Extendable Image Formats for ArcView GIS 3.1 and 3.2

An ESRI White Paper • July 1999

J-8018

Extendable Image Formats for ArcView GIS 3.1 and 3.2

An ESRI White Paper

Contents **Page**

Extendable Image Formats for ArcView GIS 3.1 and 3.2

Introduction

ArcView® GIS 3.1 and 3.2 software provides developers with the ability to easily add new image format display capabilities to ArcView GIS. There are a number of image formats in use throughout the world that are specific to an industry or market that are not included with ArcView GIS. Now developers can write an external DLL that reads an image format and hook that DLL into ArcView GIS for display of the image. The general approach to extend the ArcView GIS image support is to rely on external DLLs or shared libraries to perform image translation or extraction from a nonsupported format into a format that is supported. The extraction is done into a temporary file as part of the ArcView GIS normal display process. From the user's perspective the process of adding a theme does not change since the new image format is integrated into the current Add Theme model.

In order to extend the image formats inside ArcView GIS there are several steps to be accomplished:

- A DLL or shared library must be written that supports a defined set of entry points. The DLL should be C callable. If the DLL is created under C++ then care will have to be taken to ensure C can call the DLL.

- An ArcView GIS extension must be written that sets up the DLL inside ArcView GIS and defines items such as format name, file extension, and the location of the DLL.

The DLL is integrated into the ArcView GIS environment when the image is added or drawn.

- When a new ArcView image theme is created using one of the external image formats, the DLL is used to query basic information about the image—the number of bands, width and height of the image, and the image's spatial extent.

- Each time the image theme is drawn, the DLL is used to extract only the image data visible in the view's current display area. The extracted image is written to a temporary file in a format that ArcView GIS natively supports.

Support for multiple resolutions and pyramid layers can also be provided using the same approach. Enough information is passed to the image extraction routine to allow the external DLL to determine the appropriate resolution or pyramid layer to use. Use of

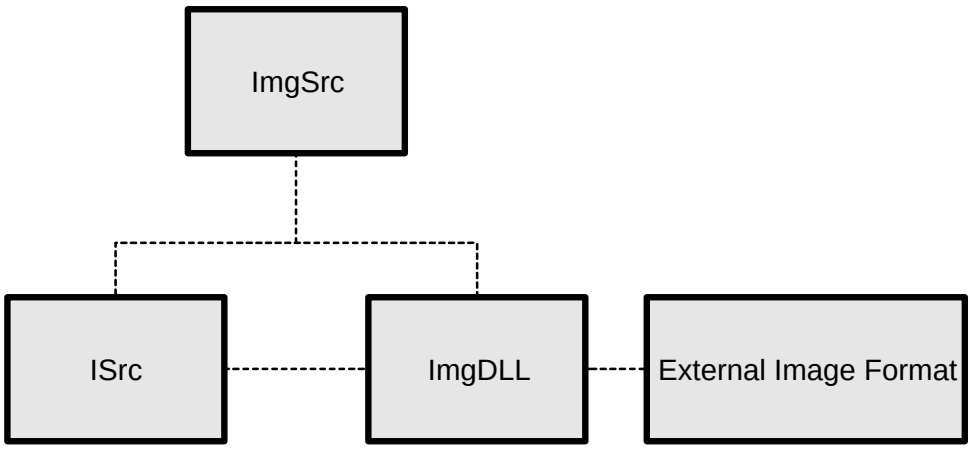
multiple resolutions and pyramid layers should greatly improve image display speed. Additionally, since ArcView GIS only requests the currently visible area of the image, the temporary files should be small in size reducing disk IO activity.

