example it is set to 1 (Adobe):

```
/XUID [1 10 25539] def
```

If you do not yet have an assigned Font XUID, one can be easily registered with Adobe.* The second and subsequent elements of the /XUID array are assigned by the developer, to ensure uniqueness, and can signify any meaning they wish. In the example, the second element designates the resource type, with a value of 10 (Adobe's convention is to specify 10 for CMap resources, and 11 for CIDFont resources). The third element is an integer that is unique across Adobe's CMap resources.

NOTE: Due to known implementation limits, we recommend that no more than four elements be specified in /XUID arrays.

The /UIDOffset value is used to allocate a range of UniqueIDs, and its sole purpose is for backwards compatibility with PostScript Level 1. Thus, only legacy CMap resources are expected to include a /UIDOffset value.

NOTE: Adobe does not recommend specifying a /UIDOffset value in CMap resources.

A /UIDOffset value is not present in the example CMap resource, *UniJIS-UTF32-H*. One is not necessary. The following is the /UIDOffset entry for the Adobe-Japan1-6 *H* CMap resource:

```
/UIDOffset 280 def
```

This value simply means that UniqueIDs for the Adobe-Japan1-6 *H* CMap resource are assigned starting from an offset of 280 UniqueIDs from a base value that is specified in a CIDFont resource, specifically as the /UIDBase dictionary value. The *H* CMap resource consumes only 94 UniqueIDs, but 100 UniqueIDs are assigned for reasons of convenience (this means that the /UIDOffset value in the next CMap resource should be no lower than 380). If the /UIDBase value of a CIDFont resource is set to 2200000, then the *H* CMap resource will begin allocating UniqueIDs starting from the value 2200280.

1.3.3 Writing Modes

There are two writing modes in PostScript, indicated by the integers 0 and 1. Writing mode 0 is horizontal, and writing mode 1 is vertical. These should agree with the last character in the name of the CMap resource, meaning 0 with -H, and 1 with -V. The following is from the example CMap resource, which is horizontal:

```
/WMode 0 def
```

The corresponding vertical CMap resource, *UniJIS-UTF32-V*, uses the following line instead:

```
/WMode 1 def
```

The origin of writing mode 0 is at coordinate 0,0, and the writing direction is from left to right. The origin of writing mode 1 begins at coordinate 0,0, and the writing direction is from top to bottom.

NOTE: The origin of writing mode 1 at coordinate 0,0 is at a vector of 500,880 from the origin of writing mode 0, meaning at the top-center of a full-width 1000-unit character.

* http://partners.adobe.com/public/developer/font/register/xuid_reg.do

1.3.4 Code Space Range

The part of the CMap resource that is described in this section specifies the code space range for the encoding to be supported, and establishes the legal character codes for the subsequent character code to CID mappings.

NOTE: It is extremely important to understand that the entire code space range should be specified, and not only the code space range from which character codes are mapped to CIDs. This is a common error that is made by developers of CMap resources! Some encodings that have disjoint second-byte ranges, such as Shift-JIS (Japanese), Big Five (Traditional Chinese), Unified Hangul Code (Korean), or Johab (Korean), are best expressed as a contiguous block.

You will learn that a rich set of encodings can be specified by a CMap resource. One- to four-byte representations can be expressed. A current limitation of CID compatibility mode (available in PostScript Version 2014 and earlier) is that only representations up to two bytes are supported, but native mode (available in PostScript Version 2015 and later) can handle up to four-byte representations.

Code space ranges are expressed as hexadecimal codes that represent the beginning and ending value of each byte of a contiguous encoding block. Uppercase hexadecimal digits A through F (as opposed to lowercase hexadecimal digits a through f) are used—as a convention rather than as a rule—to more easily distinguish the code space range definition from the character code to CID mappings, but this is not a requirement. The following is an example code space range definition:

```
1 begincodespacerange
  <0000> <FFFF>
endcodespacerange
```

In this case, the code space range is completely two-byte, and is 0x0000 through 0xFFFF (65,536 code points). This represents the now-deprecated Unicode UCS-2 encoding form.

The integer immediately before the *begincodespacerange* operator indicates and must agree with the total number of code space range entries. Furthermore, all code space ranges are required to be in increasing byte-value order.

For today's OSes and applications, the most important CMap resources are the Unicode ones, which support the UTF-8, UTF-16, and UTF-32 encoding forms. When developing OpenType/CFF fonts, the most important CMap resource is the UTF-32 one.

The code space range for UTF-32 CMap resources can be specified by a single entry, as shown below:

```
1 begincodespacerange
  <00000000> <0010FFFF>
endcodespacerange
```

Note that the byte order is explicitly big-endian, so it is technically the UTF-32BE encoding form.

