

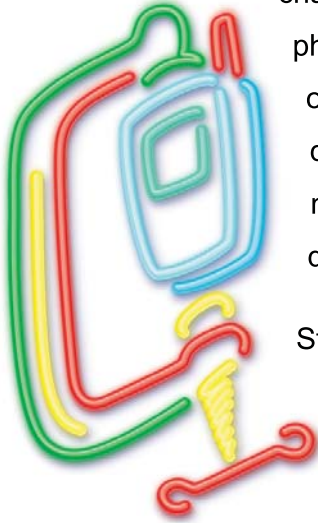
StuffIt® Wireless White Paper

Date: 6/13/2006 Version 3.0

Introduction	2
JPEG Compression	2
<i>Working with Existing JPEG Implementations</i>	5
<i>Replacing Existing JPEG Implementations</i>	6
<i>JPEG Image Compression Results</i>	8
<i>Compression Ratio Variation by JPEG Level</i>	11
StuffIt Lossless Image Format	13
<i>StuffIt Lossless Compression Performance Results</i>	14
<i>StuffIt Lossless Compression Performance: Variation by File Size</i>	17
<i>StuffIt Lossless Compared to JPEG</i>	18
The StuffIt Image Format	20
Compatibility and Interoperability	20
Summary	22
Contact Information	22

Introduction

Stuffit Wireless is a complete multimedia compression solution designed to meet the challenges of today's wireless marketplace. As the wireless phone continues to evolve into a rich media platform, a number of challenges arise. Wireless carriers need more spectrum to dedicate to multimedia transfers, and handset manufacturers must contend with how to best store this content on the device.



Stuffit Wireless combines a patent-pending JPEG compression technology with a range of other algorithms to compress other types of multimedia (music, movies, applications, etc.). This allows Stuffit Wireless to both reduce the bandwidth required to transfer rich media content over the air, and to provide a way to store this content on the phone more efficiently. This, in turn, translates into an improved end user experience.

In addition to its JPEG compression method, Stuffit Wireless technology provides a range of solutions that will allow device manufactures to compress the resource files that make up the handset's user interface and operating system. Storing the various components of the phone's operating system in a more efficient format allows device manufactures to allocate more of the handset's memory to the user's own files.

JPEG Compression

The image compression technology provided by Stuffit builds on nearly 20 years of experience in the field of data compression. Stuffit's legacy is one of continuous innovation; first released in 1986, Stuffit has continued to evolve to provide solutions to some of the most common problems faced by those in the data compression industry. The ability to further compress previously "compression-resistant" JPEG files is the latest such development.

On average, Stuffit's patent-pending JPEG compression technology is able to further reduce the size of JPEG files by an average of between 20 and 30 percent. Stuffit Wireless provides true lossless compression: a binary comparison will reveal that a JPEG file compressed and expanded with Stuffit is an identical replica of the original file.

Traditionally, JPEG images have been classified as "uncompressible" because the JPEG file format already contains a lossless compression component. The output of this compression component appears to other compression programs as a random, uncompressible string of bits.

Creating a JPEG

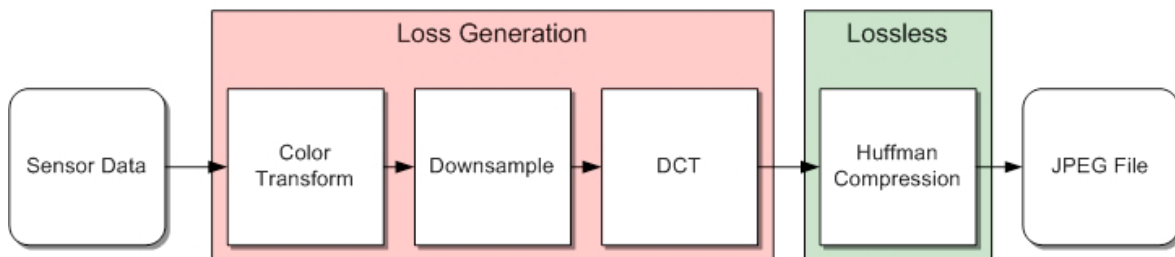


Figure 1: Creating JPEG

As illustrated by Figure 1, the creation of a JPEG image can be split into two sets of functions. In the first set, the size of the image is reduced through a reduction in image quality. The second set applies a lossless compression algorithm (Huffman) to further reduce the file size. This second function is key to understanding the nature of Stuffit's compression breakthrough. Once the Huffman encoding layer has been applied, other lossless compression methods are typically unable to further reduce the size of the file. When someone attempts to further compress a JPEG using a standard Windows compression method--such as zip--there is typically little or no reduction in file size. This is because the near-random nature of the Huffman-compressed data provides little or no structure for typical compression methods to analyze.

