



OPI™

**Open Prepress Interface
Specification 1.3**

22 September 1993

Copyright

© 1989-1993 Aldus Corporation. All rights reserved.

Trademarks

Aldus, the Aldus logo, and PageMaker are registered trademarks and TIFF and OPI are trademarks of Aldus Corporation. PostScript is a registered trademark of Adobe Systems Inc. and all references to PostScript in this document are references to either the PostScript interpreter or language.

This document is provided for developers intending to implement OPI conventions.

Aldus welcomes your feedback. If you have comments or suggestions regarding OPI, please contact:

Developer's Desk
Aldus Corporation
411 First Avenue South
Seattle, Washington 98104-2871
Tel: (206) 628-6593
Fax: (206) 343-4240
Internet: dev-desk@aldus.com

Printed in U.S.A.
Part Number 993-798

Table of Contents

Changes from Version 1.2	4
Introduction	5
How OPI works	5
The Details	6
Terminology	6
General Requirements	6
References	7
The OPI Comments	7
Example 1	13
Example 2	14
Appendix A: Creating a PostScript Transformation Matrix	15

Changes from Version 1.2

- New: %ALDImageCropFixed gives partial-pixel cropping information.
- New: %ALDImageColorType describes whether an applied color is Spot or Process, for B&W and grayscale images.
- New: %ALDImageTint contains the tint percentage for Spot colors, for B&W and grayscale images.
- New: %ALDImageOverprint gives the knockout/overprint setting for B&W and grayscale images.
- New: %ALDImageAsciiTagNNN displays the contents of all ASCII tags in a TIFF file, for additional pass-through flexibility.
- %ALDImageID is recommended again, to simplify the task of finding the correct high resolution file.
- Guidelines for clipping to an arbitrary shape have been included. See the end of the section on %ALDImageCropRect.
- The examples have been updated to use the new comments.
- The notation has been updated to conform to the Adobe Document Structuring Conventions, Version 3.0.
- Support for %%BeginData/%%EndData has been added.
- A number of sections have been clarified.

Introduction

The Open Prepress Interface (OPI) is a collection of PostScript-language comment conventions that allows a page-layout program to use low or medium resolution TIFF images for layout and proofing, and have a prepress system or OPI server automatically substitute a high resolution TIFF or other image when the final film or plates are generated. Both desktop prepress software and high-end prepress systems can use OPI comments to minimize network traffic and image storage requirements.

How OPI works

OPI comments describe the placement and size of scanned images, as well as cropping information and any adjustments to the size, brightness, or contrast.

For example, when you lay out a publication in Aldus PageMaker, you use the standard tools to place, re-size, and crop scanned images (usually, TIFF files that are lower resolution versions of a high resolution scan). When the layout is complete and you are ready to print separations, you print a composite PostScript file to disk using PageMaker's "For separation" or "EPS" option. PageMaker embeds the OPI comments (specifying the name and position of each scanned image, cropping information, and any adjustments to the size, brightness, or contrast) in the PostScript file. This file is comparatively small since it does not normally include any scanned image data.

The PostScript file can then be sent to a prepress system or OPI server where the OPI layout information is extracted and used to find the high resolution image and insert it the image data into the PostScript print stream to the imagesetter.

OPI does not require that the image be pre-separated. Conversion from RGB to CMYK can take place "on the fly" during the final printing process, if the OPI Server is able to perform such conversions. This approach has two advantages: it requires much less disk space, since a high-resolution CMYK version of the image is never stored on disk; and it gives greater separation flexibility, since it allows the image to be stored in the more device-independent RGB form during the entire prepress cycle.

The PostScript file containing OPI comments should be either an EPS file or the multi-page equivalent of an EPS file (see PageMaker's "For separation" print option.) As

such, it should be a composite color file, not a separated file. The reasons for discouraging the use of OPI within the context of separated files are:

- There are no Adobe conventions covering the use of separated PostScript files as an interchange format, and currently every application does it differently.
- In theory, PostScript Level II RIPS will be where we ultimately do our separations. To make this happen, we will be feeding these RIPS composite, not separated, PostScript streams.
- Trapping programs like Aldus TrapWise can operate much more efficiently by operating on composite PostScript streams instead of separated streams, since a composite stream is much smaller, the resulting display list is much smaller, and the output frame buffer can be created in one pass.

