

Halftone Data Format (HDF) Specification
Version: 1.0b
Author: Stephen Herron
Company: Isis Imaging Corporation
Website: www.Isisimaging.com
Date: 15 May 2006

Halftone Data Format Specification

Introduction

The Halftone Data Format (HDF) enables the interchange, data storage, portability and reuse of halftone data by specifying an image file format for encoding halftone images. HDF describes halftone images in bi-level, and multi-bit levels in grayscale, CMYK color spaces and n-color spaces. The specification allows the capture and combination of multiple types of screens from various sources on a per-separation basis.

HDF consists of two PostScript compatible formats. It is useful in PostScript workflows and is applicable when PostScript is converted to other page description languages (PDLs). HDF is compatible with applications, drivers and controller RIPs that support the Adobe Systems' EPS and Quark DCS formats.

This specification is for developers of halftone screens who require a common encoding format. However, HDF allows for the inclusion of private or special-purpose information as comments.

The format encodes images described as bitmaps. It is not a PostScript halftone dictionary. Such dictionaries describe screen angle, frequency, and halftone pre-processing information such as *settransfer* operators. This information is not applicable after the image is in its halftone state.

History

Developed by Isis Imaging Corporation and Adobe Systems, Inc. in September 1997 to solve PostScript compatibility issues concerning the encoding of halftone screens and to encode halftones generated at the creation and prepress stages of photography and publication production. The format is useful for the archival of halftone images and Read-Once-Output-Many, (ROOM) architectures. Since its inception, HDS has been used on various types of RIPs and sent to various types of output devices. It is used in many workflows requiring repository of the halftone phase.

Document formatting conventions

Text is in san serif typeface
Code and headers are in Courier typeface

Language Notes

Following words are used to define the *significance* of each particular requirement. These words are used in accordance with their definitions in RFC 2119.

- *Must* and *required*. Means that the item is an absolute requirement for the specification.
- *Should* and *recommended*. Means that there may exist valid reasons to ignore this item, but the full implications should be understood before removal.
- *May* and *optional*. Means that this item is truly optional. It enhances the communication of the data.

Normative references

EPS: Encapsulated PostScript Format Specification 3.0, Adobe Systems Inc.

DCS: The Desktop Color Separation image file format. The Quark Desktop Color Separation Specification 2.0, May 1995. Quark Inc.

Definitions

Byte order. Big-endian refers to the most significant byte stored at the memory location with the lowest address, the next byte in significance is stored at the next memory location and so on. Little-endian refers to the least significant byte first.

Interlace, Interleave. Alternating between primary colors.

Separation. Planes or channels of primary colors

Device dot. The smallest mark made by a device on a substrate, usually paper.

Opaque. The data in the form of bits describing device dots in a plane replaces any underlying data. When a stack of graphic objects is flattened the top most object on the list replaces all objects coinciding with the objects coordinates. This the appropriate method of combining halftoned object.

Transparent. All of the graphic objects in a stack contributes to the result. The colors are blended. Blending must occur before halftoning and creating a HDS.

Format control and Support

The most recent version of the HDF specification is available in PDF format on the www.isimaging.com website.

Submitting a revision proposal: Any person or group that wants to propose a change or addition to the HDF specification should prepare a proposal that includes the following information:

- Name of the person or group
- The reason for the request
- The charges and/or additions
- Discussion of the impact on the installed base.
- A list of contacts supporting your position.

Please send your proposal to support@isisimaging.com

Specification

Overview

This specification maintains agreement with standard file formats Encapsulated PostScript, EPS and Desktop Color Separation, DCS Type 1 and DCS Type 2.

This format requires a halftone writer and a halftone reader. The reader and writer may be an source of raster such as a camera or scanner, page description language originator, a printer driver, or printer controller.

Halftone Writer. Any bit-depth data conversion process where the output is a halftone screen.

Halftone Reader. Any data conversion process where the halftone data is converted to marks on paper.

Halftone screen is an opaque paradigm and an alpha channel or blending is not appropriate.

Format identity

The file extension for HDF must be either **.dcs** for Desktop Color Separation or **.eps** for Encapsulated PostScript format. HDF is not a file extension.

General HDF Document structure

1. Header. *%ASCII*
2. Preview, *%32-bit RGB image*
3. A PostScript clipping path *%See PostScript Language Reference Version 3.*
4. PostScript image setup code *%ASCII*
5. Image halftone data
 - a. Grayscale
 - i. *1 plane of data (one 1-dimensional array)*
 - b. CMYK color
 - i. *Separation. (four 1-dimensional arrays)*
 - ii. *Interleave. (One 4-dimensional array)*

Compression

CCITT group 3.1, 1-dimensional Modified Huffman run-length encoding may be used for bi-level halftone screens.