Stuffit's JPEG compressor does not attempt to simply compress the near-random data output of a JPEG file, instead it seeks to first convert this data to a state more conducive to compression, then compress that resulting data.

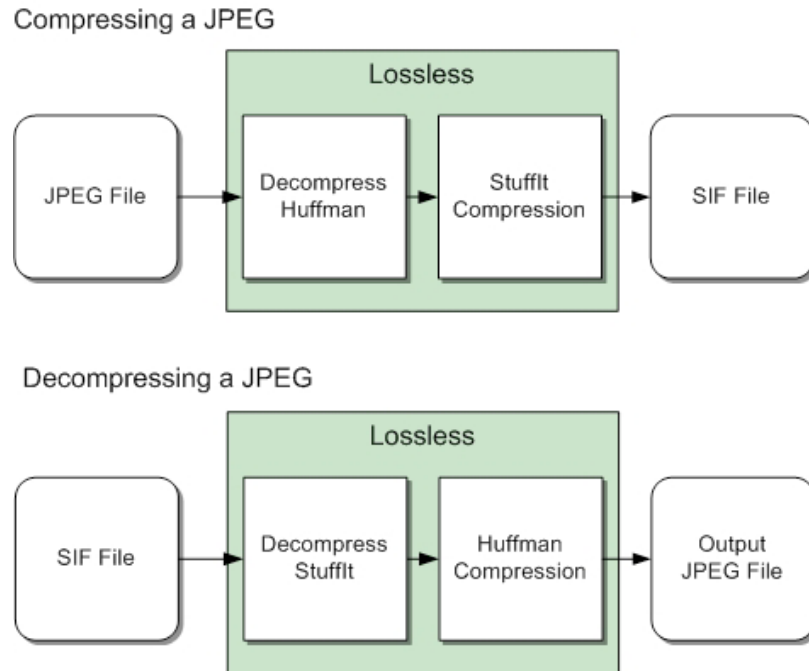


Figure 2: Stuffit JPEG Compression and Decompression

As illustrated by Figure 2, this conversion involves decompressing the Huffman compressed image blocks and then re-compressing them with a new Stuffit algorithm. Any header data (such as EXIF metadata) is separately compressed with a generic compressor and stored within the file. When a Stuffit compressed JPEG is decompressed, the process is reversed; the Stuffit compression method is removed, and the original Huffman encoding is re-applied to the file. In summary, Stuffit achieves its 20 to 30 percent compression gain by replacing an outdated compression scheme with one custom-designed for the task at hand.

The lossless nature of Stuffit's compression method provides the key differentiator between Stuffit's image compression technology and other image compression methods

such as JPEG 2000. While the process of simply opening and re-saving a JPEG image results in image information being lost, images can be converted back and forth between Stuffit and JPEG format without this loss occurring.

Working with Existing JPEG Implementations

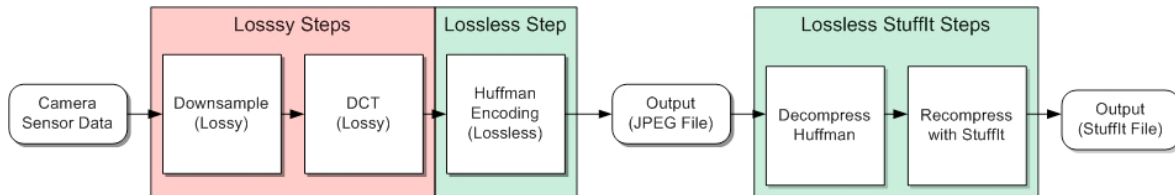


Figure 3: Stuffit JPEG compression works alongside existing JPEG compression methods

As illustrated by figure 3, one of benefits of Stuffit Wireless is that it can exist alongside an existing JPEG implementation. On a device such as a camera that saves output as a JPEG file, Stuffit Wireless can be inserted as an additional process that uses the JPEG file created by the camera. The additional Stuffit processing can be performed automatically whenever a new JPEG is created by the device (camera), or Stuffit can be applied contextually on-demand.