If you must use separated files, be sure to at least follow the existing de facto %%PlateColor convention, so that the OPI consumer can at least know what the current separation is. (To create separations directly from Aldus PageMaker 5.0, choose "Separations" in the Print/Color dialog.)

The Details

This section describes the OPI comments and defines in detail the requirements for using these comments.

Terminology

In the following discussion, we call the creator of the PostScript file the “page-layout program.” We call the reader of the OPI comments in the PostScript file the “OPI server.” The “OPI server” can be any of the following: an OPI-capable print server, running on a file server; a high end prepress workstation; a specially modified PostScript RIP; or any software package that can read OPI comments and substitute high-resolution image data, such as Aldus PrePrint.

A scanned image that is placed (imported) into a page-layout (or illustration) program may be a lower-resolution version of a high resolution image that was scanned on an OPI server or color prepress system, or it may be a final, full resolution image that simply needs to be separated into CMYK by a separation program such as Aldus PrePrint.

We will use the term “bitmap” to refer to any 1-bit image, usually a 1-bit TIFF file or MacPaint file. These are also called black-and-white images, or bi-level images. A “grayscale” image has pixels that are shades of gray, and are generally 8 bits deep. OPI allows bitmap and grayscale images to have a color other than black applied to them. (See %ALDImageColor and related comments, below.)

General Requirements

OPI was designed for an environment in which the PostScript file conforms to the following standards:

Adobe Systems’ Document Structuring Conventions

Adobe Systems’ Document Structuring Conventions describe general information about the publication, such as the name of the publication, the number of pages, and the size of each page. See Appendix G, “Document Structuring Conventions—Version 3.0,” in the [PostScript Language Reference Manual](#), Second Edition. In this specification, we will use the abbreviation “DSC” to refer to the Document Structuring Conventions. (Do not get this confused with “DCS”, which is similar to CMYK

TIFF but stores the data in 5 files instead of 1, in a PostScript-specific format.)

The BNF notation used in the DSC 3.0 document is also used in this specification.

Adobe Systems’ Color Separation Conventions

Adobe Systems’ Color Separation Conventions define guidelines and color pseudo-operators for use in color separation environments. Refer to Adobe Systems’ “Proposal for Color Separation Conventions for PostScript Language Programs,” Technical Note #5044, dated 12/14/89, for further information.

One set of OPI comments per image

One set of OPI comments should either replace each scanned image or encapsulate the scanned-image data. The OPI comments specify the name and position of each scanned image, as well as high resolution links, cropping, and adjustments to size, rotation, color, and other attributes.

PostScript setup before the OPI comments

Each set of OPI comments should be preceded by the PostScript **save** operator and followed by the **restore** operator. This allows all VM used by the imaging code to be recovered and the graphics state to be restored.

See Section H.3.2 in the Adobe red book for further suggestions, since the situation is similar to that of importing and printing an EPS file.

Ignore unknown comments.

An OPI server should ignore comments that it does not recognize, so that it can continue to process documents that contain newer OPI comments. In this way, OPI can be expanded to meet industry demands without requiring that all OPI servers be immediately updated when a new page layout software version is released.

It is also possible that enumerated values may be added to existing OPI comments at some point in the future, so parsing software should be prepared to encounter an unexpected value and respond appropriately. (The best strategy is probably to assume the default value.)

Defaults

If a default is listed for an OPI comment, then the OPI reader must assume that default value if that OPI comment does not exist.

References

1. Appendix G, "Document Structuring Conventions—Version 3.0," in the PostScript Language Reference Manual, Second Edition, Adobe Systems Incorporated. The PostScript Language Reference Manual is often referred to as "the red book."
2. "Proposal for Color Separation Conventions for PostScript Language Programs," Adobe Systems Technical Note #5044, dated 12/14/89.
3. TIFF Revision 6.0, Aldus Corporation, June 3, 1992. (Call 206.628.6593 to order.)