The code space range for UTF-16 CMap resources also uses big-endian byte order, and can be specified by the three entries below:

```
3 begincodespacerange
  <0000>     <D7FF>     % BMP
  <D800DC00> <DBFFDFFF> % Surrogates
  <E000>     <FFFF>     % BMP
endcodespacerange
```

The UTF-8 encoding form uses 8-bit code units, so there is only one byte order. The code space ranges for UTF-8 CMap resources are shown below:

```
4 begincodespacerange
  <00>       <7F>
  <C080>     <DFBF>
  <E08080>   <EFBFBF>
  <F0808080> <F7BFBFBF>
endcodespacerange
```

Other code space ranges, for legacy encodings, include the examples that follow. These can be used as study material, and may be useful for developing other CMap resources. The first example is for two-byte 7-bit ISO-2022 encoding (applicable to JIS X 0208, JIS X 0212, GB 2312, KS X 1001, and each individual plane of CNS 11643):

```
1 begincodespacerange
 <2121> <7E7E>
endcodespacerange
```

This specifies the usual 8,836 code points in a 94×94 matrix.

The next example is for Shift-JIS encoding (JIS X 0201 and JIS X 0208):

```
4 begincodespacerange
  <00>   <80>   % JIS X 0201 Latin
  <8140> <9FFC> % JIS X 0208 first two-byte block
  <A0>   <DF>   % JIS X 0201 Half-width katakana
  <E040> <FCFC> % JIS X 0208 second two-byte block
endcodespacerange
```

Note how the user-defined range (0xF040 through 0xFCFC) is included as part of the second two-byte block. Although the second-byte range of Shift-JIS is disjoint (0x40 through 0xFC, but not including 0x7F), it is expressed here as a contiguous range to simplify the character code to CID mappings that are assigned later in the CMap resource.

Another example is for EUC-JP encoding (JIS X 0201, JIS X 0208, and JIS X 0212), which makes use of one-, two-, and three-byte character codes:

```
4 begincodespacerange
<00>     <80>     % EUC code set 0 (JIS X 0201 Latin)
<8EA0>   <8EDF>   % EUC code set 2 (JIS X 0201 half-width katakana)
<8FA1A1> <8FFEFE> % EUC code set 3 (JIS X 0212)
<A1A1>   <FEFE>   % EUC code set 1 (JIS X 0208)
endcodespacerange
```

The next example is for EUC-CN (GB 1988 and GB 2312) and EUC-KR (KS X 1003 and KS X 1001) encodings:

```
2 begincodespacerange
  <00>   <80>   % EUC code set 0 (GB 1988 or KS X 1003)
  <A1A1> <FEFE> % EUC code set 1 (GB 2312 or KS X 1001)
endcodespacerange
```

The next example is for EUC-TW encoding (ASCII and CNS 11643):

```
18 begincodespacerange
  <00>       <80>       % EUC code set 0 (ASCII)
  <8EA1A1A1> <8EA1FEFE> % EUC code set 2 (Plane 1)
  <8EA2A1A1> <8EA2FEFE> % EUC code set 2 (Plane 2)
  <8EA3A1A1> <8EA3FEFE> % EUC code set 2 (Plane 3)
  <8EA4A1A1> <8EA4FEFE> % EUC code set 2 (Plane 4)
  <8EA5A1A1> <8EA5FEFE> % EUC code set 2 (Plane 5)
  <8EA6A1A1> <8EA6FEFE> % EUC code set 2 (Plane 6)
  <8EA7A1A1> <8EA7FEFE> % EUC code set 2 (Plane 7)
  <8EA8A1A1> <8EA8FEFE> % EUC code set 2 (Plane 8)
  <8EA9A1A1> <8EA9FEFE> % EUC code set 2 (Plane 9)
  <8EAAA1A1> <8EAAFEFE> % EUC code set 2 (Plane 10)
  <8EABA1A1> <8EABFEFE> % EUC code set 2 (Plane 11)
  <8EACA1A1> <8EACFEFE> % EUC code set 2 (Plane 12)
  <8EADA1A1> <8EADFEFE> % EUC code set 2 (Plane 13)
```

```
  <8EAEA1A1> <8EAEFEFE> % EUC code set 2 (Plane 14)
  <8EAFA1A1> <8EAFFEFE> % EUC code set 2 (Plane 15)
  <8EB0A1A1> <8EB0FEFE> % EUC code set 2 (Plane 16)
  <A1A1>     <FEFE>     % EUC code set 1 (Plane 1)
endcodespacerange
```

Note how CNS 11643 Plane 1 is (redundantly) encoded in both EUC-TW code sets 1 and 2, and how a mixture of one-, two-, and four-byte character codes is used. You may be wondering whether it is possible to express the second through seventeenth entries as the following single entry:

```
<8EA1A1> <8EB0FEFE> % EUC code set 2 (CNS 11643)
```

The answer is yes and no. If the CMap resource does not specify a /UIDOffset value, then the answer is *yes*. If a /UIDOffset value is specified, then the answer is *no*. The reason is due to the way in which UniqueIDs are allocated. The least significant byte can have up to 256 values, 0x00 through 0xFF. A single UniqueID can handle up to 256 character codes. This means that the number of UniqueIDs required for each code space range line is determined by the number of unique values in the second byte from the right (highlighted in bold above). If any other byte beyond the second byte is represented by a range, then UniqueID assignment will not work properly. In the single code space range entry above, one of the bytes beyond the second byte, specifically the second of the four bytes, ranges from 0xA1 through 0xB0.

