The Image Foundation DataBlade module allows you to store, manage, and manipulate image data and image metadata using SQL statements in the same system in which you store more traditional data.

With the Image Foundation DataBlade module, you can:

- Store and retrieve images in the database or on remote computers and storage servers
- Store and retrieve image metadata
- Convert data among several image file formats
- Transform images using the industry-standard CVT command set

Formats Supported

With the Image Foundation DataBlade module, users can read, write, get image attributes from, and convert to/from any of the formats below:

- TIFF
- FlashPix
- JPEG
- PNG
- GIF
- Photoshop

Image Transformations

The supported image operations are:

- scaling (zoom)
- cropping
- translation
- rotation
- blur
- sharpen
- contrast modification
- color twist - color hue and saturation can be modified

The CVT command is part of the Internet Imaging Protocol (IIP) standard, which was originally structured to take advantage of the FlashPix image format architecture. This command syntax is used as a uniform method of handling an image stored in any format, in a resolution-independent manner. A

DataBlade function implements the CVT functionality, so in a single function call you can specify rotation, zoom factors, color processing and crop parameters.

Getting Started

Here is a simple statement:

```
SELECT NewImage( CVT( photo, 'wid=300'), 'tree300.jpg')
FROM images WHERE id='tree';
```

'photo' is a column in your table where you store images in their original format (for example TIFF), 'tree300.jpg' is the name of the destination image (the format of a file is determined by the server in this case from the file extension), 'wid=300' determines the width of the result image.
In the example above, CVT() reads the selected image and transforms it according to the second UDR argument. NewImage() takes care of converting the result into any of the supported image file formats.

Let's say that you need a 100 pixel and a 200 pixel wide version of your tree high-resolution image. In preparing your image for the Web, you would manually scale the original image using a tool of your choice and produce 2 images out of it that you would need to reference in your HTML pages:

http://your_server/tree100.jpg
http://your_server/tree200.jpg

With Informix Image Foundation DataBlade module, this task can be easily accomplished in a single statement:

```
SELECT
  NewImage( CVT( photo, 'wid=100'), 'tree100.jpg')
  NewImage( CVT( photo, 'wid=200'), 'tree200.jpg')
FROM images WHERE id='tree';
```

If you use the Informix Web DataBlade, the server can generate these images when an Informix App-Page containing the code below is parsed:

```
<?MISQL SQL=
  "SELECT
    NewImage( CVT( photo, 'wid=100'),'blob:///?fmt=image/jpeg')::blob,
    NewImage( CVT( photo, 'wid=200'),'blob:///?fmt=image/jpeg')::blob
  FROM images WHERE id='tree';">
<IMG SRC="$WEB_HOME?LO=$1">
<IMG SRC="$WEB_HOME?LO=$2">
<?/MISQL>
```

By doing this, you completely remove the manual process of creating special Web image versions, and replace them with the simple URL based on-demand image manipulation of a single image source. If you need an additional size or to change the sizes of images as your HTML pages change, the new image is just a URL away.

JPEG Quality

JPEG quality may be specified with the 'quality' property:

```
SELECT NewImage( CVT( photo, 'wid=250'),
  'bridge.jpg?FMT=image/jpeg,quality=90')
FROM images WHERE id='bridge';
```

The quality is specified as a number between 10 and 100 where 10 represents the lowest level of quality and 100 is the maximum. Low values give poor image quality, but high compression ratios; high numbers give good image quality, but less compression. The default quality setting is 75.

Here are examples of images with qualities 15, 30, 60 and 90:




Using this command, you can serve different qualities of images depending on the bandwidth constraints that you are dealing with. Again, everything is just a query away.

For each image file format, there are several image properties (for example the color space) that you can control using a similar syntax.

Image Size

You can set a size of an image by using the CVT() WID and HEI modifiers. Let's look at our example.

```
SELECT NewImage( CVT( photo, 'wid=300'), 'tree300.jpg')
FROM images WHERE id='tree';
```

This URL delivers an image that is 300 pixels wide. The height of the image is set automatically by the server to preserve the image's aspect ratio.

If you want the server to deliver the image at a bigger size, all you have to do is change the width value:

```
SELECT NewImage( CVT( photo, 'wid=400'), 'tree400.jpg')
FROM images WHERE id='tree';
```

You can do the same by specifying only the height (HEI=200 for example), or specifying both the width and the height. By specifying both the width and the height of your image, however, the aspect ratio will not be maintained. This would result in the image being stretched non-uniformly.