The OPI Comments

Here are the OPI comments, listed in the order in which they should occur when describing an image.

%ALDImageFileName: <filename>

Required.

This comment records the name of the image (TIFF file) as it was imported into the page-layout program. Include the entire path name to the file, using the standard form for the operating system in which the page-layout program is currently running. Thus, Macintosh folder names will be separated by colons, DOS directories by backslashes, UNIX directories by slashes, and so on. For example, a %ALDImageFileName comment for a PostScript file from the Macintosh might look like:

```
%ALDImageFileName: HD40:Pubs:Pizza Ad:Pizza Picture.tif
```

The value of %ALDImageFileName can include embedded spaces without requiring parentheses around the string.

This entire comment line (%ALDImageFileName:, together with the path and filename) should not exceed 255 characters, the Adobe recommended maximum line length for any line in a PostScript file. If necessary, the comment can be extended using the standard "%%+" convention. (See section G.5.2 in Appendix G of the PostScript Language Reference Manual.)

Note: If the TIFF file is a low-resolution version of an image on an OPI Server, the name of the TIFF file should generally be sufficiently similar to the name of the image on the OPI Server to allow for an automatic match between the versions of the image, especially if %ALDImageID (see below) is not used.

%ALDImageID: <filename>

Optional.

An identifying string that an OPI Server wishes to use to uniquely identify the full-resolution version of the image as stored on the OPI Server.

This information is stored in the TIFF file as part of a reduced-resolution image, as Tag #32781 (0x800D). The TIFF data type is ASCII. See the TIFF 6.0 specification for information on TIFF data types and the structure of a TIFF file.

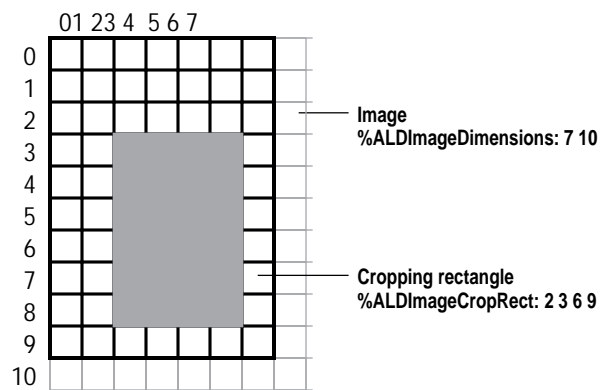
The value of %ALDImageFileName can include embedded spaces without requiring parentheses around the string. Any other printable ASCII character is legal.

This field can be used in addition to the %ALDImageFileName comment to identify the correct full-resolution image.

%ALDObjectComments: <text>

Optional.

This comment provides a way of sending suggestions to the OPI server operator on how the image should be



treated. Text can be any ASCII text and can be extended to multiple lines using the DSC “%%+” convention.

See also %ALDImageAsciiTagNNNNN, described below.

%ALDImageDimensions: <pixelwide> <pixelshigh>

<pixelwide> ::= <uint>

<pixelshigh> ::= <uint>

Required.

Pixelwide and pixelshigh give the dimensions of the placed TIFF file, in pixels. This information, together with the %ALDImageCropRect comment described below, allows the page-layout program to pass rectangular source-cropping information to the OPI server.

%ALDImageCropRect: <left> <top> <right> <bottom>

<left> ::= <uint>

<top> ::= <uint>

<right> ::= <uint>

<bottom> ::= <uint>

Required.

Left, top, right, and bottom describe the size of the cropping rectangle and its placement on the image. The coordinates are in pixels, where left = 0, top = 0, right = pixelwide, and bottom = pixelshigh describe the entire image. (Pixelwide and pixelshigh are the parameters for %ALDImageDimensions.) Frequently, %ALDImageCropRect is the whole image.

The coordinates describe the upper-left corner of a pixel. In other words, (left, top) are the coordinates of the upper-left corner of the upper-left pixel of the cropping rectangle, and (right, bottom) are the coordinates of the upper-left corner of the pixel that is *just outside* the lower-right corner of the cropping rectangle.