Type = SHORT *%Compression is not recommended for 2- and 4-bit halftones.*

Data Description

The following data types are supported:

- BINARY. 1-bit unsigned integer in little-endian order.
- BYTE. 8-bit unsigned integer

Negative values are not supported. No padding or NULL place-holders are supported.

Blank separations describe white, (1.0) or black (0.0).

All halftone formats must use an opaque marking paradigm. For the PostScript language use the 'image' operator. The 'imagemask' operator is transparent and cause interference patterns when overlapping other screens. The layers of halftone screens follow the opaque-stack method with top layers overwriting lower layers.

- A bi-level grayscale halftone image contains black marks and no marks.
0 = no mark or white, 1 = mark or black
- A 2-bit screen consists of three densities and no mark
00 = no mark or white, 01 = first gray, 10 = second gray, 11 = black
- A 4-bit screen consists of 15 densities and no mark.
0000 = no mark and white, through thirteen densities and 1111 = black

Rows (scan lines) and Columns

The image is organized as a rectangular array of device dots. The dimensions are stored in image length, columns, first followed by image depth, rows. Columns are the run-length of the image. Rows are the scan lines of the image. Rows per unit must equal columns per unit.

Coordinate transformations

PostScript Coordinate Transformation Matrix (CTM) operations of rotate, offset, scale or skew do not apply to halftone screens. Halftone screens may be tiled and joined with *moveto* PostScript operators.

Resolution

Number of device dots per unit measure:

- Inch
 - Default is inch
- Centimeter

The location of data in the file is described according to the EPS and DCS format specifications

Data should be described in planes, bands and chunks. Strips and chunks are tiled with a CTM. Chunks are discouraged due to the complexity of CTM offsets for accurate alignment.

Description of planes, rows and chunks:

- Rows (scan lines) and columns (run length) per plane
- Rows (scan lines) per bands
- Rows (scan lines) and columns (run length) per chunk

All bands and chunks must have the same resolution and bit-depth or each image but may change by integer multiples of the device's resolution.

Grayscale images

Planer separation

CMYK Images

- Interleave = CMYK CMYK CMYK CMYK CMYK
- Planar separations = CCCC... MMMM... MMMM... KKKK...

N-color images. Any number of colors are supported as specified in the DCS and EPS header comments.

- Interleave = CMYKXX' CMYKXX' CMYKXX' CMYKXX' CMYKXX'
- Planar separations = CCCCCC... MMMMMM... YYYYYY... KKKKKK... XXXXXX...
X'X'X'X'X'X'...

There is no limit on the amount of data per plane, band or chunk. However, the halftone reader or writer may have a data amount limitation.

General DCS 1 and 2 format

Supports N-colors.

The format encodes images described as halftones. It does not contain halftone dictionary descriptions such as screen angle or frequency, or pre-processing information such as 'settransfer' operators.

Header structure. 8-bit byte containing a 7-bit ASCII code. (All of the following items must be included except where indicated.)

See the Desktop Color Separation image file format. The Quark Desktop Color Separation Specification 2.0, May 1995. Quark Inc. for information regarding Type 1 and Type 2 formats.

The longest PostScript line is 255 ASCII characters.

General Document structure

- Header
- Preview *%may be placed according to DCS normative reference
%+(not shown in the following example)*
- A PostScript clipping path *%See PostScript Language Reference Version 3*
- Halftone encoded in separation *% Beginning of CMYK*