If you specify no width, the server will return the source image at its original resolution.

GIF Properties

The user can force specific colors to appear in a GIF image. GIFs can have at most 256 colors so it is sometimes necessary to reduce the number of colors in an image when displaying or saving it as a GIF. Specifically, this means that most colors in the original image will not appear in the resulting GIF.

When colors are reduced, the server creates a palette for the image by finding clusters of similar colors in the image and replacing them with a single representative color. The reduced color image is created by replacing colors in the original image with the nearest color in the palette. The server tries to chose representative colors to minimize the difference between the original image and the reduced color image. The setcolor property allows specific colors to be inserted into the palette; any remaining colors are determined as before.

The format for the setcolor property is as follows:

```
execute function NewImage(
  CVT('photo.psd', ''),
  'photo.gif?fmt=image/gif,setcolor=0x00RRGGBB'
);
```

where 0x00RRGGBB represents the color hexadecimal value that determines the background color of your image.

The setcolor GIF property specifies the color that is to be preserved as a 6-digit hexadecimal number with two digits for each of the red, green and blue components of the color (in that order).

Transparent GIFs
The following illustrates an example of how the transparency property would be used in creating transparent GIF images:

To create a transparent GIF, an index for the transparent color must be specified. This index is set using the "transparency" option. In order to make a specific color in the original image appear transparent, that color should be inserted into the palette using the setcolor option. Setcolor also ensures that the transparent color will be at a certain index because colors specified using the "setcolor" option are indexed as each "setcolor" option is encountered from left to right. The first setcolor is indexed as 0, the next is indexed as 1 and so on.

The CVT command for creating a transparent GIF would be as follows:

```
execute function NewImage(
  CVT('photo.psd', ''),
  'photo.gif?fmt=image/gif,setcolor=0x00RRGGBB,transparency=0
```

This example assumes that white is the transparent color and inserts it into the palette using the property setcolor=ffffff. The inserted color is placed at the first palette index, which is 0. The option transparency=0 specifies that palette index 0 is to represent the transparent color in the resulting GIF image.

BGColor

The background color of an image with an alpha channel or with a transparent background normally shows through as white. With the implementation of the BGColor modifier, you can set the background to any RGB color value:

```
execute function NewImage(
  CVT('ifmxlogo.psd', 'bgcolor=0x00RRGGBB'),
  'ifmxlogo_y.gif?fmt=image/gif,setcolor=0x00RRGGBB'
);
```

where 0x00RRGGBB represents the color hexadecimal value that determines the background color of your image. The 'setcolor' GIF property forces the usage of the same color in the GIF color pallette, so that the background is properly rendered.

Below you will find examples of the Informix logo, with an alpha channel, served up with various different background colors illustrating the functionality of the BGColor modifier. This is very useful when placing an image on a background other than white; the background of the image being placed on a Web page, for example, can be set to match the background color of the Web page itself.


```
execute function NewImage(CVT('ifmxlogo.psd', 'bgcolor=0x00ffcccc'),
'ifmxlogo_r.gif?fmt=image/gif,setcolor=0x00ffcccc');
```


```
execute function NewImage(CVT('ifmxlogo.psd', 'bgcolor=0x0099cc99'),
'ifmxlogo_g.gif?fmt=image/gif,setcolor=0x0099cc99');
```


```
execute function NewImage(CVT('ifmxlogo.psd', 'bgcolor=0x0099cccc'),
'ifmxlogo_b.gif?fmt=image/gif,setcolor=0x0099cccc');
```

execute function NewImage(CVT('ifmxlogo.psd', 'bgcolor=0x0099cc99'),
'ifmxlogo_y.gif?fmt=image/gif,setcolor=0x0099cc99');

Contrast

The CNT modifier increases or reduces the image contrast:

```
SELECT NewImage( CVT( photo, 'WID=240&CNT=2.0'),
  'shuttle.jpg')
FROM images WHERE id='shuttle';
```

'shuttle' identifies the image below:

The adjustment is specified as a single floating point number: Using the number 1.0 leaves the contrast unchanged; values > 1.0 increase the contrast; values between 0.0 and 1.0 reduce the contrast; 0.0 makes the entire image a mid-tone grey.