Thus, the difference between the right and left coordinates represents the width, in pixels, of the cropped area, and the difference between the bottom and top coordinates represents the height, in pixels, of the cropped area.

An OPI Server should assume the default value for %ALDImageCropRect to be the entire image.

Cropping allows a page-layout program to print only the portion of the image inside the cropping rectangle, thus reducing print time. You create a cropping rectangle in

the page-layout program. (In Aldus PageMaker, you use the cropping tool to define the cropping rectangle.) In page-layout programs that do not support true source cropping, the image-cropping rectangle is the entire image.

To use OPI to clip to an arbitrary shape, the page-layout program can set up a the required clipping path outside the OPI context, i.e., the clipping path should be set up before the first OPI comment for the image. Then when high-resolution data is inserted, it will be clipped automatically. An image may be both cropped (using %ALDImageCropRect) and clipped (using a clipping path).

%ALDImageCropFixed: <left> <top> <right> <bottom>

<left> ::= <real>

<top> ::= <real>

<right> ::= <real>

<bottom> ::= <real>

Optional.

Same as %ALDImageCropRect, but with floating point values, giving more accuracy in cases where the cropping rectangle may encompass partial pixels on one or more sides.

If this comment is not present, use the values in %ALDImageCropRect.

Use of %ALDImageCropFixed does not change the area covered by the image on the page; it only affects how the image is cropped and scaled to fit into the desired space on the page. In real-world jobs, the difference between using %ALDImageCropRect and %ALDImageCropFixed is either difficult or impossible to detect.

%ALDImagePosition: <LLx> <LLy> <ULx> <ULy> <URx> <URy> <LRx> <LRy>

<LLx> ::= <real>

<LLy> ::= <real>

<ULx> ::= <real>

<ULy> ::= <real>

<URx> ::= <real>

<URy> ::= <real>

<LRx> ::= <real>

<LRy> ::= <real>

Required.

LLx, LLy, ULx, ULy, URx, URy, LRx, and LRy specify where on the page the cropped portion of the image (see %ALDImageCropRect) is to be printed, and determine the scaling, rotation, flip, and skew of the image. (LLx,LLy) corresponds to the lower-left corner of the %ALDImageCropRect; (URx,URy) corresponds to the upper-right corner of the %ALDImageCropRect; and so on.

You specify these coordinates using the default PostScript user coordinate system—72 units per inch, origin at lower left, greater y values are up.

For example, if you want to print the image in the lower-left corner of a page at a size of 3 inches wide by 1 inch high, the %ALDImagePosition comment would look like this:

```
%ALDImagePosition: 0.0 0.0 0.0 72.0 216.0 72.0 216.0 0.0
```

Note that the combination of %ALDImageCropRect and %ALDImagePosition determines the scale factor in each dimension, (and rotation, flip, and skew,) since %ALDImagePosition specifies the location of all four corners.

Current Transformation Matrix. As mentioned earlier, you specify the coordinates for %ALDImagePosition using the default PostScript user coordinate system. Therefore, the page-layout program must ensure that the current transformation matrix (CTM) is set to the default PostScript coordinate system before it writes the OPI comments for the image. An OPI Server must be able to insert PostScript code that assumes that the default PostScript coordinate system is in effect.

initmatrix: If the page-layout program does not typically use the default PostScript coordinate system, it must not simply reset the coordinate system to the default coordinate system. In particular, it must not call **initmatrix** when creating the separation or EPS file. If it does, it will create a PostScript file that cannot be repositioned, re-scaled, or rotated. Instead, it must save the current transformation matrix that is in effect at the start of your PostScript stream (it will usually be the default coordinate system, but not always), and use that matrix to set the current coordinate system before it writes the OPI comments for an image.

Why four corners. To support rotation, flip, and skew, a 2D transformation must be specified either by using at least three points or by using a transformation matrix. OPI uses the three-point approach, for conceptual simplicity. A PostScript-type transformation matrix can be computed by solving a system of six linear equations in six

unknowns (actually, two sets of three equations in three unknowns), using any 3 of the 4 corner points. See the appendix at the end of this specification.