This paper will help educate you to the steps involved in building a DLL to add additional image formats to ArcView GIS. The sections of this paper are

- The Object Model
- Building an ArcView GIS Extension to Load the External Image Format
- Building the DLL-Required Entry Points
- Creating the Temporary BIL/BIP/BSQ Image File
  - Specification of BIL/BIP/BSQ Image Format
  - The BIL/BIP/BSQ Image File
  - The Header File
  - The Colormap File
  - The Statistics File

The Object Model

The object model to support extendable images is quite simple: a new subclass of ImgSrc is defined, ImgDLL, which has an ISrc object as part of its private implementation. The private ISrc object performs the actual image display. Additionally, most of the ImgSrc and ISrc requests provided by ImgDLL are implemented as simple wrappers on top of the ISrc requests. The ImgDLL class also contains the logic to manage the list of external formats and contains the functions for interacting with the external DLLs.



Building an ArcView GIS Extension to Load the External Image Format

Creating an extension to load your external image format is quite easy. First you will want to open a new project in ArcView GIS. Then you need to create four scripts. These four scripts control the creation, loading, and unloading of the extension, and whether the extension can be unloaded. After all four scripts are completed and compiled you need to run the make extension script and your extension will be created. The .avx file created should be placed in the $AVEXT directory or in a directory pointed to by the

$USEREXT environment variable. After this you will be able to load it with the ArcView GIS Extension Manager dialog (from the FILE | Extensions menu choice).

The following are sample scripts that create the "MyImageFormat" Image Support extension. The first is the script that can be used to create the extension file. It is used only to create the extension file and is not included in the extension.

```
' ExternalImage.MakeExt
theExt = Extension.Make("c:\myimg.avx".AsFileName,
                        "My Image Support",
                        NIL,
                        NIL,
                        {})

theExt.SetAbout("Extends ArcView to support My Image images (v3.1).")
theExt.SetExtVersion(3100)
theExt.SetCanUnloadScript(av.FindScript("ExternalImage.CanUnload"))
theExt.SetLoadScript(av.FindScript("ExternalImage.Load"))
theExt.SetUnloadScript(av.FindScript("ExternalImage.Unload"))

theExt.Commit
```

You will need to change the name of the extension file and the name of the extension in the Extension.Make request to reflect your image format. You will also need to change the "About" string given by theExt.SetAbout to reflect information for your image format. You may want to change the version number or leave it to reflect the version number for ArcView GIS.

The next step is to create the load script. It is used to register your external image format DLL with the ImgDLL class when the extension loads so that users have access to your image data sources.

```
' ExternalImage.Load
ImgDLL.RegisterFormat("$AVBIN\myimg.dll".AsFileName,
                      "myi",
                      "bip")
```

You will need to change the path and name of the DLL given by the first argument of the ImgDLL.RegisterFormat to point to your external image format DLL. The second argument should be changed to a three-character extension that will be used to both identify your external image format from others and browse for image data sources in the Add Theme dialog. Your image data source must have a file extension that matches what is given here. The last argument will need to be changed to "bil" or "bsq" if that is the type of temporary image file your external image format DLL creates.

The next script to create is the unload script. It is used to unregister your external image format DLL from the ImgDLL class when the extension is unloaded.

```
' ExternalImage.Unload
ImgDLL.UnregisterFormat("myi")
```

You will need to use the three-character extension of your image file.

The final script to create is the Can Unload script. It is used to check if ArcView GIS currently is using your external image format DLL. If it is, then the Extension Manager dialog will not let the user unload the extension. If it is not in use, then you will be able to unload the extension.

```
' ExternalImage.CanUnload
av.PurgeObjects

if (ImgDLL.GetFormatUseCount("myi") = 0) then
  return(TRUE)
end

return(FALSE)
```

You will need to use the three-character extension of your image file.