See the different contrast levels in the examples below:

CNT=2.0

CNT=0.5

h3>Blur and Sharpen

An image may also be blurred or sharpened using the FTR modifier. The example shown below will apply a blur filter (with a value of -17) to the image.

```
SELECT NewImage( CVT( photo, 'WID=250&FTR=-17'),
  'ctryroad.jpg')
FROM images WHERE id='country road';
```

'country road' identifies the image below:

The amount of filtering is specified by a number between -20 and 20. Negative numbers blur the image and positive numbers sharpen it.

See the filter examples below:

FTR=-17.0

FTR=+17.0

Color Correction

Image colors may be modified by specifying a color correction matrix, (4 rows by 4 columns), using the CTW modifier. This command allows you to apply complex color manipulations to your image.

The values of the CTW modifiers are determined through mathematical matrix multiplications. To combine different color 'twists' and brightness levels as described above, you can simply multiply the appropriate matrix values until you achieve the color correction you want to apply to your image.

We will show you a few examples beginning with an image in its original format, then different color 'twists' applied to the image using the CTW modifier will demonstrate the various color effects you can achieve.

Cropping

By using the RGN modifier, you may request that a specified region of the image be delivered by the server.

The source region rectangle (your original image, prior to cropping) is specified in a resolution independent coordinate system where (0,0) are the coordinates of the upper left corner of the image, and (R,1) are the coordinates of the lower right corner.

Note: "R" represents the aspect ratio (width/height) of the image.

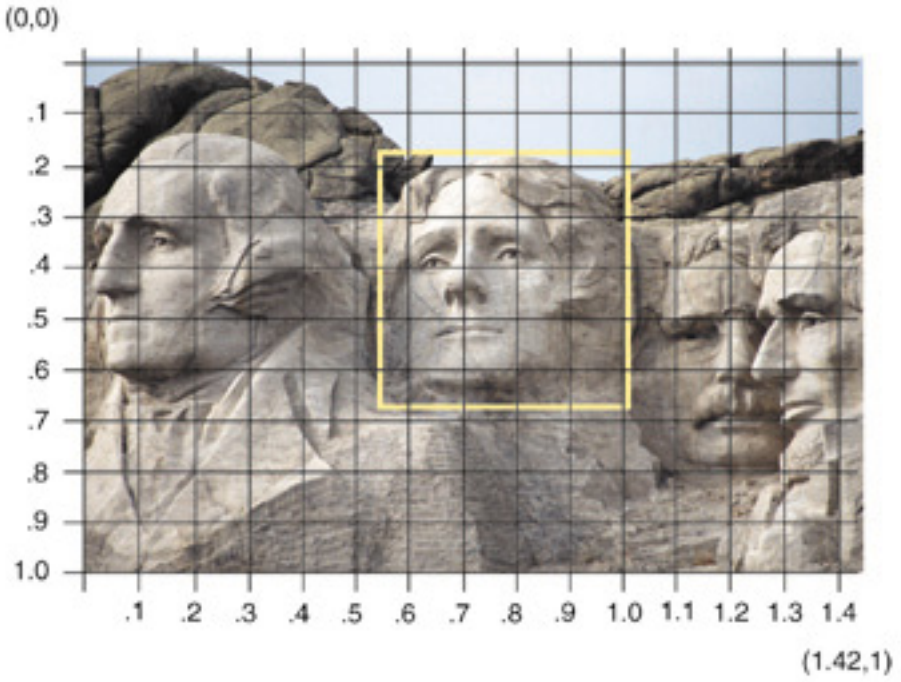
```
SELECT NewImage( CVT( photo, 'RGN=0,0,0.75,0.5&WID=250'),
  'mtrushmore.jpg')
FROM images WHERE id='mount rushmore';
```

The values of the cropped region you wish to call up are represented as:

RGN=x,y,width,height

where (x,y) are the coordinates of the upper left corner of the cropped region. In our example, the default height is calculated from the aspect ratio and therefore does not need to be specified.

Note: In order to determine the (x,y) coordinates of the cropped region rectangle, you may use a 'grid system'. If you draw grid lines (horizontally and vertically) on your original image, in 0.1 increments, for example, you can visually approximate the (x,y) coordinates for your cropped region (indicated in yellow). Use these approximate values for your RGN modifier. You will have to adjust the (x,y) values within the RGN modifier until you achieve the desired end result.



The RGN modifier is a convenient way of specifying a requested crop region of an image, qualifying it with an upright rectangle.

Below is a more complex example of scaling and cropping.

