



[image on]

Reliable Document Image Viewing: Through Thick and Thin

Web-centric Viewing

It's been a wild ride from DARPA's concept of the World Wide Web to where we are today, and there is little doubt that Web pages are now the dominant vehicle for distribution of information. An insurance company, engineering firm, or government agency can simply post their documents on a Web site, and easily grant extensive viewing capabilities to whomever they choose, worldwide. Or can they?

The fact is, unless your documents are stored as text, the only way you can actually put them **directly** on the Web is as JPEG, GIF, or PNG images. These are the only image formats that Web browsers support. And, as we will discuss, all three of them are generally impractical formats for documents. Any other file format requires third-party software for interpretation and decompression of the images.

Birth of a Document

So, where do documents come from? If they are generated by a computer and intended from the start for distribution, they can easily be formatted as Web pages for others to view. But there are billions of documents that predate the computer era, and billions more created every year that have arrived by postal or FAX delivery, or that were created on incompatible systems. Many of these documents contain information that is vital to business processes, such as invoices, forms, and medical records, to mention just a few.

The most common way to capture paper documents for archive or distribution purposes is via scanning operations, including flatbed and sheet-fed scanners and FAX machines. As images of these documents are digitized, they are compressed using specialized algorithms designed to save storage space and transmission time.

The sheer volume of pages mandates highly-efficient storage methods, even in the face of continuously decreasing media costs. The most efficient, widely-used format that preserves the fidelity of black and white text is provided by the TIFF (Tagged Image File Format) standard using Group 4 (known as G4) compression. The format was originally created for use in FAX transmission.

Why TIFF? Let's say you scan a sample text page in black-and-white, saving every resulting pixel in a bitmap file. This 8.5 x 11 inch page was scanned at 300 dpi, and the scanner created about 8 million pixels (2551 wide by 3299 high). These were packed 8 pixels per byte into about 1.06 million bytes and saved as a BMP file. In preparation to view that page image on a Web site, I could choose to save that file in one of the following formats:

Using Barcodes in Documents – Best Practices

Barcode Basics

A basic understanding of barcode image algorithms can help you make implementation choices that optimize your success. Since 1-dimensional (1D) barcodes, such as Code 128, Code 39, and other codes that consist of bars of varying widths, are detected using subtle different methods than the more sophisticated 2-dimensional (2D) barcodes such as PDF417, DataMatrix, and QR. There are two basic ways for either. Finding the barcode's bar edge, or bar end, and determining what information the barcode contains, or decoding.

To locate 1D codes on a page, the software essentially scans the entire width or height of the document image, searching for vertical patterns of pixels that are typical of barcodes. The more compressed spaces that fall out with a similar pattern, the more likely you are to have located what you are looking for.

The 2D decoding process consists of counting and comparing groups of image pixels in each region of interest to determine the relative width and location of the bars and the white space between them. Modified patterns of the first and last of each barcode generally identify its type. Proper barcode boxes can identify about 50 different 2D barcode types. The pattern between the start and end characters on these barcodes is used to determine the data. The data is then decoded into its original form. The barcode's data is then decoded into its original form. The barcode's data is then decoded into its original form.

Loading and decoding 2D barcodes is significantly more complex than the linear process of finding "barcodes" that simply identify the barcode type. A 2D barcode includes several vertical bars that support multiple lines of a 2D barcode. DataMatrix codes can support three distinct patterns of vertical lines on left and bottom edges, with identifying files and empty lines. DataMatrix codes are the top and right. These identifying features are shown in red in the illustration to the right. The rest of the image barcodes also consists of black or white modules. A QR code includes distinct square "modules" or blocks of four squares. The "module" block length measures the numerical of the QR code. These barcodes utilize 2D barcodes also outline the complete region of interest that surrounds the encoded content.

Once the identifying features of a 2D barcode are found, the presence of black or white pixels in each module can be used to derive the numerical of the barcode according to specific flags. The 2D barcodes can be decoded with varying amounts of resolution, allowing the user to view the accuracy decoded even when up to 30% of the modules are damaged or missing. The higher the




File Format	Size (bytes)
JPEG (Max quality)	761,121
JPEG (Low quality)	255,841
GIF	164,718
PNG	98,936

The JPEG format was created to efficiently compress color photographs. It was designed to maintain smooth transitions between many colors. In fact, black-and-white text represents a worst-case scenario for JPEG, as it creates very sudden color changes in color in a format that uses 24-bits (a byte of red, blue, and green) for each pixel.