To summarize, the three class requests on the new ImgDLL class are

- ImgDLL.RegisterFormat(dllFN, inputExt, outputExt)
- ImgDLL.UnregisterFormat(inputExt)
- ImgDLL.GetFormatUseCount(inputExt)

Building the DLL-Required Entry Points

The external image format DLL is used to query basic information about the input image and to extract temporary BIL/BIP/BSQ images for display. The external image format DLL must contain the following exported entry points:

- IsValidFile
- QueryImageInfo
- GetImage

The external image format DLL can also contain the following optional exported entry point:

- QueryBandStats

Each of these entry points is described below:

```
/* Local definitions. */

/* The ImgDLL external image DLL supports the following functions.  Each
*  of these are defined as typedefs below.
*
+ int IsValidFile(path)—Returns TRUE if the input path name points to a
*  valid image file.  FALSE is returned otherwise.
```

```
*
+ int QueryImageInfo(path, nbands, width, height, mbr)—Queries the input
*  image defined by path for the number of bands, image width in pixels,
*  image height in pixels, and the georeferenced minimum bounding
*  rectangle.  The mbr is expressed as left, top, right, and bottom.
*  TRUE is returned on success, FALSE is returned otherwise.
*
+ int GetImage(path, out_path, clip_mbr, out_width, out_height)—Extracts
* an image from path into the temporary image out_path in the output
* format defined when the format is registered with ImgDLL with the
* RegisterFormat class request.  The output format must be either a BIL,
* BIP, or BSQ image. The clip_mbr is the extent needed from the image in
* map coordinates.

* The out_width and out_height is the needed size for the temporary
* output image in pixels.  The mbr is expressed as left, top, right, and
* bottom.  TRUE is returned on success, FALSE on error.
*
+ int QueryBandStats(path, band, stats)—Queries the input image defined by
*  path for the given band for the band stats.  The stats are returned as
*  min, max, mean, and stdev.  This function is optional.
*/

typedef int (*IsValidFileFuncType)(char *);

typedef int (*QueryImageInfoFuncType)(char *, long *, long *, long *,
double *);
typedef int (*GetImageFuncType)(char *, char *, double *, long, long);
typedef int (*QueryBandStatsFuncType)(char *, long, double *);
```

Creating the Temporary BIL/BIP/BSQ Image File

When ArcView GIS needs to access a piece of an external image format image, the ImgDLL is used to extract only the subset of image data needed. ArcView GIS will pass a map extent, image width, and image height needed from your external image format image. The ImgDLL then needs to take this information and create a temporary BIL/BIP/BSQ image that ArcView GIS then uses. The ImgDLL is responsible for accessing the correct part of the external image format image, resampling it to the resolution ArcView GIS asked for, and properly setting the georeferencing image for the temporary BIL/BIP/BSQ image.

The georeferencing information for the BIL/BIP/BSQ image is stored in the .hdr file with the keywords XDIM, YDIM, ULXMAP, and ULYMAP. These parameters need to be set to the correct coordinates according to your external image format georeferencing and the piece of the image you are extracting. The following is a theoretical example. Say you have an external image format image that has a width of 2,500 pixels and a height of 4,000 pixels. It has a cell size of ten and the coordinate of the center of the upper-left pixel is 330000, 6500000. The upper-left corner of the image will have the coordinate 329995, 6500005 and the lower-right corner will have the coordinate of 354995, 6460005. ArcView GIS asks for the following area where mbr defines an

extent, mbr[0] is the left side of that extent, mbr[1] is the top of that extent, mbr[2] is the right side of that extent, and mbr[3] is the bottom of that extent:

```
mbr[0] = 340000
mbr[1] = 6486666
mbr[2] = 349999
mbr[3] = 6480000
width = 600
height = 400
```

Your external image format DLL should return a BIL/BIP/BSQ .hdr file like

```
BYTEORDER M
LAYOUT BIP
NROWS 400
NCOLS 600
NBANDS 1
NBITS 8
ULXMAP 340000
ULYMAP 6486666
XDIM 16.665
YDIM 16.665
```

You can use the following equations to calculate the extents.

```
XDIM = (mbr[2] - mbr[0]) / width
YDIM = (mbr[1] - mbr[3]) / height
ULXMAP = mbr[0] + (XDIM / 2)
ULYMAP = mbr[1] - (YDIM / 2)
```

XDIM and YDIM should always result in the same number. The aspect ratio between mbr and width/height should always be the same.