The next example is for IBM's DBCS-Host encoding (applicable for Chinese, Japanese, and Korean):

```
2 begincodespacerange
  <4040> <4040> % Full-width space character
  <4141> <FEFE> % Two-byte characters
endcodespacerange
```

Pay careful attention to how the code point for the full-width "space" character, 0x4040, is handled: an code space range that consists of a single two-byte code point.

The next example is for Big Five encoding (Traditional Chinese, and equivalent to CNS 11643 Planes 1 and 2):

```
2 begincodespacerange
  <00>   <80>   % ASCII
  <A140> <FEFE> % Big Five
endcodespacerange
```

Big Five encoding uses two disjoint two-byte encoding blocks, specifically 0xA140 through 0xFE7E and 0xA1A1 through 0xFEFE, but the code space range specified above joins them by including the intervening encoding block, meaning 0xA17F through 0xFEA0. This is done to simplify the character code to CID mappings that are specified later in the CMap resource.

The next example is for Johab encoding, which includes all 11,172 hangul (KS X 1003 and KS X 1001):

```
4 begincodespacerange
  <00>   <80>   % KS X 1003
  <8441> <D3FE> % KS X 1001 Hangul
  <D831> <DEFE> % KS X 1001 Symbols
  <E031> <F9FE> % KS X 1001 Hanja
endcodespacerange
```

The next example is for Unified Hangul Code (UHC) encoding, supported in older versions of Microsoft Windows for Korean. It supports the same character set as Johab encoding, but its code space range is different:

```
2 begincodespacerange
  <00>   <80>   % KS X 1003
  <8141> <FEFE> % KS X 1001 plus 8,822 additional hangul
endcodespacerange
```

The final example demonstrates that one-byte CMaps resources can also be defined:

```
1 begincodespacerange
  <00> <FF> % ASCII or EBCDIC
endcodespacerange
```

This code space range is applicable for both ASCII- or EBCDIC-style encodings.

The code space range section is not found in vertical CMap resources. Vertical CMap resources instead inherit the code space range from the appropriate horizontal CMap resource, as specified by the *usecmap* operator.

It is possible to specify the largest possible code space range, such as the following for 7-bit ISO-2022 encoding, which is normally set to 0x2121 through 0x7E7E:

```
1 begincodespacerange
  <0000> <FFFF>
endcodespacerange
```

This method wastes a significant amount of UniqueIDs (over 150), but more importantly it permits a large number of character codes outside the valid 7-bit ISO-2022 encoding range. Joining non-contiguous encoding blocks that share the same first-byte values, such as we saw with Shift-JIS and Big Five encodings, does not waste any UniqueIDs.

1.3.5 .Notdef Range

If a special mapping is desired for particular characters or character ranges, such as for the ASCII control character range, 0x00 through 0x1F, or U+0000 through U+001F in Unicode, a specific CID can be assigned for this purpose. By default, unmapped character codes are assigned CID+0 (*.notdef*). This is not always desired because the width of CID+0 is typically 1000 units (one em), and may also render with a specific glyph. The following is an example:

```
1 beginnotdefrange
<00000000> <0000001f> 1
endnotdefrange
```

In this case, the Unicode (UTF-32) control character range, U+0000 through U+001F, is assigned CID+1, which represents the glyph for a proportional-width space in the Adobe-CNS1-6, Adobe-GB1-5, Adobe-Japan1-6, and Adobe-Korea1-2 character collections. The glyph for a half-width space can be assigned for Shift-JIS or EUC-JP encoding as follows (the specified CID, CID+231, applies only to the Adobe-Japan1-6 character collection):

```
1 beginnotdefrange
<00> <1f> 231
endnotdefrange
```

Whether to specify the glyph for a half- or proportional-width space depends on the characteristics of the CMap resource. For example, the Macintosh CMap resource for Japanese, *90pv-RKSJ-H*, uses proportional-width glyphs for the JIS X 0201 character set, and thus specifies CID+1 (a proportional-width space) for the control character range as follows:

```
1 beginnotdefrange
<00> <1f> 1
endnotdefrange
```

CMap resources that use a half-width ASCII (or equivalent) character set should specify the CID that corresponds to a half-width space for this section, such as CID+231 for the Adobe-Japan1-6 character collection.

NOTE: Some implementations ignore the .notdef range that is specified in a CMap resource.

1.3.6 Character Code to CID Mapping Basics

This section represents the most important part of the CMap resource because it specifies all of the character code to CID mappings. The following minimal mapping supports a single character code:

```
1 begincidchar
<4e00> 1200
endcidchar
```

Character codes are expressed in hexadecimal notation, and are enclosed in < > brackets. CIDs are expressed as decimal integers.