The market for wireless camera-phones illustrates a prime case scenario. Most of the data traffic carried by wireless networks is comprised of MMS messages, consisting of a few lines of text and a JPEG image. The MMS client, in this case, can be configured to call a Stuffit compression API when the user has selected an image as an attachment to a message.

Replacing Existing JPEG Implementations

Although Stuffit Wireless can compress existing JPEG files, some OEM's may prefer to bypass the JPEG format altogether, generating images directly in Stuffit format. For this implementation, Stuffit takes sensor data from the camera as its input rather than an existing JPEG file. This alternative implementation can best be described as a sensor-level interface.

Stuffit Sensor Interface

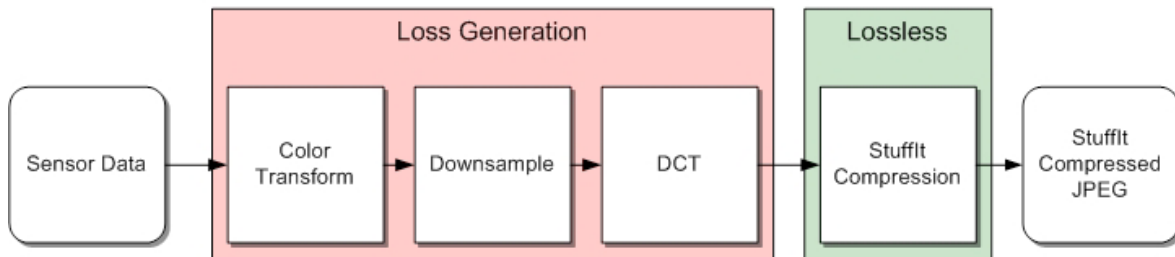


Figure 4: JPEG Compression direct from the Image Sensor

When the sensor interface technique is used (Figure 4), the captured image is never written out in JPEG format. The loss-generating portion of the JPEG creation process can still be employed if needed, but this step is handled by Stuffit, rather than a legacy JPEG implementation. After the loss-generating processes have been performed, Stuffit's lossless compression method is applied directly.