Even with high compression, the size is still over 250 kb. Because the JPEG format is optimized for photographs, compressed black and white images introduce visible noise, as shown in this enlarged section from the saved image:



JPG - Highly compressed - 256 kb

Clearly, JPEG is not a very good choice for text documents. The rippling effects, known as **artifacts** can not only make the image appear sloppy, they also create serious problems for optical character recognition (OCR) applications.

The GIF and PNG formats can each create a **lossless** representation of the document image while offering respectable compression ratios of about 6:1 and 11:1 respectively. But the same information stored in TIFF format with G4 compression

occupies only 57,684 bytes, a 42% reduction in the space required for the PNG file,

Xpress

TIF - G4 Compressed - 58 kb

the best “native” option. This is a fairly typical size TIFF for a document scanned at 300 dpi. When many millions of documents are to be stored, these space savings represent very significant cost savings.

So, Why no TIFF Viewing?

There are various reasons that browsers may not directly support viewing of TIFF files, related to their support for multi-page images, and historical developments. But, for today, they do not. Any method of distributing this immense collection of valuable stored information via the Web relies on third-party software located **somewhere** to perform decompression of the stored image files.

Depending on where the decompression is to be performed, that software could be: a) located on an image conversion server, b) downloaded from the server and automatically installed for first use by the browser, or c) pre-installed on the client PC as an add-on component to the browser. These might be referred to **no-client** or **zero footprint**, **thin** client and **thick** client viewing solutions. The term “thin client” was historically used to describe a hardware solution using a relatively inexpensive display device that was typically diskless and had limited computing power, but is used today for software solutions that require little or no installed software beyond



the basic browser. Accusoft Pegasus provides solutions for all three of these approaches, and each of them has advantages and disadvantages, as discussed below.

ASP.NET and AJAX – Or, Getting the Server to do my Homework

The so-called no-client solution is typically implemented using AJAX technology. It may be referred to as no-client because no software is permanently installed on the PC. The original document TIFF images are all stored on Web servers, often along with thumbnails of the images that can be used to help navigate to the desired one. The selected TIFF is decompressed **on the server**, and either the full image, or a selected portion of it, is then transmitted down to the browser in one of the formats that it already knows how to interpret (GIF, PNG, or JPG). Any software that is needed to select or manipulate the image is usually also downloaded from the server as **JavaScript** that can be executed by the browser when the page is displayed. The JavaScript code is the only software that is downloaded to the client. That's not exactly no-client or zero footprint, but it's not actually installed, and it is usually not enough code to cause a significant delay.

Where delay can become problematic is on the server. It must respond to every request from every user for a new image, or even any portion of an image that has not been previously viewed, and is therefore not in the image cache. For large images that are being constantly panned, or when paging through a large, multi-page document, there can be many requests for sections of the image (or file) to be decompressed and transmitted, while the user waits. If the Web

site is intended for very wide distribution, it may be very difficult to predict how many servers are needed to support everyone who might be viewing images. This could result in highly variable service quality for end users.

Accusoft Pegasus offers an ASP.NET solution that implements many advanced viewing functions in JavaScript, where the full power of the client CPU is available. This yields a much faster viewing experience than competing ASP.NET solutions. The client communicates with server-based code, where images are converted for delivery to the client as needed.

A basic ASP.NET demonstration, using [ImageGear®](#) for image processing on the server, can be found at www.accusoft.com. It is straightforward to add support for any of the hundreds of other capabilities that ImageGear provides, including viewing of PDF, JPEG-XR, and many other file formats, image cleanup functions such as despeckle and deskew, as well as the basic image manipulation functions like rotate and zoom.

Thin is In

A “*thin*” client is the second way that documents, including TIFF files, can be viewed via Web browsers. In this implementation, the code that performs the TIFF decompression is actually running on the local PC. That code, often written in Java, can be downloaded and installed with relatively little intervention on the part of the end user. In this case, panning to a new area of an image, or modifying it by rotating, brightening, or another operation, does not require any server interaction. This improves the responsiveness of the user



interface, and allows each server to handle more simultaneous users. Only true native code, as in the thick client implementation described below, can run faster.