The character code 0x4E00 simply maps to CID+1200. The number that appears before the *begincidchar* operator indicates the number of mapping lines. The *endcidchar* operator terminates a block of mappings. If a CMap resource specifies more than 100 mappings, they must be segmented into blocks of 100 mapping, as follows:

```
100 begincidrange
<2121> <217e> 633
<2221> <222e> 727
<223a> <2241> 741
<224a> <2250> 749
<...92 mappings omitted...>
<6021> <607e> 5594
<6121> <617e> 5688
<6221> <627e> 5782
<6321> <637e> 5876
endcidrange

18 begincidrange
<6421> <647e> 5970
<6521> <657e> 6064
<6621> <667e> 6158
<6721> <677e> 6252
<...10 mappings omitted...>
<7221> <727e> 7286
<7321> <737e> 7380
<7421> <7424> 7474
<7425> <7426> 8284
endcidrange
```

In the above example, the 118 mappings that are specified in the Adobe-Japan1-6 *H* CMap resource are separated into a block of 100 followed by a block of 18.

As the two examples above exemplify, there are two types of mappings: *single* and *range*.

Single mappings map a single character code to a CID, and are delimited by the *begincidchar* and *endcidchar* operators. The two entries of a single mapping comprise the character code and the CID.

Range mappings map a contiguous range of character codes to a contiguous range of CIDs, and are delimited by the *begincidrange* and *endcidrange* operators. The three entries of a range mapping comprise the starting character code, the ending character code, and the starting CID. It is possible to represent single mappings as range mappings, meaning that the starting and ending character codes are identical.

All single mappings are specified first in a CMap resource, followed by all range mappings.

The following range mapping specifies the 94 characters that make up the first row of *kanji* from the Adobe-Japan1-6 *H* CMap resource—this is 7-bit ISO-2022 encoding:

```
<3021> <307e> 1125
```

This is equivalent to writing the following single mappings:

```
<3021> 1125
<3022> 1126
<3023> 1127
<3024> 1128
<...86 mappings omitted...>
<307b> 1215
```

```
<307c> 1216
<307d> 1217
<307e> 1218
```

but is much more efficient. In principle, if both character code and CID ranges are contiguous, they should be collapsed into a range mapping. This results in a shorter (smaller) CMap resource, and increases efficiency.

The corresponding 94 character codes in the Adobe-Japan1-6 *78-H* (JIS C 6226-1978) CMap resource does not have completely contiguous CIDs, although the character codes are contiguous. The following shows how the same set of 94 character codes as shown above, specifically 0x3021 through 0x307E, are mapped to their corresponding Adobe-Japan1-6 CIDs, and use single and range mappings as appropriate:

```
9 begincidchar
<3021> 1125
<3022> 7633 % JIS C 6226-1978 kanji
<3029> 8266 % JIS C 6226-1978 kanji
<3032> 7961 % JIS C 6226-1978 kanji
<3033> 7330 % JIS C 6226-1978 kanji
<303b> 7634 % JIS C 6226-1978 kanji
<306e> 7635 % JIS C 6226-1978 kanji
<3073> 7636 % JIS C 6226-1978 kanji
<307c> 7637 % JIS C 6226-1978 kanji
endcidchar

7 begincidrange
<3023> <3028> 1127
<302a> <3031> 1134
<3034> <303a> 1144
<303c> <306d> 1152
<306f> <3072> 1203
<3074> <307b> 1208
<307d> <307e> 1217
endcidrange
```

Note how non-contiguity of CIDs results in significantly more mappings (16 for the *78-H* CMap resource, compared to only one for the *H* CMap resource). The order of mapping lines must also be in increasing byte-value order, meaning that the following mapping order is not acceptable, and results in an invalid CMap resource:

```
<3121> <317e> 1219
<3021> <307e> 1125
```

It must instead be as follows:

```
<3021> <307e> 1125
<3121> <317e> 1219
```

While the character codes must be in increasing byte-value order, there is no such stipulation for the order of CIDs.

In some cases, a CMap resource may exist, but contain no character code to CID mapping lines. The vertical CMap resources for the now-deprecated Adobe-Japan2-0 character collection, *Hojo-V* and *Hojo-EUC-V*, are such examples. Their sole purpose is to specify writing mode 1 (vertical), but no character code to CID mappings are necessary because all of the character code to CID mappings are inherited from their corresponding horizontal CMap resources.

1.3.7 Building Character Code to CID Mappings

The previous section described the basics of character code to CID mappings. This section contains practical advice about how to build up these mappings from scratch using tables or other sources. Proven tools for simplifying CMap resource development are also presented. These tools can be used as-is, or customized for other purposes.

One must begin with a table that maps each CID to a particular encoding, or to a name that represents a particular encoding. The following example represents the complete set of 6,355 kanji in JIS X 0208, and the CIDs are based on the Adobe-Japan1-6 character collection:

```
1125 3021
1126 3022
1127 3023
1128 3024
<. . . 6,347 contiguous CIDs and character codes omitted. . . >
7476 7423
7477 7424
8284 7425
8285 7426
```

The first column represents the CID (in decimal), and the second column represents the corresponding Japanese 7-bit ISO-2022 character code (in hexadecimal). Such a single CID to character code mapping alone is not very useful in building new CMap resources (because a 7-bit ISO-2022 CMap resource, *H*, already exists), but together with other mapping tables, can be used as a powerful development aid.