EPS files that contain scanned images. An illustration or page-layout program that obeys Adobe Systems' Encapsulated PostScript (EPS) file conventions (see Appendix H in the [Adobe PostScript Language Reference Manual](#)) may include OPI comments to describe a scanned image when creating an EPS file. When you place such an EPS file into a page-layout program, the %ALDImagePosition values may seem to be incorrect because they do not reflect positioning and sizing of the EPS file after placement. Nevertheless, the page-layout program must not attempt to update these comments! The OPI Server uses these comments without modification to insert high-resolution data into the PostScript stream at print time, and everything works as expected because the transformation matrix set up by the page-layout program for the EPS file handles all the positioning, sizing, rotation, and so on. This process is recursive, so that EPS files with OPI comments can be embedded in other EPS files which are in turn embedded in other EPS files, and so on.

%ALDImageResolution: <horizRes> <vertRes>

```
<horizRes> ::= <real>  
<vertRes> ::= <real>
```

Optional. No default.

These values specify the resolution of the original image, in pixels per inch. The resolution can be used to report to the user how much a particular image has been scaled. For example, if the %ALDImageCropRect comment indicates that the image is 1000 by 1000 pixels, and the %ALDImagePosition parameter positions the image in a 1" by 1" square, and the %ALDImageResolution parameter indicates that the original resolution was 500 pixels per inch, then the scale factor is .5; that is, the image has been scaled down by a factor of 2.0 in both dimensions.

%ALDImageColorType: Spot | Process | Separation

Optional. Applicable to bitmap and grayscale images only. Default is "Spot".

%ALDImageColorType gives the type of color for %ALDImageColor (see below).

"Separation" means that the image is to be printed on all separations. Typically this corresponds to a special color called "Registration" in the page-layout application.

In the “Separation” case, if %ALDImageTint (see below) is zero, then the image is to be knocked out of all separations. Typically this corresponds to a special color called “Paper” or “White” in the page-layout application.

When “Separation” is the ColorType, the CMYK values of %ALDImageColor are ignored.

%ALDImageColor: <C> <M> <Y> <K> <colorname>

<C> ::= <real>
 <M> ::= <real>
 <Y> ::= <real>
 <K> ::= <real>
 <colorname> ::= <text>

Optional. Applicable to bitmap and grayscale images only. Default is “0 0 0 1 (Black)”.

C, M, Y, and K are values between 0 and 1 and determine the percentages of cyan, magenta, yellow, and black ink to be used when printing the “black” pixels of this image.

colorname is a text string that names the color. Parentheses around colorName are recommended, and are required if blanks or special characters may be embedded the colorName.

A grayscale image is to be rendered in shades of %ALDImageColor. For example, suppose the color is Process red, with CMYK value = 0 1 1 0. Then the black pixels in the grayscale image will be print full red (0 1 1 0), the 50% gray pixels will print in a 50% tint of red (0 .5 .5 0), and the white pixels will have no ink at all applied (0 0 0 0). Seeing that the appropriate PostScript is generated to make this happen for the high-resolution image is the responsibility of the OPI server.

%ALDImageTint: <real>

Optional. Applicable to bitmap and grayscale images only. Default is 1.

Percentage of the ink or color (see %ALDImageColor) to be applied to this image, as a value between 0 (no ink) and 1 (solid).

The tint value should not be reflected in the CMYK values listed in %ALDImageColor. It is up to the OPI Server to multiply the CMYK values by the tint value to achieve the correct tinted color effect.

%ALDImageTint is valid for all 3 %ALDImageColorType types.

%ALDImageOverprint: false | true

Optional. Default is “false”.

Set to “true” if the image is to overprint underlying objects on other separations, and “false” if the image is to knock out underlying objects on other separations.

%ALDImageOverprint is applicable to bitmap and grayscale images only. (But %ALDImageColor need not be black.)

%ALDImageType: <samples> <bits>

<samples> ::= <uint>
 <bits> ::= <uint>

Optional.