The above only works if the mbr falls within the extents of the external image format image. There are two special cases that need to be addressed:

1. The two extents intersect.
2. The mbr is completely outside the extent defined in the external image format image.

In the case that two extents intersect, you would not want to create a temporary BIL/BIP/BSQ image as big as the width and height given to you, but a temporary BIL/BIP/BSQ image that would cover the intersected portion. If ArcView GIS asks for the following area:

```
mbr[0] = 340000
mbr[1] = 6490000
mbr[2] = 360000
mbr[3] = 6476666.666667
width = 600
height = 400
```

Your external image format DLL should return a BIL/BIP/BSQ .hdr file like

```
BYTEORDER M
LAYOUT BIP
NROWS 400
NCOLS 450
NBANDS 1
NBITS 8
ULXMAP 340016.6666667
ULYMAP 6489983.333333
XDIM 33.333333333
YDIM 33.333333333
```

The calculations for XDIM, YDIM, ULXMAP, and ULYMAP do not change in the case above. However, NROWS and NCOLS should be adjusted with the following equations, remembering to always round up to the closest integer. X_min, y_min, x_max, and y_max refer to the full external image format image extent.

```
NCOLS = ((mbr[2] Min x_max) - (mbr[0] Max x_min)) / XDIM
NROWS = ((mbr[1] Min y_max) - (mbr[3] Max y_min)) / YDIM
```

The other case that needs to be addressed is when mbr[0] is less than the minimum x extent or mbr[1] is greater than the maximum y extent of the external image format image. An example of this case is when the mbr falls completely outside the external image format image extent. If ArcView GIS asks for the following area:

```
mbr[0] = 310000
mbr[1] = 6600000
mbr[2] = 550000
mbr[3] = 6440000
width = 600
height = 400
```

Your external image format DLL should return a BIL/BIP/BSQ .hdr file like

```
BYTEORDER M
LAYOUT BIP
NROWS 100
NCOLS 63
NBANDS 1
NBITS 8
ULXMAP 330000
ULYMAP 6500000
XDIM 400
YDIM 400
```

The calculations for XDIM, YDIM, NROWS, and NCOLS do not differ from the above example. However, ULXMAP and ULYMAP are the same as the center of the upper-left

pixel in the image, not from the mbr. If part of the mbr is outside the extent of the external image format image, you want to avoid filling a temporary BIL/BIP/BSQ image with zero values. A zero value will be displayed as a color and possibly mask out other data.

Specification of BIL/BIP/BSQ Image Format

Band interleaved by line (BIL), band interleaved by pixel (BIP), and band sequential (BSQ) are three common methods of organizing image data for multiband images. BIL, BIP, and BSQ are not in themselves image formats, but are schemes for storing the actual pixel values of an image in a file. The pixel data is typically preceded by a file header that contains such ancillary data about the image as the number of rows and columns in the image, a colormap, latitude and longitude, and other information.

The BIL/BIP/BSQ image format consists of four different files. Each file of an image will have the same name but a different file extension. The first is a binary file that actually holds the image data. This file will have a .bil, .bip, or .bsq file extension depending on how the image data is stored. The second file is an ASCII file that holds descriptive information that describes the image data. This file will have a .hdr file extension. The last two files are optional. They are both ASCII files. The colormap file describes the image colormap for single-band pseudocolor images and will have a .clr file extension. The statistics file describes image statistics for each spectral band in a grayscale or multiband image and has a .stx file extension.

The BIL/BIP/BSQ Image File

The binary image file for the BIL/BIP/BSQ image format is merely a bit stream of the image data. How the image data is arranged in that bit stream defines whether it is a BIL, BIP, or BSQ image.

Band interleaved by line data stores pixel information band by band for each line, or row, of the image. For example, given a three-band image, all three bands of data are written for row one, all three bands of data are written for row two, and so on, until the total number of rows in the image is reached.