An example of this power can be demonstrated when trying to build a Unicode or DBCS-Host CMap resource. The Unicode Consortium provides a number of useful mapping tables for this purpose.* One such table maps Japanese 7-bit ISO-2022 to Unicode. IBM can also provide similar mappings tables for their DBCS-Host encoding. Such mapping tables provide the following type of information (Row 0x45 of Japanese DBCS-Host encoding, which has 190 characters, being the example used below):

```
4541 306c
4542 4673
4543 3b30
4544 3b4d
<. . . 182 character code mappings omitted. . . >
45fb 314a
45fc 346f
45fd 364c
45fe 423f
```

The first column is the DBCS-Host character code (in hexadecimal), and the second column is the Japanese 7-bit ISO-2022 character code (also in hexadecimal). The DBCS-Host encoding range for this excerpt is 0x4541 through 0x45FE. It is important that such mapping tables are sorted in increasing byte-value order.

By leveraging powerful text-processing tools, such as Awk, Perl, Python, or Ruby, one can use tables with thousands of such mappings, and quickly and easily create character code to CID mappings. Associative arrays serve as an ideal data structure for loading and retrieving such mapping information. Perl is an excellent programming language for manipulating CMap resources. It is not only free, and available for numerous platforms, but is extremely powerful. Its primary strength lies in its regular expressions, associative arrays, and parsing capabilities. Of course, the same statement also applies to other programming languages.

The following short Perl program is designed to create the raw (aka, not collapsed into range mappings) character code to CID mappings:

```
#!/usr/local/bin/perl

$count = 0;
open(CID,"<cid.tbl") || die "Error opening cid.tbl file.\n";
while (defined($line = <CID>)) {
  chop($line);
  ($cid,$code) = split(/\s+/,$line);
  $code =~ tr/A-F/a-f/;
  $table{$code} = $cid;
  $count++;
}
```

* http://www.unicode.org/Public/MAPPINGS/

```
close(CID);
print STDERR "Processed $count CID mappings...\n";

$count = 0;
while (defined($line = <STDIN>)) {
  chop($line);
  $line =~ tr/A-F/a-f/;
  ($new,$code) = split(/\s+/,$line);
  $cid = $table{$code};
  print STDOUT "<$new> $cid\n";
  $count++;
}
print STDERR "Created $count raw character code to CID mappings.\n";
```

A file that specifies the CID plus the equivalent Japanese 7-bit ISO-2022 code is first read in by explicitly opening the file named *cid.tbl*. The other mapping table in which each line indicates the target encoding plus the equivalent 7-bit ISO-2022 code is read in as standard input (STDIN). The result, which conforms to CMap resource syntax, is then written to standard output (STDOUT).

If the character code mapping table has multiple fields (that is, more than two), the following line of code:

```
($new,$code) = split(/\s+/,$line);
```

can be replaced with something like:

```
($new,$code) = (split(/\s+/,$line))[0,4];
```

to extract only the two desired fields. In the above line of code, the first field at array index 0 (zero) is assigned to the variable *$new*, and the fifth field at array index 4 is assigned to the variable *$code*. Perl arrays, like in C, begin at index 0 (zero). Other digits can be used to extract specific fields.

When using two appropriate mappings tables with this Perl utility, one can create the following 190 lines of CMap resource code that map DBCS-Host character codes to Adobe-Japan1-6 CIDs:

```
<4541> 1200
<4542> 3275
<4543> 2174
<4544> 2203
<...182 mappings omitted...>
<45fb> 1260
<45fc> 1579
<45fd> 1732
<45fe> 2847
```

Note how the character codes are in increasing byte-value order, and that neither the character codes nor the CIDs are contiguous, resulting in single mappings. The order of the Adobe-Japan1-6 CIDs that represent kanji favors legacy Japanese encodings, such as 7-bit ISO-2022, Shift-JIS, and EUC-JP. Encodings such as Unicode or DBCS-Host do not follow the same ordering (that is, the sequence in which characters are encoded), thus usually require many more mappings.

When building complex CMap resources with a large number of mappings that use non-contiguous character codes or non-contiguous CIDs, it is easier to begin with single mappings, which is the output of the above Perl program, then to later collapse those mappings that are contiguous. A tool, written in Perl, that can collapse contiguous mappings of a CMap resource is described at the end of this document, in Appendix A, and is also attached to this document. When processing these 190 DBCS-Host mappings with this tool, they are reduced by only one mapping, resulting in 189 mappings. Only the following two mappings have contiguous character codes and contiguous CIDs:

```
<45fa> 1259
<45fb> 1260
```

These two single mappings are subsequently reduced to the following range mapping:

```
<45fa> <45fb> 1259
```

This slight reduction in mappings is common when dealing with DBCS-Host and Unicode encodings. When building the entire Japanese DBCS-Host CMap resource based on the Adobe-Japan1-6 character collection, the approximately 7,200 character code to CID mappings are reduced to just under 5,000 mapping lines.