Samples is the samples per pixel, and bits is the bits per sample at which the image was scanned. For example:

Image type	Samples per pixel	Bits per sample
1-bit (bitmap)	1	1
4-bit grayscale	1	4
8-bit grayscale	1	8
8-bit palette color	1	8
24-bit RGB color	3	8
32-bit CMYK color	4	8

An OPI Server can use %ALDImageType to give the user more complete feedback when relinking images.

%ALDImageGrayMap: <uint> ...

Optional. For bitmap and grayscale images only. The default is a linear ramp matching the PhotometricInterpretation value of the image (BlackZero or WhiteZero - see the TIFF 6.0 specification.)

`%ALDImageGrayMap` records changes made to the brightness or contrast of the image in the page-layout program.

This “lookup table” gives a 16-bit intensity (or reflectance) value for each possible pixel value, starting with pixel value 0 and ending with pixel value 255 (for 8-bit data), where 0 is black and 65535 is white. An application should invert the values to get the desired percent-dot when doing separations.

A bitmap will have two GrayMap values (corresponding to pixel values 0 and 1); a 4-bit grayscale image will have 16 values; and an 8-bit image will have 256 values.

Note that the `%ALDImageGrayMap` can be either ascending or descending, depending on whether the image is `BlackZero` or `WhiteZero`, respectively. (Note that this means that an OPI server should not change the image from `BlackZero` to `WhiteZero` or vice versa when creating a low-res image from a high-res image, otherwise `%ALDImageGrayMap` will be applied incorrectly.)

When you have a large number of values, such as for an 8-bit image, list them on multiple lines, using the `%%+` continuation line convention. Adobe recommends that there be no more than 255 characters per line in a DSC-conforming PostScript file.

`%ALDImageTransparency: true | false`

Optional. For bitmaps only. Default is "true".

This comment defines whether or not white pixels are to be rendered as transparent. Transparency can be achieved by using the PostScript “`imagemask`” operator instead of the “`image`” operator.

`ALDImageTransparency` is not applicable to grayscale or color images at this time, since PostScript currently has no mechanism for printing completely or partially transparent pixels in grayscale or color images. Instead, a clipping path must be used to mask off parts of grayscale and color images.

`%ALDImageAsciiTag<NNN>: <tagtext>`

`<NNN>` ::= `<uint>` (TIFF tag number, in decimal)
`<tagtext>` ::= `<textline>` (contents of TIFF field)

Optional.

A page-layout program can use this comment convention to pass TIFF ASCII tag values through to the OPI Server.

“`NNN`” is the TIFF tag number. It must not be padded on the left with zeros.

The value of `%ALDImageAsciiTagNNN` is the value of the corresponding TIFF ASCII field. The page-layout program should check for special (non-printing) characters such as carriage return and line feed and translate them into question marks or other printable characters. (Otherwise a new line would be created that the PostScript RIP would attempt to interpret as PostScript commands, causing the print job to fail. See the PostScript red book for further information on PostScript interpretation rules.) Lines longer than 255 characters should be broken into multiple lines using the “`%%+`” convention.

See the TIFF 6.0 specification for further information on TIFF.

`%%BeginObject: image` `%%EndObject`

Required.

`%%BeginObject` immediately follows the last OPI comment. If the page-layout program has included PostScript code for the image (Aldus PageMaker has an option for including image data or not), the data must begin after `%%BeginObject`. `%%EndObject` follows the image data. When printing, an OPI Server typically removes everything between `%%BeginObject` and `%%EndObject` (if there is anything to remove) and replaces it with its own data (and any necessary PostScript procedure definitions and invocations) for that image.

`%%BeginObject` and `%%EndObject` are documented in the Adobe DSC document.

`%%BeginBinary: <#ofBytes>` `%%EndBinary`

or, preferably, use the newer:

`%%BeginData: <#ofBytes> Binary Bytes` `%%EndData`

`<#ofBytes>` ::= `<uint>`

Required if image data is included in an OPI-compliant PostScript file.