Band interleaved by pixel data is similar to BIL data, except that the data for each pixel is written band by band. For example, with the same three-band image, the data for bands one, two, and three is written for the first pixel in column one; the data for bands one, two, and three is written for the first pixel in column two; and so on.

Band sequential format stores information for the image one band at a time. In other words, data for all pixels for band one is stored first, then data for all pixels for band two, and so on.

The Header File

The header file describes the image data. You must generate a header file for each image to be displayed. Name the header file after the image, with a .hdr extension; for example,

<image>.hdr

The header file contains a set of entries, each of which describes a particular attribute of the image. For example, an entry can describe the number of rows or columns in the image. The format of each entry is

keyword value

where *keyword* indicates the particular attribute that is being set, and *value* is the value the attribute is being set to. The entries in the header can be in any order, but each must be on a separate line of the file. Any line in the file that does not begin with a keyword is treated as a comment and ignored.

The following list identifies the valid keywords and describes the particular image attribute they define. Several of the keywords have default values. When the default value accurately represents the image data, you can omit the keyword from the header file.

- nrows—The number of rows in the image. Rows are parallel to the x-axis of the map coordinate system. There is no default.

- ncols—The number of columns in the image. Columns are parallel to the y-axis of the map coordinate system. There is no default.

- nbands—The number of spectral bands in the image. The default is 1.

- nbits—The number of bits per pixel per band. Acceptable values are 1, 4, 8, 16, and 32. The default value is eight bits per pixel per band. For a true color image with three bands (R, G, B) stored using eight bits for each pixel in each band, nbits equals eight and nbands equals three, for a total of twenty-four bits per pixel. For an image with nbits equal to one, nbands must also equal one.

- byteorder—The byte order in which image pixel values are stored. The byte order is important for sixteen-bit images, with two bytes per pixel. Acceptable values are

  I—Intel byte order (Silicon Graphics, DEC Alpha, PC)—Also known as little-endian.

  M—Motorola byte order (Sun, HP, etc.)—Also known as big-endian.

  The default byte order is the same as that of the host machine executing the software.

- layout—The organization of the bands in the image file. Acceptable values are

  bil—Band interleaved by line.
  bip—Band interleaved by pixel.
  bsq—Band sequential.

  The default layout is bil.

- skipbytes—The number of bytes of data in the image file to skip in order to reach the start of the image data. This keyword allows you to bypass any existing image header information in the file. The default value is zero bytes.

- ulxmap—The x-axis map coordinate of the center of the upper-left pixel. If you specify this parameter, set ulymap, too, otherwise a default value is used.

- ulymap—The y-axis map coordinate of the center of the upper-left pixel. If this parameter is specified, ulxmap must also be set, otherwise a default value is used.

- xdim—The x-dimension of a pixel in map units. If this parameter is specified, ydim must also be set, otherwise a default value is used.

- ydim—The y-dimension of a pixel in map units. If this parameter is specified, xdim must also be set, otherwise a default value is used.

- bandrowbytes—The number of bytes per band per row. This must be an integer. This keyword is used only with BIL files when there are extra bits at the end of each band within a row that must be skipped.

Bandrowbytes can be thought of as an index to the starting point of the next band of data. Starting at the beginning of any band in a row, moving bandrowbytes along that row leads to the beginning of the next band.

To set bandrowbytes, you must know the layout of the image data, or more specifically, how many bytes are used to store pixel values for each band in a row. If bandrowbytes is not specified, a default value is calculated with the following equation:

bandrowbytes = the smallest integer(ncols * nbits) / 8

The default value handles cases when

- There are no extra trailing bits at the end of each band in a row.

- The number of bytes per band per row is the smallest integer number of bytes that will adequately store the data for the band (e.g., if the data required 2.5 bytes, 3 bytes is the smallest integer number of bytes that could store the data).

In these two cases, bandrowbytes does not need to be set. If, however, the number of bytes per band per row is greater than the default, set bandrowbytes accordingly.