RGN=0,0,0.75,0.5

RGN=0.75,0,0.75,0.5

RGN=0,0.5,0.75,0.5

RGN=0.75,0.5,0.75,0.5

The cropped regions that are specified with the RGN modifier, can also be specified using a combination of RAR and AFN modifiers. In fact, the server internally implements the RGN modifier in terms of the RAR and AFN modifiers. As a result, the RGN modifier overrides the RAR modifier.

The Spatial Transforms page provides more details about the result aspect ratio (RAR) and the affine transformation (AFN) modifiers and how they work.

Flips and Rotations

Rotations of 90, 180, and 270 degrees can be specified using the RFM command:

RFM=rot

where **rot** is one of the above values. Rotation is counterclockwise. Flips may be specified by adding another parameter:

RFM=rot, flip

where **flip** is 0 for a vertical flip (mirrored in the horizontal axis) and 90 for a horizontal flip (mirrored in the vertical axis); **rot** may also be 0 in this case. Here is an example of an image transformed using RFM:

```
SELECT NewImage( CVT( photo, 'RFM=0,90&WID=250'),
  'surfer.jpg')
FROM images WHERE id='surfer';
```

Original

RFM=0,90

Any transformation specified with the RFM modifier could also be specified with a combination of result aspect ratio and affine modifiers.

Spatial Transformations

In order to better explain the affine transformation modifier (AFN), the rectangle of interest (ROI), and the result aspect ratio (RAR), we include some background information of the Flashpix format coordinate spaces. There are two resolution independent coordinate spaces used: the source space, and the result space. Both of these coordinate spaces have an **x** coordinate which increases to the right and a **y** coordinate which increases downwards.

In the source space the source image data occupies a rectangle whose upper left corner is at (0,0) and the lower right corner at (R_s,1), where R_s is the aspect ratio of the highest resolution image in the source image data. The result image occupies a rectangle with the upper left corner at (0,0), and the lower right corner at (R_r,1), where R_r may be specified using one of the CVT modifiers described below.

A source space rectangle of interest defines the bounds of the "valid" source image data. This rectangle of interest may include the whole source image rectangle, or it may crop only part of it. The image is considered to be transparent (or white) outside of this rectangle of interest. This region of interest may be specified by the ROI modifier; the default rectangle of interest is the entire source image. If parts of the image that are outside the rectangle of interest are exposed, they are rendered as black by the CVT command.

The result and source image coordinate spaces are related by an affine transformation which maps the result space into the source space. An affine transformation can express rotations, translations, scales, and shears. The matrix representing this affine transformation may be specified by the AFN modifier.

The result image is always rendered as an upright rectangle. The color of a result image point is determined by the color of the source image at the transformed location of that point. If this location falls outside of the source rectangle of interest it is rendered as black.

Result Aspect Ratio

The result aspect ratio is given as a single floating point number using the RAR modifier. It defines the result image rectangle as described above. For example the modifier

RAR=0.5

defines a result image that is twice as high as it is wide. The RAR modifier is primarily used in conjunction with the affine transformation to perform cropping.

```
SELECT NewImage( CVT( photo, 'RAR=0.5&HEI=300'),
  'tree.jpg')
FROM images WHERE id='tree';
```

Affine Matrix

The affine matrix (4x4) is specified as a sequence of 16 coefficients using the AFN modifier.

As a simple example, the following CVT command returns the specified image flipped vertically:

```
SELECT NewImage(
  CVT( photo, 'AFN=1,0,0,0,0,-1,0,1,0,0,1,0,0,0,0,1&WID=300'),
  'ctryroad.jpg')
FROM images WHERE id='country road';
```

The following sequence of examples illustrates how the AFN modifier can be used to rotate a 'result' image, or in this case, a 'cropped' image. The examples will also show the result image scaled, translated from its 'source state' position, and finally how to change the cropped region shape. These examples are incremental in nature illustrating how one operation is performed to an image and then the result image of this initial operation becomes the source image for the subsequent example.

These examples also use the CVT modifier BGCOLOR=0x00808080 so that the transparent parts of the image will appear as grey.

Region of Interest

The region of interest is specified as a rectangle in source image coordinates using the ROI command. It has the same format as the RGN modifier, but the meaning is different. The parts of the source image that fall outside of this rectangle are still present but deemed to be transparent; they are not "clipped" away.
The transparent area can be "colorized" using the BGCOLOR modifier (see the example below):

```
SELECT NewImage(
  CVT( photo, 'ROI=0.2,0.2,1.1,0.6&WID=300&BGCOLOR=0x0099CCCC'),
  'tree-roi.jpg')
FROM images WHERE id='tree';
```