1.4 The CMap Resource Trailer

The following static lines terminate the CMap resource by closing the dictionary and defining it as a CMap resource:

```
endcmap
CMapName currentdict /CMap defineresource pop
end
end
%%EndResource
%%EOF
```

1.5 The Identity CMap Resource

The Identity CMap resource is a special-purpose CMap resource whose purpose is to allow all CIDs of a character collection to be accessible through the use of the CID values themselves. A standard CMap resource allows a subset of the character collection to be accessed through the use of conventional character codes. The name of an Identity CMap resource is the concatenation of the /Registry, /Ordering, and /Supplement values of a character collection, using a single hyphen as the joining element. So, the Identity CMap resource for the Adobe-Japan1-6 character collection is named as follows:

```
Adobe-Japan1-6
```

In other words, the same name as the character collection itself. This naming convention makes its content and purpose explicit.

One use for an Identity CMap resource is for proofing CIDFont resources. It is considered a “best practice” to proof a CIDFont resource by CID rather than by character code.

NOTE: The CMap resources, at least the ones developed by Adobe, are fully tested. They do not change from CIDFont resource to CIDFont resource. They are common to all CIDFont resources of a given character collection.

Adobe issues an Identity CMap resource for each Supplement of every public character collection. The special-purpose Adobe-Identity-0 character collection includes two special-purpose Identity CMap resources named *Identity-H* and *Identity-V*.

1.5.1 Unique Entries and Values

An Identity CMap resource includes a special dictionary entry not found in other CMap resources, specifically a /CIDCount value. An example is as follows (taken from the Identity CMap resource for the Adobe-Japan1-6 character collection):

```
/CIDCount 23058 def
```

This entry explicitly indicates the number of CIDs in the specified character collection.

The following are lines extracted from the Identity CMap resource for the Adobe-Japan1-6 character collection (two from the CMap resource header, and one from the CMap resource body):

```
%%BeginResource: CMap (Identity)
%%Title: (Identity Adobe Japan1 6)
/CMapName /Adobe-Japan1-6 def
```

Note how the string "Identity" is used for the CMap resource name in the header, but not in the body.

1.5.2 Identity CMap Resource Code Space Range

The code space range for Identity CMap resources begin with 0x0000, with an upper limit of 0xFFFF (this supports character collections with up to 65,535 CIDs, which is the maximum number of CIDs in a CIDFont resource, meaning CIDs 0 through 65534). But, instead of using the following code space range:

```
1 begincodespacerange
<0000> <FFFF>
endcodespacerange
```

the last occupied "row" is instead used as the upper limit, as follows (the example below is for the Adobe-Japan1-6 character collection, which has 23,058 CIDs):

```
1 begincodespacerange
<0000> <5AFF>
endcodespacerange
```

The hexadecimal equivalent of CID+23057 is 0x5A11, and the upper limit of 0x5AFF as expressed above covers this. Exactly what upper limit is specified depends on the number of CIDs in the character collection. The reason 0x0000 through 0xFFFF is not used is because the size of the code space range can affect memory usage.

Because an Identity CMap resource's character codes and CIDs are contiguous by definition, it is simply a matter of mapping blocks of 256 character codes, with second-byte values ranging from 0x00 through 0xFF, to the appropriate CIDs, as follows:

```
91 begincidrange
<0000> <00ff> 0
<0100> <01ff> 256
<0200> <02ff> 512
<0300> <03ff> 768
<. . . 83 mappings omitted. . . >
<5700> <57ff> 22272
<5800> <58ff> 22528
<5900> <59ff> 22784
<5a00> <5a11> 23040
endcidrange
```

1.5.3 Using Identity CMap Resources

The following examples demonstrates how Identity CMap resources can be used as arguments of the *findfont* or *selectfont* (PostScript Level 2 or PostScript 3) procedures:

```
/HeiseiMinStdN-W3--Adobe-Japan1-6 findfont 12 scalefont setfont
/HeiseiMinStdN-W3--Adobe-Japan1-6 12 selectfont
```

This usage and behavior is no different from that of conventional CMap resources. Subsequent use of the *show* operator take the following form, using CIDs 1500 and 1501 as the example:

```
<05dc 05dd> show
```

Values inside < > brackets are interpreted as hexadecimal values, and spaces are ignored. PostScript looping structures can be used to easily render all glyphs in a CIDFont resource.