The following two examples show the default behavior of bandrowbytes. The first example describes the cases for which there are no trailing bits at the end of a band in a row and the second describes the cases for which there are.

If there are no extra trailing bits at the end of a band, bandrowbytes equals the number of bytes used to store the image data.

For example, given a six-by-six image with three bands and eight bits (1 byte) per pixel, the image data will require six bytes per band per row.

```
bytes per band per row = ncols * nbits
= 6 * 8
= 48 bits or 6 bytes
```

By default, bandrowbytes is set to six bytes, as shown by the following equation:

```
bandrowbytes = (ncols * nbits) / 8
= (6 * 8) / 8
= 48 / 8
bandrowbytes = 6 bytes
```

Because the number of bytes per band per row equals bandrowbytes, the default value is the appropriate setting. Thus, bandrowbytes does not need to be explicitly specified. The next example illustrates a case where bandrowbytes does not equal the number of bytes of data per band per row.

Suppose you have a three-band image of five rows and five columns with four bits per pixel. By default, bandrowbytes is set to the smallest integer number of bytes that will adequately hold the data. In this case, the defaulted value is three. This is calculated as follows:

```
bandrowbytes = (ncols * nbits) / 8
= (5 * 4) / 8
= 20 / 8
= 2.5
= 3 (when rounded up to the nearest integer)
```

The image data, however, only requires 2.5 bytes, which is calculated by multiplying ncols by nbits. Thus, the number of bytes that will be skipped is .5 bytes (4 bits), or the difference between three bytes (bandrowbytes) and 2.5 bytes (image data bytes).

- totalrowbytes—The total number of bytes of data per row. Use totalrowbytes when there are extra trailing bits at the end of each row.

For a BIL file, the default value for totalrowbytes is calculated with the following equation:

```
totalrowbytes = nbands * bandrowbytes
```

The default value assumes that there are no extra trailing bits at the end of each row. If there are, set totalrowbytes accordingly. For example, given a three-band image with bandrowbytes equal to three, totalrowbytes will equal nine by default. If there is an extra trailing byte of data at the end of the row, set totalrowbytes to ten.

For a BIP file, the default value is calculated with a different equation:

```
totalrowbytes =
the smallest integer ³ (ncols * nbands * nbits) / 8
```

Totalrowbytes is rounded up to the nearest integer number of bytes that can adequately store the pixel data for the row. For example, given a five-by-five BIP image with three bands and four bits per pixel, the default value for totalrowbytes is

```
totalrowbytes = (ncols * nbands * nbits) / 8
= (5 * 3 * 4) / 8
= 60 / 8
= 7.5
= 8 (when rounded up to the nearest integer)
```

If the default value for totalrowbytes does not accurately represent the layout of the data, totalrowbytes must be set to the appropriate number of bytes in each row.

- bandgapbytes—The number of bytes between bands in a BSQ format image. The default is zero.

The following file is a typical header file that might be generated for a band interleaved by line satellite image where the image data is preceded by a 128-byte header.

```
Sample BIL header file
Lines that don't begin with a keyword are treated as comments.
nrows 1024  Comments can be placed here as well
ncols 1024
nbands 3
nbits 8
layout bil
skipbytes 128
```

The following table gives a summary of the keywords that can be used in the .hdr file.

Keyword	Acceptable Values	Default
nrows	any integer > 0	none
ncols	any integer > 0	none
nbands	any integer > 0	1
nbits	1, 4, 8, 16, 32	8
byteorder	I = Intel; M = Motorola	same as host machine
layout	bil, bip, bsq	bil
skipbytes	any integer >= 0	0
ulxmap	any real number	0
ulymap	any real number	nrows - 1
xdim	any real number	1
ydim	any real number	1
bandrowbytes	any integer > 0	smallest integer >= (ncols * nbits) / 8
totalrowbytes	any integer > 0	for bil - nbands * bandrowbytes for bip - smallest integer >= (ncols * nbands * nbits) / 8
bandgapbytes	any integer >= 0	0

The Colormap File

The colormap file is an optional ASCII file that describes the image colormap for single-band pseudocolor images. If this file doesn't exist, the image displays as a grayscale.