The thinness of the client is a bit of an illusion. The Java download itself can be quite large. Some products will also download **more** code every time you choose to view a new file type, while you wait. And Java code, unlike JavaScript which is supported natively in the browsers, cannot run without a Java Virtual Machine (JVM). The JVM includes software libraries that contain many of the functions that the downloaded Java program uses to do its work. This, for example, is where the TIFF decompress code you're running usually resides.

The universality of Java is a big advantage, since it can run on Windows, Macintosh, and Unix clients. This allows the same viewer to be used on the widest available array of clients. Also, because the installation is almost fully automatic, it's very easy to push updates out to all users.

[NetVue®](#) is the thin client solution from Accusoft Pegasus. NetVue uses a client-server approach that allows Microsoft Office documents, for example, to be rendered on the server using a print driver, with the images sent for viewing by the NetVue client. The client can be either an ActiveX plug-in for Internet Explorer, or a Java client on almost any browser.

The Thick of the Matter

The third solution for TIFF viewing is the thick client. Like the thin client, image decompression is performed on the client PC. But, unlike the thin client, all of the code can now be better supported by the viewer

vendor. Since it doesn't have to run on a wide variety of platforms, which is a requirement for Java, all aspects of the application can be optimized for the target client hardware and operating system. Thus, thick clients offer the fastest image viewing performance of the three options.

The Accusoft Pegasus [Prizm® Viewer](#) is the leading commercial TIFF viewer available today. Already installed on over a million PCs, Prizm Viewer has proven that it reliably supports the needs of mission-critical businesses in the defense, banking, insurance, legal, manufacturing, and many other industries.

Installation of the client is actually no more difficult than installing the JVM alone, and no additional viewer software needs to be downloaded before displaying your first TIFF. The complete installation can also be pushed from a server using a single MSI file.

Over its long history, Prizm Viewer has evolved many features supporting business usage, such as **list files** that allow grouping of many disparate file types into a single virtual multi-page document. It offers full-featured annotation with local or server storage, extensive customization including controlled preference settings per user or per Web page, advanced image enhancement, viewing of very large images, support for a wide variety of image types that browsers cannot display, and many other features. For more information, visit www.accusoft.com, where you can also download a fully-functional trial version. Prizm Viewer is fully supported on Internet Explorer and popular Mozilla-based browsers, such as Firefox.



Is that Silverlight at the End of the Tunnel?

We've discussed some of the features of no-client, thin-client, and thick-client implementations. Microsoft has recently added another technology that enables an excellent viewer solution. A new edition of ImageGear designed specifically for Silverlight delivers imaging tools you can use to quickly build your own web-based or desktop viewer. The Silverlight solution offers client-based image decompression and manipulation, providing fast response times, and it also supports annotations that can be used to construct document workflows that enable collaboration between team members.

The ImageGear for .NET (with ASP.NET support) and ImageGear for Silverlight SDKs, and the Prizm Viewer and NetVue applications can all be downloaded for trial use at www.accusoft.com.

You can find Accusoft Pegasus product downloads and features at www.accusoft.com. Please contact us at sales@accusoft.com or support@accusoft.com for more information.

About The Author

Paul B. Firth, Product Manager, Accusoft Pegasus

Paul has been helping to guide the overall product strategy for Accusoft Pegasus since 2005. In this role he works closely with the Sales and R&D teams to identify and satisfy product needs in the highly-dynamic software tools market. He previously led high-tech marketing groups, and has managed teams of both software and hardware developers. Paul holds an M. B. A. in Marketing and Strategic Management from New York University, and a B. S. in Electrical Engineering from the University of Rochester.

About Accusoft Pegasus

Founded in 1991 under the corporate name Pegasus Imaging, and headquartered in Tampa, Florida, Accusoft Pegasus is the largest source for imaging software development kits (SDKs) and image viewers. Imaging technology solutions include barcode, compression, DICOM, editing, forms processing, OCR, PDF, scanning, video, and viewing. Technology is delivered for Microsoft .NET, ActiveX, Silverlight, AJAX, ASP.NET, Windows Workflow, and Java environments. Multiple 32-bit and 64-bit platforms are supported, including Windows, Windows Mobile, Linux, Solaris x86, Solaris SPARC, Mac OS X, and IBM AIX. Visit www.accusoft.com for more information.