1.5.4 Example Identity CMap Resource

The following is a complete Identity CMap resource, specifically that for the Adobe-Japan1-6 character collection. It can be used as a template for creating Identity CMap resources for other character collections.

```
%!PS-Adobe-3.0 Resource-CMap
%%DocumentNeededResources: ProcSet (CIDInit)
%%IncludeResource: ProcSet (CIDInit)
%%BeginResource: CMap (Identity)
%%Title: (Identity Adobe Japan1 6)
%%Version: 1.003
%%Copyright: -----------------------------------------------------------
%%Copyright: Copyright 1990-2009 Adobe Systems Incorporated.
%%Copyright: All rights reserved.
%%Copyright:
%%Copyright: Redistribution and use in source and binary forms, with or
%%Copyright: without modification, are permitted provided that the
%%Copyright: following conditions are met:
%%Copyright:
%%Copyright: Redistributions of source code must retain the above
%%Copyright: copyright notice, this list of conditions and the following
%%Copyright: disclaimer.
%%Copyright:
%%Copyright: Redistributions in binary form must reproduce the above
%%Copyright: copyright notice, this list of conditions and the following
%%Copyright: disclaimer in the documentation and/or other materials
%%Copyright: provided with the distribution.
%%Copyright:
%%Copyright: Neither the name of Adobe Systems Incorporated nor the names
%%Copyright: of its contributors may be used to endorse or promote
%%Copyright: products derived from this software without specific prior
%%Copyright: written permission.
%%Copyright:
%%Copyright: THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
%%Copyright: CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
%%Copyright: INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
%%Copyright: MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
%%Copyright: DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
%%Copyright: CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
%%Copyright: SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
%%Copyright: NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
%%Copyright: LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
%%Copyright: HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
%%Copyright: CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
%%Copyright: OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
%%Copyright: SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
%%Copyright: -----------------------------------------------------------
%%EndComments

/CIDInit /ProcSet findresource begin

12 dict begin

begincmap

/CIDSystemInfo 3 dict dup begin
  /Registry (Adobe) def
  /Ordering (Japan1) def
  /Supplement 6 def
end def

/CMapName /Adobe-Japan1-6 def
/CMapVersion 1.003 def
/CMapType 1 def

/XUID [1 10 25614] def

/WMode 0 def

/CIDCount 23058 def
```

```
1 begincodespacerange
  <0000> <5AFF>
endcodespacerange

91 begincidrange
<0000> <00ff> 0
<0100> <01ff> 256
<0200> <02ff> 512
<0300> <03ff> 768
<0400> <04ff> 1024
<0500> <05ff> 1280
<0600> <06ff> 1536
<0700> <07ff> 1792
<0800> <08ff> 2048
<0900> <09ff> 2304
<0a00> <0aff> 2560
<0b00> <0bff> 2816
<0c00> <0cff> 3072
<0d00> <0dff> 3328
<0e00> <0eff> 3584
<0f00> <0fff> 3840
<1000> <10ff> 4096
<1100> <11ff> 4352
<1200> <12ff> 4608
<1300> <13ff> 4864
<1400> <14ff> 5120
<1500> <15ff> 5376
<1600> <16ff> 5632
<1700> <17ff> 5888
<1800> <18ff> 6144
<1900> <19ff> 6400
<1a00> <1aff> 6656
<1b00> <1bff> 6912
<1c00> <1cff> 7168
<1d00> <1dff> 7424
<1e00> <1eff> 7680
<1f00> <1fff> 7936
<2000> <20ff> 8192
<2100> <21ff> 8448
<2200> <22ff> 8704
<2300> <23ff> 8960
<2400> <24ff> 9216
<2500> <25ff> 9472
<2600> <26ff> 9728
<2700> <27ff> 9984
<2800> <28ff> 10240
<2900> <29ff> 10496
<2a00> <2aff> 10752
<2b00> <2bff> 11008
<2c00> <2cff> 11264
<2d00> <2dff> 11520
<2e00> <2eff> 11776
<2f00> <2fff> 12032
<3000> <30ff> 12288
<3100> <31ff> 12544
<3200> <32ff> 12800
<3300> <33ff> 13056
<3400> <34ff> 13312
<3500> <35ff> 13568
<3600> <36ff> 13824
<3700> <37ff> 14080
<3800> <38ff> 14336
<3900> <39ff> 14592
<3a00> <3aff> 14848
<3b00> <3bff> 15104
<3c00> <3cff> 15360
```

```
<3d00> <3dff> 15616
<3e00> <3eff> 15872
<3f00> <3fff> 16128
<4000> <40ff> 16384
<4100> <41ff> 16640
<4200> <42ff> 16896
<4300> <43ff> 17152
<4400> <44ff> 17408
<4500> <45ff> 17664
<4600> <46ff> 17920
<4700> <47ff> 18176
<4800> <48ff> 18432
<4900> <49ff> 18688
<4a00> <4aff> 18944
<4b00> <4bff> 19200
<4c00> <4cff> 19456
<4d00> <4dff> 19712
<4e00> <4eff> 19968
<4f00> <4fff> 20224
<5000> <50ff> 20480
<5100> <51ff> 20736
<5200> <52ff> 20992
<5300> <53ff> 21248
<5400> <54ff> 21504
<5500> <55ff> 21760
<5600> <56ff> 22016
<5700> <57ff> 22272
<5800> <58ff> 22528
<5900> <59ff> 22784
<5a00> <5a11> 23040
endcidrange
endcmap
CMapName currentdict /CMap defineresource pop
end
end

%%EndResource
%%EOF
```

1.6 CMap Resource Filenames & Naming Conventions

CMap resources, when instantiated as a file, are named according to their /CMapName value (without the preceding slash).

CMap resource names established by Adobe are generally composed of up to three parts, each separated by a hyphen. The first part indicates the character set, the second part indicates the encoding, and the third part indicates writing direction. Although CMap resources can use arbitrary names, these conventions make it easier to identify characteristics of the character set and encoding that they are intended to support.