The colormap file records the colors to be associated with pixel values in the image. Colors are defined using the RGB color model that describes colors in the amount of red, green, and blue they contain. The file consists of a set of entries, each on a separate line, that describes the color corresponding to a pixel value in the image.

Each entry has the following format:

value red green blue

where *value* is a given pixel value and *red*, *green*, and *blue* are the color components for the pixel. All entries should be in ascending order by pixel value. If the first nonblank character on the line is not a number, the line is considered to be a comment and ignored. Any nonblank characters in a line beyond the fourth parameter (blue) are ignored and may be used as comments as well.

The red, green, and blue components are described using a scale with values ranging from zero to 255. As the color value increases, so does the intensity of the particular color component. The default color for a pixel value with no entry is black. A sample colormap file for a raster soils map with pixel values of 11, 16, 18, 19, 21, 98, and 99 is shown below.

```
Colormap file for Soils map
Entries are sorted in ascending order by pixel value.

11 255 0 0 (red)
16 255 165 0 (orange)
18 255 255 0 (yellow)
19 0 255 0 (green)
21 0 0 255 (blue)
98 0 255 255 (cyan)
99 160 32 240 (purple)
```

Colormap files are only used with single-band images. Any single-band image with a colormap file will be interpreted as a pseudocolor image. Colormap files that accompany multiband images are ignored.

The Statistics File

The statistics file is an optional file that describes image statistics for each spectral band in a grayscale or multiband image. The file consists of a series of entries, one for each band, that records the minimum pixel value, the maximum pixel value, the mean, the standard deviation, and two linear contrast stretch parameters.

Each entry has the format (all values appear on the same line in the file for each band):

band minimum maximum {mean} {std_deviation}{linear_stretch_min} {linear_stretch_max}

where *band* is the band number, *minimum* is the minimum pixel value in the band, *maximum* is the maximum pixel value in the band, *mean* is the mean pixel value, *std_deviation* is the standard deviation, *linear_stretch_min* is the minimum pixel value for a linear contrast stretch, and *linear_stretch_max* is the maximum pixel value for a linear contrast stretch.

The values for each parameter are entered on one line. Any entry where the first nonblank character is not a number is treated as a comment and ignored. The band number and the minimum and maximum pixel values are required parameters; the mean, the standard deviation, linear stretch minimum value, and linear stretch maximum value are optional parameters. Use a "#" to skip the optional parameters.

Band numbers can range from one to nbands. The linear_stretch_min and linear_stretch_max parameters are used to expand the contrast of the displayed image. Pixel values less than linear_stretch_min are displayed with black, and pixel values greater than linear_stretch_max are displayed with white. The pixel values that fall in between the minimum and maximum linear stretch parameters are displayed using shades of gray, with lower pixel values being displayed with darker shades of gray.

The pixel values that fall between the linear stretch parameters are displayed with the maximum number of gray shades available on the display device.

If linear_stretch_min and linear_stretch_max are not specified, they default to the mean minus two standard deviations for linear_stretch_min and the mean plus two standard

deviations for linear_stretch_max. If the standard deviation is not given, the minimum and maximum pixel values are used as the contrast stretching parameters.

For multiband images, each band is stretched before the composite image displays. The presence of a color file (.clr) will override the linear contrast stretching of a single-band grayscale image and, instead, display the image as a pseudocolor image.

The following file is a sample statistics file for a four-band satellite image with eight bits per pixel per band:

```
Image statistics file
1 2 118 67 10
Band 2 has linear contrast stretch parameters:
2 23 251 112 23 80 90
3 68 91 73 4
Band 4 does not contain values for mean and standard deviation:
4 126 198 # # 135 167
```