N-color DCS Example

```
%!PS-Adobe-3.0 EPSF-3.0
%%Creator: Application name Version #
%%Title: Doc.dcs
%%CreationDate: day/month/year time
%%BoundingBox: 0 0 0 0 % x axis and then y axis; res/72 trunc
%%HiResBoundingBox: 0.00 0.00 0.00 0.00 % same but to 1/100 accuracy
%%DocumentData: 1, 2, 4
%%SupressDotGainCompensation
%%DocumentProcessColors: Cyan Magenta Yellow Black % List colors here
%%PlateFile: (Cyan) #offset size % Offset from beginning of file to the separation
%%PlateFile: (Magenta) # offset size% offset is in bytes.(header, DCS comments & clip path)
%%PlateFile: (Yellow) # offset size %+ determined by finding the smallest offset
%%PlateFile: (Black) # offset size %+ between "%!PS-Adobe and %%Trailer
% Size is in bytes of separation file including
%+ PostScript, clipping path and image data
%+ offset and size fields represented by decimal numbers
%+ therefore the last offset + size = entire file size

%%EndComments
%%BeginProlog
%%EndProlog
%%BeginSetup
%%EndSetup
gsave
1.00 dup scale % This number must be 1.00 since scaling is not supported

%----- Beginning of clipping path

%!PS-Adobe-3.0 EPSF-3.0
%%CreationDate: day/month/year time
%%Title: (Clipping path name)
%%BoundingBox: 0 0 0 0
%%HiResBoundingBox: 0.0 0.0 0.0 0.0
%%EndComments
%%EndProlog
%%BeginClippingPath
/m{moveto}def /l{lineto}def
/c{curveto}def /C /c load def
```

```

/y{pl 2 copy curveto}def /Y /y load def
/v{currentpoint 6 2 roll pl curveto}def
/pl{transform 0.25 sub round 0.25 add exch 0.25 sub round 0.25 add exch
itransform}def
/L /l load def /n{}def
0.0 0.0 m
0.0 0.0 l
0.0 0.0 l
0.0 0.0 l
n
0.0 0.0 m
0.0 0.0 l
0.0 0.0 l
0.0 0.0 l
0.0 0.0 l
n
4{gsave{flattenpath eoclip} stopped{grestore 2 mul dup setflat dup 200
ge {pop pop exit} if
}{
grestore setflat exit} ifelse
} loop eoclip
%%EndClippingPath           % End Clipping path PS code.
                              %+ Return to the PS code from application name

20 dict begin
/l {load} def
/level systemdict /languagelevel known def
/rows 0 def                 % from scaled image doc
/cols 0 def                 % from scaled image doc
/res 0 def                 % from scaled image doc
/picstr cols 8 div ceiling cvi string def
/Dasr {currentfile picstr readstring pop} def

cols res div 72 mul rows res div 72 mul scale % Changes total pixels to inches.
                                                 %+Images mapped at 0% and must be scaled to size
/select level {/image 1 def} {{begin
                                                %for 1-bit binary halftone
                                                 %+change to 2, 4, and 8 bits

Width Height BitsPerComponent ImageMatrix
/DataSource 1 image end} def} ifelse
/dopic {
select
} def

/ImageType 1 def
/Width cols def
/Height rows def
/ImageMatrix [cols 0 0 rows neg 0 rows] def
/BitsPerComponent 1 def
/Decode [0 1] def
/DataSource {Dasr} def
currentdict
%%BeginData: 1068 Binary Bytes
dopic
                                     % Place binary of color plane here
%%EndData
end
grestore

```

%%Trailer

% begin next colors, etc.

General EPS format

Supports grayscale and CMYK inksets

Header structure. 8-bit byte containing a 7-bit ASCII code. This format does not contain halftone dictionary descriptions containing screen angle or frequency, or pre-processing information such as 'settransfer' operators.

The format encodes images described as halftones. It does not contain halftone dictionary descriptions such as screen angle or frequency, or pre-processing information such as 'settransfer' operators.

Header structure. 8-bit byte containing a 7-bit ASCII code. *(All of the following items must be included except where indicated.)*

The longest PostScript line is 255 ASCII characters.

General Document structure

- Header
- Preview *%may be placed according to DCS normative reference
%+(not shown in the following examples)*
- A PostScript clipping path *%See PostScript Language Reference Version 3*
- Halftone encoded in interleaved *% CMYK data*
- Halftone encoded in planer *% Grayscale data*

CMYK EPS Document

```
!PS-Adobe-3.0 EPSF-3.0
%%Creator: Application name Version #
%%Title: Doc.EPS
%%CreationDate: day/month/year time
%%Copyright %May be included
%%BoundingBox: 0 0 0 0
%%HiResBoundingBox: 0.0 0.0 0.0 0.0
%%ImageWidth % X resolution must equal Y resolution
%%ImageLength
%%Planar Configuration %May be included
%%ColorSequence
%%NumberOfinks
%%halftoneBitsPerSample
%%Compression %Should be included
%%SuppressDotGainCompensation
%%DocumentProcessColors: Cyan Magenta Yellow Black
%%Private comments: %Comments that are appropriate for the output device
%%EndComments
%%BeginProlog
%%EndProlog
%%BeginSetup
%%EndSetup
gsave
1.00 dup scale % This number must be 1.00 since scaling is not supported
```

```

%----- Beginning of clipping path

%!PS-Adobe-3.0 EPSF-3.0
%%CreationDate: day/month/year time
%%Title: (Clipping path name)
%%BoundingBox: 0 0 0 0
%%HiResBoundingBox: 0.0 0.0 0.0 0.0
%%EndComments
%%EndProlog
%%BeginClippingPath
/m{moveto}def /l{lineto}def
/c{curveto}def /C /c load def
/y{pl 2 copy curveto}def /Y /y load def
/v{currentpoint 6 2 roll pl curveto}def
/pl{transform 0.25 sub round 0.25 add exch 0.25 sub round 0.25 add exch
itransform}def
/L /l load def /n{ }def
0.0 0.0 m
0.0 0.0 l
0.0 0.0 l
0.0 0.0 l
n
0.0 0.0 m
0.0 0.0 l
0.0 0.0 l
0.0 0.0 l
0.0 0.0 l
n
4{gsave{flattenpath eoclip} stopped{grestore 2 mul dup setflat dup 200
ge {pop pop exit} if
}{
grestore setflat exit} ifelse
} loop eoclip
%%EndClippingPath           % End Clipping path PS code.
                           %+ Return to the PS code from application name

50 dict begin % temp dict for EPS definitions

  /colorimage where {           % colorimage is defined
    pop
    /paintimage {
      {currentfile cyanstr readstring pop}
      {currentfile magentastr readstring pop}
      {currentfile yellowstr readstring pop}
      {currentfile blackstr readstring pop}
      true 4 colorimage
    } bind def
  }{
    %colorimage not defined in B&W environment

    /paintimage {
      {currentfile cyanstr readstring pop pop
      currentfile magentastr readstring pop pop
      currentfile yellowstr readstring pop pop
      currentfile blackstr readstring pop pop}
      image
    } bind def
  } ifelse