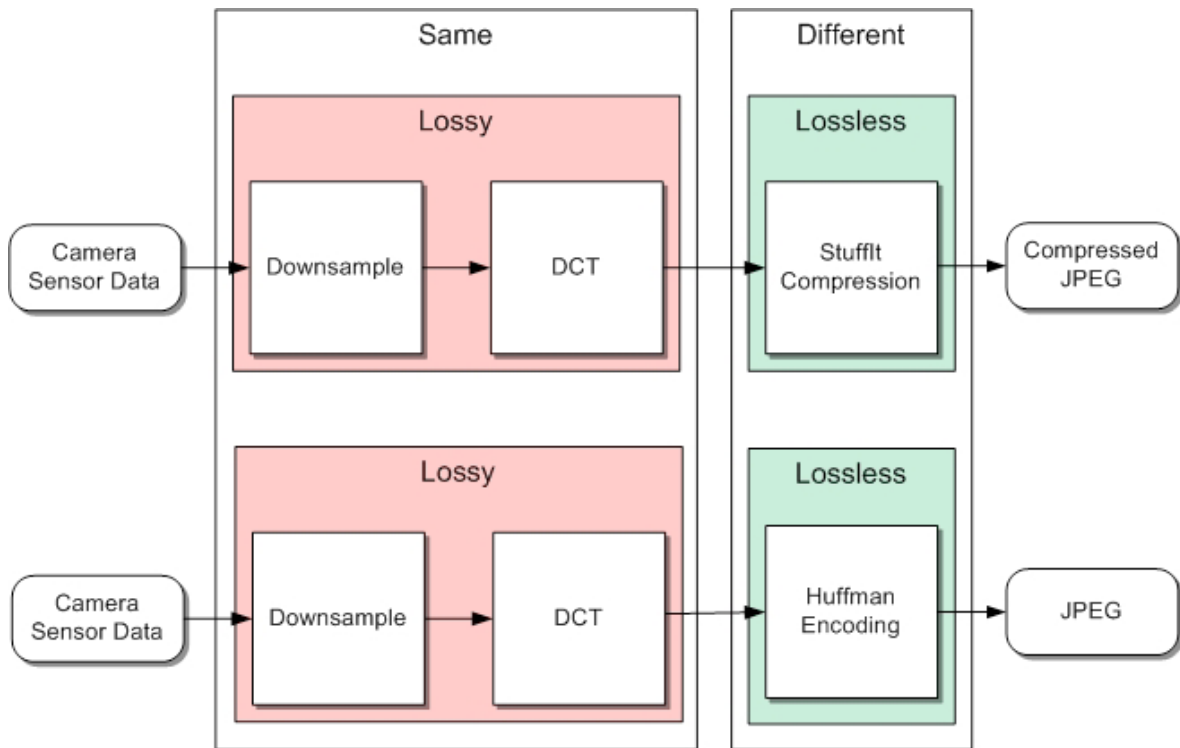


Figure 5: StuffIt's Sensor interface compared to standard JPEG creation

Figure 5 compares the creation of a regular JPEG image with the process described in figure 4. In creating a regular JPEG, Huffman encoding is applied after the loss-generating processes have been performed. In the second scenario, a direct sensor level interface allows StuffIt compression to be applied much more directly. One clear benefit is that Huffman decompression can be skipped altogether, allowing StuffIt-compressed images to be created more quickly.

It is important to note that images saved directly into StuffIt format can be converted back to JPEG format without any additional loss in image quality. This can be achieved by removing the StuffIt lossless compression layer and replacing it with Huffman.

JPEG Image Compression Results

The compression tests detailed in this section were performed against a set of images taken with a current generation wireless handset. The camera used was selected for its high resolution (2 megapixel) image sensor. All the images used in this test are 1600 x 1200 in size and were taken at the maximum supported image quality setting.

Timing tests were performed on two different platforms. The first platform was the handset originally used to capture the images. A BREW application was written to first compress each JPEG into Stuffit Image Format (SIF), and then to decompress the SIF file back to the original JPEG. The second test platform was a Windows Mobile PDA with an ARM 9 process at 266mhz.



Figure 6: Sample Images taken with a 2 MegaPixel camera phone

In order to generate the numbers presented in Table 2 (below), the files described in Figure 6 were compressed using a BREW version of Stuffit Wireless running on the previously described phone handset. "Compression Time" describes the time required

to take a JPEG file and compress it into Stuffit format. “Decompression Time” refers to the time required to take a Stuffit-compressed version of that file and convert it back to JPEG.

File Name	Original Size	Compressed Size	Percent Saved	Time to Compress (in Seconds)	Time to De-compress (in seconds)
121805_1223a.jpg	245092	199890	18.44	13.9	20
121905_0929a.jpg	234494	166716	28.90	12.9	17
121905_0953a.jpg	335337	240565	28.26	17.2	23.3
121905_0954a.jpg	304123	217245	28.57	16.3	21.4
121905_0954b.jpg	306782	245416	20.00	18.9	24.6
121905_1031a.jpg	380577	270469	28.93	19.2	25.4
121905_1031b.jpg	235791	191864	18.63	13.4	17.4
121905_1032a.jpg	267663	221691	17.18	14.1	19.5
121905_1032b.jpg	301310	239286	20.58	18.2	24.2
121905_1033a.jpg	235308	174767	25.73	16.1	20.8
121905_1033b.jpg	306980	246131	19.82	18.6	24.7
121905_1034b.jpg	267854	206426	22.93	16.5	22.2
121905_1035a.jpg	283868	218526	23.02	17.8	23.2
Average			23.15	16.4	21.8

Figure 7: JPEG Compression via Brew application on the A970

Figure 7 shows that, across the range of images provided in the sample data set (presented in Table 1), the average compression ratio was 23 percent. The average time for compression was 16.4 seconds, and the average time for decompression was 21.8 seconds. While the timing results may seem to be somewhat slow, it is worth noting this appears to be as much to do with the the limitations running the code in a high level environment such as BREW. The second of the test platforms described above can be used to suggest how performance might be improved by running the Stuffit JPEG compression code natively.