If image data is included in the file, it must be surrounded by the %%BeginData/%%EndData comments. #ofBytes indicates the number of bytes between the %%BeginData and %%EndData comments. These comments allow the OPI server to scan quickly to the end of the image data. Note that the %%BeginData/%%EndData pair must be contained within the %%BeginObject/%%EndObject pair (see above).

Use of BeginBinary/EndBinary by page-layout programs is no longer recommended. Use the newer BeginData/EndData comments, which explicitly supports hex data as well as binary data.

See the Adobe DSC document for further information on %%BeginData/%%EndData, such as the proper syntax for writing out hex data. An OPI Server must support hex as well as binary image data.

Example 1

This example shows the OPI comments describing two scanned images: an 8-bit grayscale image and a 24-bit color image.

The image data is not included in this example.

```
<PostScript header>
<text and graphics>
<save>
<set up PostScript default space>
%ALDImageFileName: theDisk:Graphics:Images:TIF - gray:DigitalDarkroom:Barn.Tiff(8bit)(LD2)
%ALDImageDimensions: 356 290
%ALDImageCropRect: 0 0 356 290
%ALDImageCropFixed: 0.0 0.0 356.0 290.0
%ALDImagePosition: 353.300 571.250 353.300 738.000 558.000 738.000 558.000 571.250
%ALDImageResolution: 150.000 150.000
%ALDImageColorType: Process
%ALDImageColor: 0.00 0.00 0.00 1.00 (Black)
%ALDImageTint: 1.00
%ALDImageOverprint: false
%ALDImageType: 1 8
%ALDImageGrayMap: 65535 65273 65011 64749 64486 64290 64028 63831 63569 63307 63045 62783 62520 62258 61996 61734
%%+ 61537 61275 60948 60685 60489 60227 59965 59702 59375 59113 58850 58588 58326 57998 57736 57474
%%+ 57278 57015 56688 56426 56229 55967 55705 55443 55180 54918 54656 54394 54132 53935 53739 53477
%%+ 53214 52952 52690 52428 52166 51904 51642 51379 51183 50921 50724 50462 50200 49938 49676 49413
%%+ 49151 48889 48627 48430 48168 47841 47578 47382 47120 46858 46595 46268 46006 45743 45481 45219
%%+ 44891 44629 44367 44171 43908 43581 43319 43122 42860 42598 42336 42073 41811 41549 41287 41025
%%+ 40828 40632 40370 40107 39845 39583 39321 39059 38797 38535 38272 38076 37814 37617 37355 37093
%%+ 36831 36569 36306 36044 35782 35520 35323 35061 34734 34471 34275 34013 33751 33488 33161 32899
%%+ 32636 32374 32112 31784 31522 31260 31064 30801 30474 30212 30015 29753 29491 29229 28966 28704
%%+ 28442 28180 27918 27721 27525 27263 27000 26738 26476 26214 25952 25690 25428 25165 24969 24707
%%+ 24510 24248 23986 23724 23462 23199 22937 22675 22413 22216 21954 21627 21364 21168 20906 20644
%%+ 20381 20054 19792 19529 19267 19005 18677 18415 18153 17957 17694 17367 17105 16908 16646 16384
%%+ 16122 15859 15597 15335 15073 14811 14614 14418 14156 13893 13631 13369 13107 12845 12583 12321
%%+ 12058 11862 11600 11403 11141 10879 10617 10355 10092 9830 9568 9306 9109 8847 8520 8257
%%+ 8061 7799 7537 7274 6947 6685 6422 6160 5898 5570 5308 5046 4850 4587 4260 3998
%%+ 3801 3539 3277 3015 2752 2490 2228 1966 1704 1507 1311 1049 786 524 262 0
%%BeginObject: image
%%EndObject
<restore>
<possibly some more PostScript text and graphics>
<save>
<set up PostScript default space>
%ALDImageFileName: theDisk:Graphics:Images:TIF - RGB:Flamingo.TIF(LD)
%ALDImageDimensions: 416 410
%ALDImageCropRect: 0 0 416 410
%ALDImageCropFixed: 0.0 0.0 416.0 410.0
%ALDImagePosition: 72.000 69.250 72.000 479.250 488.000 479.250 488.000 69.250
%ALDImageResolution: 150.000 150.000
%ALDImageType: 3 8
%%BeginObject: image
%%EndObject
<restore>
<possibly some more PostScript text and graphics>
<end-of-job stuff>
```

