HOWTO: Scan Unsupported Image Formats

DotTwain supports scanning into a .NET Bitmap for common formats such as 1-bit, 4-bit, 8-bit (color and grayscale), 16-bit color (RGB 5-6-5), 24-bit and 32-bit. DotTwain also supports JPEG compressed memory transfers. However, there may be times when you need to get a format not supported, or you may want to use Group4 (or some other) compressed memory transfer. This article will go over the process of getting this image data and provides a simple demo as a starting point.

Getting The Data

To gain access to the data you must set the Device.**TransferMethod** to **TWSX_MEMORY** and handle the **MemoryDataTransfer** and **TwainDataTransfer** events. These provide information during and after each page is scanned, including a pointer to the raw image data.

The **MemoryDataTransfer** event is raised multiple times as the data is transferred from the scanner into DotTwain in small chunks. Our demo will use this event to get the width, height and bytes per row of the raw image data. While this event does provide a pointer to the data, it would require combining all of the data each time this event is raised to get the entire image, so we will use the pointer in the **TwainDataTransfer** event instead.

The **TwainDataTransfer** event is raised when all data has been received for the image. It provides a pointer to all of the image data, making it easy to work with. Here are the steps we take to convert the raw data into an **AtalaImage**:

1. Because we will be hanlding the image data, the first thing we must do is set the e.**DataHandled** property to true so DotTwain will not try to process the image.
2. The data comes in as raw packed data, so we must unpack the data to make it DWORD aligned in order to create an AtalaImage out of it.
3. We then verify the pixel format and create a palette if needed.
4. A **GlobalAllocPixelMemory** class is constructed to hold the image data and an **AtalaImage** is created from the pixel memory.
5. We set the palette and resolution of the AtalaImage.
6. The Device.**PixelFlavor** is looked at to determine if we need to invert the image or swap the Red and Blue channels.

And we're done with processing the raw data into an AtalaImage, which can then be saved or processed with other **ImageCommand** classes.

HOWTO: Scan Unsupported Image Formats

The attached zip file contains source code for a working example of the steps listed above.

Original Article:

Q10241 - HOWTO: Scan Unsupported Image Formats

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50225/HOWTO-Scan-Unsupported-Image-Formats