File Name	Original Size	Compressed Size	Percent Saved	Time to Compress (in seconds)	Time to De-compress (in seconds)
121805_1223a.jpg	245092	199890	18.44	3.79	4.84
121905_0929a.jpg	234494	166716	28.90	3.71	4.70
121905_0953a.jpg	335337	240565	28.26	5.21	6.66
121905_0954a.jpg	304123	217245	28.57	4.67	5.94
121905_0954b.jpg	306782	245416	20.00	5.12	6.60
121905_1031a.jpg	380577	270469	28.93	5.77	7.46
121905_1031b.jpg	235791	191864	18.63	3.66	4.69
121905_1032a.jpg	267663	221691	17.18	4.10	5.25
121905_1032b.jpg	301310	239286	20.58	5.04	6.50
121905_1033a.jpg	235308	174767	25.73	4.08	5.28
121905_1033b.jpg	306980	246131	19.82	5.09	6.63
121905_1034b.jpg	267854	206426	22.93	4.33	5.50
121905_1035a.jpg	283868	218526	23.02	4.68	6.00
Average			23.15	4.56	5.85

Figure 8: JPEG Compression on PDA running Windows Mobile

The results presented in Figure 8 were generated by compressing the same set of images on a PDA device running Windows Mobile. The compressed size of the JPEG, in each case, remains the same. What can be seen, however, is a dramatic improvement in compression times.

While the numbers presented in Figure 7 represent the actual performance of the first generation StuffIt Wireless BREW application, Figure 8 offers a picture of what StuffIt Wireless compression performance might look like if StuffIt technology were integrated directly into the imaging software.

Figure 9 (below) presents an additional data set suggesting that it should also be possible to improve the compression ratio achieved. The current desktop implementation of StuffIt JPEG compression technology offers two compression “levels.” The user can either minimize compression time or maximize the degree of compression.

Compression Ratio Variation by JPEG Level

One factor known to have a significant impact on the compression efficiency of StuffIt Wireless is the JPEG quality level. In the context of JPEG images, the image quality level defines the amount of loss generated from the original image. The example in Figure 9 illustrates the extremes. The first image is compressed with JPEG level 100, the highest quality level, while image 2 is compressed at level 0, the lowest available setting.

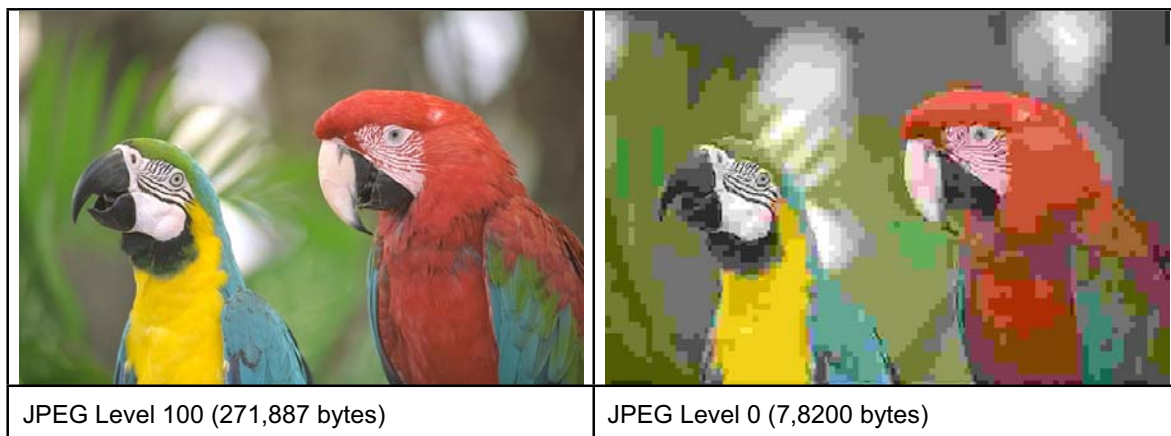


Figure 9: Kodim23 when converted to JPEG level 100 and JPEG level 0

As the performance of StuffIt's JPEG compression method is measured across the range of JPEG quality settings it can be seen that the compression ratio declines as the amount of image information increases. The information presented in Figures 10 and 11 illustrates the ways in which the compression ratio changes with the JPEG quality level.