```

```

%%EndProlog

/rows 0 def           %from scaled image doc
/cols 0 def           %from scaled image doc
/res 0 def            %from scaled image doc

0 setgray

/cyanstr 0 string def
/magentastr 0 string def
/yellowstr 0 string def
/blackstr 0 string def

currentdict

cols res div 72 mul rows res div 72 mul scale
cols rows 1 [cols 0 0 rows neg 0 rows]   %for 1-bit binary halftone
                                           %+change to 2, 4, and 8 bits
%%BeginData: 0 Binary Bytes             %insert number of bytes here
paintimage

%----- Place binary here. Color planes in CMYK interleaved order

%%EndData
grestore
%%Trailer
end
%---Place an ASCII return here
%EOF                                     %End of file

```

EPS Monochrome image

```

%!PS-Adobe-3.0 EPSF-3.0
%%Creator: Application name Version
%%Title: Doc.eps
%%CreationDate: day/month/year time
%%BoundingBox: 0 0 280 429                % x axis and then y axis; res/72 trunc
%%HiResBoundingBox: 0 0 280.00 429.00    % same but to 1/100 accuracy
%%DocumentData: Binary
%%SupressDotGainCompensation
%%DocumentCustomColors: (Black)
%%Extensions: CMYK
%%EndComments
%%BeginProlog
%%EndProlog
%%BeginSetup
gsave
1.00 dup scale      % This number must be 1.00 since scaling is not supported

```

%----- Beginning of clipping path

```

%!PS-Adobe-3.0 EPSF-3.0
%%CreationDate: day/month/year time
%%Title: (Clipping path name)
%%BoundingBox: 0 0 0 0
%%HiResBoundingBox: 0.0 0.0 0.0 0.0
%%EndComments

```

```

%%EndProlog
%%BeginClippingPath
/m{moveto}def /l{lineto}def
/c{curveto}def /C /c load def
/y{pl 2 copy curveto}def /Y /y load def
/v{currentpoint 6 2 roll pl curveto}def
/pl{transform 0.25 sub round 0.25 add exch 0.25 sub round 0.25 add exch
ittransform}def
/L /l load def /n{ }def
0.0 0.0 m
0.0 0.0 l
0.0 0.0 l
0.0 0.0 l
n
0.0 0.0 m
0.0 0.0 l
0.0 0.0 l
0.0 0.0 l
0.0 0.0 l
n
4{gsave{flattenpath eoclip} stopped{grestore 2 mul dup setflat dup 200
ge {pop pop exit} if
}{
grestore setflat exit} ifelse
} loop eoclip
%%EndClippingPath           % End Clipping path PS code.
                             %+ Return to the PS code from application name

%%EndSetup
/l {load} def
/level systemdict /languagelevel known
{languagelevel 2 ge} {false} ifelse def
/rows 0 def                    %from scaled image doc
/cols 0 def                    %from scaled image doc
/res 0 def                    %from scaled image doc
/picstr cols 8 div ceiling cvi string def
/Dasr {currentfile picstr readstring pop} def

/pictdim {cols rows 1 [cols 0 0 rows neg 0 rows]} def
                             %for 1-bit binary halftone
                             %+change to 2, 4, and 8 bits
cols res div 72 mul rows res div 72 mul scale

/paintimage level
  {{pictdim {Dasr} image}def}
  {{pictdim {Dasr} true 1 colorimage} def}
  ifelse

currentdict
%%BeginData: 0000  Binary Bytes % physical data in bytes
paintimage

%----- Place binary here
%%EndData
grestore
%%Trailer
end
%----Place an ASCII return here
%%EOF                          %End of file

```