Example 2

In this example, the image data has not been stripped out (the image is small—only 4 by 4 pixels). Note especially the use of %%BeginData and %%EndData.

```
<start save context>
<set up default coordinate space>
%ALDImageFileName: theDisk:Graphics:Images:TIFF - gray:All16.tif
%ALDImageID: /home/images/gray/All16.tif
%ALDImageDimensions: 4 4
%ALDImageCropRect: 0 0 4 4
%ALDImageCropFixed: 0.0 0.0 4.0 4.0
%ALDImagePosition: 235.400 336.400 235.400 484.900 383.900 484.900 383.900 336.400
%ALDImageResolution: 72.000 72.000
%ALDImageColorType: Spot
%ALDImageColor: 1.000 0.100 0.100 0.000 (Aquamarine)
%ALDImageTint: 0.20
%ALDImageOverprint: true
%ALDImageType: 1 4
%ALDImageGrayMap: 0 4260 8847 13107 17367 21954 26214 30474 35061 39321 43581 48168 52428 56688 61275 65535
%ALDImageAsciiTag270: This is a really small test file.
%ALDImageAsciiTag271: Acme Scanner Company
%ALDImageAsciiTag272: BigScanner 1.0
%%BeginObject: image
<parameters for a PostScript procedure go here>
<ours happens to be named "S_IMAGEV3">
%%BeginData: 30 Hex Bytes
S_IMAGEV3
FEDC
BA98
7654
3210
%%EndData
%%EndObject
<restore the save context>
```

Appendix A: Creating a PostScript Transformation Matrix

```
/* Function to determine PostScript matrix necessary to transform default user coord. space (DUCS) into a space that
represents transformed image.
   Knowns:      - 4 corner coords. of transformed image in DUCS
                 - Image width & height
   To determine: PS matrix
   Notes:      This assumes you want to transform the DUCS independent of the image coord. space,
allowing the matrix used in the invocation of the PS 'image' operator to represent the mapping between the unit square
of user space to the boundary of the image in image space (see sec. 4.10 of the new Red Book or 4.7 of the old).
*/
void GetImageMatrix (llx, lly, ulx, uly, urx, ury, lrx, lry, w, h, matrix)
double llx, lly, ulx, uly, urx, ury, lrx, lry, w, h;
char *matrix;
{
  /* We need 6 linear equations to solve for 6 unknowns - args. of PS matrix,
  a, b, c, d, tx and ty. 'p' points in equations below represent transformed, OPI points passed in, while non-p x,y
  pairs represent coords. of DUCS image rectangle.

  1) x1p = ax1 + cy1 + tx
  2) y1p = bx1 + dy1 + ty
  3) x2p = ax2 + cy2 + tx
  4) y2p = bx2 + dy2 + ty
  5) x4p = ax4 + cy4 + tx
  6) y4p = bx4 + dy4 + ty

  Note: can skip 4th point as we only need equiv. #of equations for unknowns.

  Using image coords. with one corner always 0,0 dramatically simplifies things:
  x1,y1 = 0,0; x2,y2 = 0,h; x3,y3 = w,h and x4,y4 = w,0.

  This makes eqs. 1) & 2) boil down to:

  tx = llx & ty = lly

  then, from 3) & 4):

  c = (x2p - llx)/h & d = (y2p - lly)/h

  and from 5) & 6):

  a = (x4p - llx)/w & b = (y4p - lly)/w

  Finally, the PS matrix form is [a b c d tx ty].
  */
  sprintf (matrix, "[% .31f % .31f % .31f % .31f % .31f % .31f]", (lrx - llx)/w, (lry - lly)/w,
    (ulx - llx)/h, (uly - lly)/h, llx, lly);
}
```