JPEG Level	JPEG File Size	Compressed Size	Ratio
100	271,887	216,873	21%
90	77,329	56,538	27%
80	48,757	34,508	30%
70	37,812	25,989	32%
60	31,631	21,125	34%
50	27,754	18,111	35%
40	24,223	15,306	37%
30	20,620	12,543	40%
20	16,427	9,196	45%
10	11,638	5,303	55%
0	7,820	1,822	77%

Figure 10: Compression Ratios for Kodim23 from level 0 to level 100 (all sizes in bytes)

At the lowest JPEG quality level, StuffIt is able to achieve a 77% compression ratio, the caveat being that the source image is of extremely poor quality. The significant data point in Table 6 is that, even when the JPEG file contains the maximum amount of image information, StuffIt is still able to compress the image by 21%, delivering a result that is within the expected performance parameters.

Figure 11 illustrates the rate at which StuffIt compression ratios decline as the original JPEG quality is increased. For minimally-compressed JPEGs (in the 0-20 range), using StuffIt results in a significant compression gain. In the 20 to 80 range--the most common range of JPEG quality levels--there is little change in StuffIt compression efficiency.

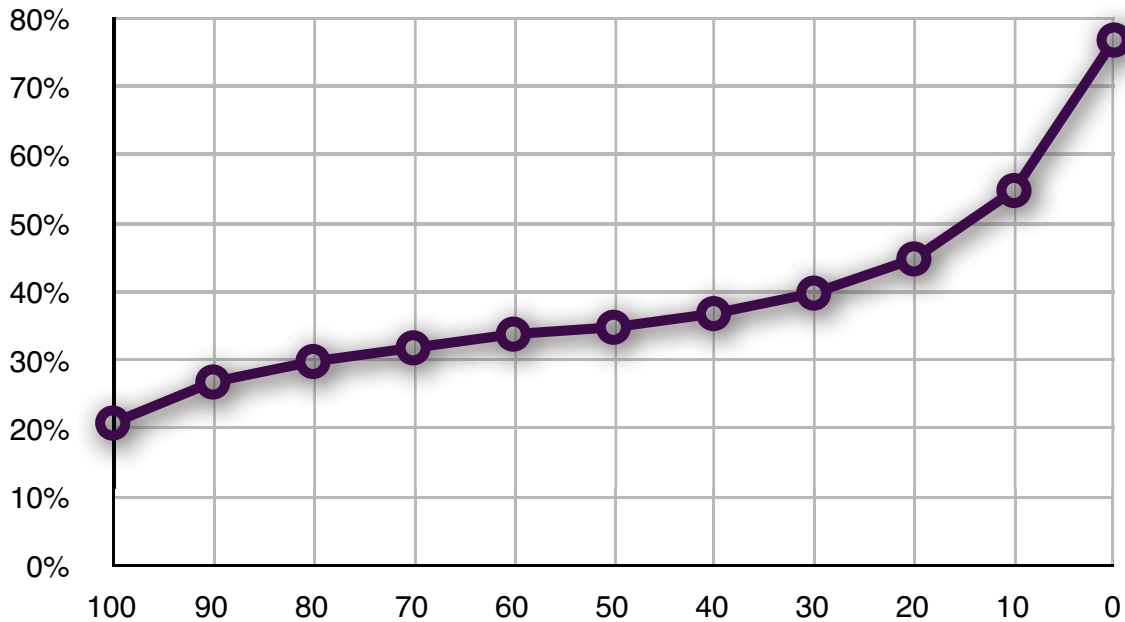


Figure 11: A line chart illustrating the compression ratio for Kodim23 at various JPEG Quality levels

StuffIt Lossless Image Format

StuffIt's ability to further compress JPEG images in a lossless fashion is an important attribute of StuffIt's JPEG compression, but this alone cannot correct the loss of image quality caused by the original use of the JPEG standard. Regardless of the quality level used, any time that an image is converted to the JPEG format, image information is lost.

To address this issue, StuffIt's image compression technology includes a 24-bit lossless compression method that bypasses JPEG completely. StuffIt Wireless allows users to store full-quality, 24-bit images within a file size that is equivalent to a maximum quality JPEG with no loss of image information. While StuffIt's lossless 24-bit compression provides similar functionality to existing file formats--such as TIFF and PNG--StuffIt is able to store the same image information in a file that is an average of 50 percent smaller than the equivalent TIFF or PNG.