The table below lists the character set designators based on existing Adobe CMap resources, with the Unicode ones listed first because of their elevated importance:

Character Set Designator	Description
UniJIS	Adobe-Japan1 subset of Unicode
UniJIS2004	"UniJIS" with JIS X 0213:2004 prototypical glyphs
UniJISX0213	Adobe-Japan1 subset of Unicode—65 symbols map to proportional glyphs, not full-width ones

Character Set Designator	Description
UniJISX02132004	"UniJISX0213" with JIS X 0213:2004 prototypical glyphs
UniHojo	Adobe-Japan2 subset of Unicode *(deprecated)*
UniGB	Adobe-GB1 subset of Unicode
UniCNS	Adobe-CNS1 subset of Unicode
UniKS	Adobe-Korea1 subset of Unicode
78	JIS C 6226-1978 prototypical glyphs
78ms	"90ms" with JIS C 6226-1978 prototypical glyphs
83pv	Apple Macintosh KanjiTalk Version 6 character set
90ms	Microsoft Windows 3.1J character set
90msp	"90ms" with proportional glyphs for JIS X 0201
90pv	Apple Macintosh KanjiTalk Version 7 character set
Add	Fujitsu FMR Japanese character set
Ext	NEC Japanese character set
NWP	NEC Word Processor
Hojo	JIS X 0212-1990
GB	GB 2312-80
GBpc	"GB" with proportional glyphs for GB 1988
GBT	GB/T 12345-90
GBTpc	"GBT" with proportional glyphs for GB 1988
GBK	GBK
GBKp	"GBK" with proportional glyphs for GB 1988
GBK2K	GB 18030
CNS	CNS 11643
CNS1	CNS 11643 Plane 1
CNS2	CNS 11643 Plane 2
ETen	Big Five plus ETen extensions
ETenms	"ETen" with proportional glyphs for ASCII
ETHK	"HKscs" plus ETen extensions
HKgccs	Hong Kong GCCS
HKscs	Hong Kong SCS

Character Set Designator	Description
KSC	KS X 1001
KSCpc	"KSC" with proportional glyphs for KS X 1003 plus Macintosh extensions
KSCms	"KSC" with proportional glyphs for KS X 1003 plus Unified Hangul Code extensions

The table below lists the encoding designators based on existing Adobe CMap resources:

Encoding Designator	Description
UTF8	Unicode UTF-8 encoding form
UTF16	Unicode UTF-16 (UTF-16BE) encoding form
UTF32	Unicode UTF-32 (UTF-32BE) encoding form
UCS2	Unicode UCS-2 encoding form *(deprecated)*
EUC	Extended Unix Code
RKSJ	Shift-JIS encoding
B5	Big Five encoding
Johab	Johab encoding
UHC	Unified Hangul Code

If the encoding is 7-bit ISO-2022, the encoding value is omitted. Also, the actual encoding ranges used for EUC and RKSJ may differ depending on the character collection or supported character set. The writing mode must be either H or V, and should agree with the /WMode dictionary entry in the CMap resource itself. A -V CMap resource should contain only those character code to CID mappings that are specific to the vertical writing mode, and should also specify the *usecmap* operator to indicate from which CMap resource to inherit its character code to CID mappings.

The CMap resources developed by Adobe are unique in that their names do not contain, as a prefix, the string that corresponds to the /Registry string, because they are intended to be public, and thus used by any font developer. Developers other than Adobe are required to register a short /Registry string that should be prefixed to their CMap resource names (/CMapName), regardless of the character collection on which they are based. /Registry strings can be registered through Microsoft.* The purpose of this policy is to avoid duplicate CMap resource names.

* http://www.microsoft.com/typography/links/vendorlist.aspx

Appendix A: CMap Resource Compiler/Decompiler

Attached to this document is a tool, written in Perl and named *cmap-tool.pl*, that can be used to compile or decompile CMap resources. It assumes a complete CMap resource as input. This tool takes a CMap resource from standard input (STDIN), and outputs a CMap resource to standard output (STDOUT). In addition to sorting the character codes, collapsing single mappings with contiguous character codes and contiguous CIDs into range mappings, and separating the mapping lines into groups of 100, statistics, such as

- the total number of input character codes and mappings;
- the total number of collapsed mappings (if any);
- the total number of output character codes and mappings; and
- the average number of character codes per mapping

are written to standard error (STDERR).

When this tool detects that it is processing a UTF-32 CMap resource (the /CMapName string includes "UTF32"), it automatically generates matching UTF-8 and UTF-16 CMap resources. In other words, there is no need to separately develop UTF-8 and UTF-16 CMap resources, at least when using this tool, because they will be created automatically when compiling the UTF-32 one.

This tool also includes an "-e" command-line option, which, when invoked, expands all range mappings into a single block of single mappings. In effect, this command-line option reverses this tool's default behavior of collapsing single mappings with contiguous character codes and contiguous CIDs into range mappings.

Other miscellaneous error detection is automatically performed, such as if a character code is outside the specified code space range definition.

Author

Dr. Ken Lunde, Senior Computer Scientist, CJKV Type Development, Adobe Systems Incorporated

Publishing Date

March 14, 2012

Better by Adobe™

Adobe Systems Incorporated • 345 Park Avenue, San Jose, CA 95110-2704 USA • *www.adobe.com*