StuffIt Lossless Compression Performance Results

In order to test the efficiency of StuffIt's lossless image compression technology, a set of reference images available on Kodak's website is used.

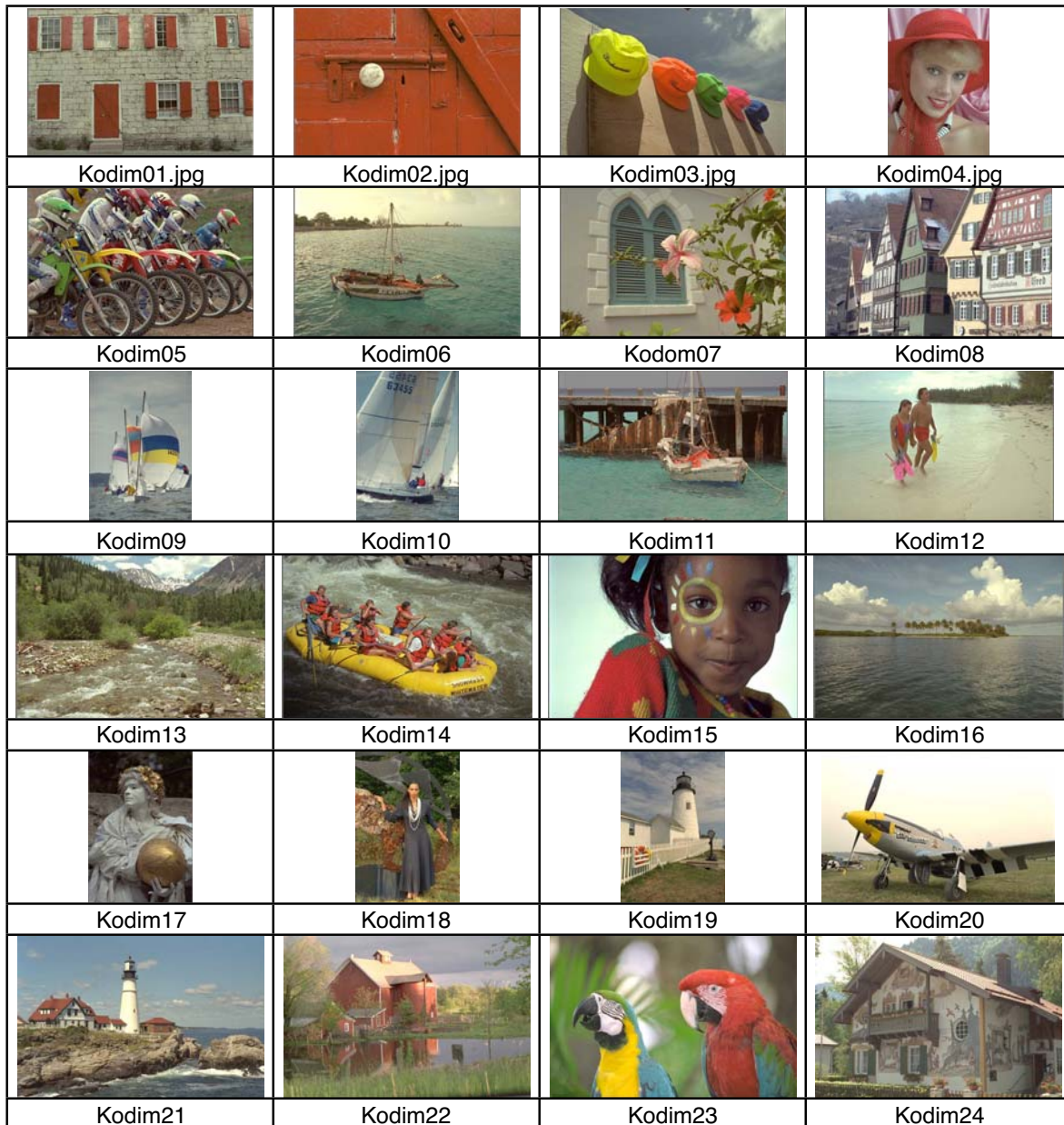


Figure 12: Kodak Reference Images

To test StuffIt's Lossless compression, the original .PNG files were first converted to bitmap format and then compressed with StuffIt Wireless.

File	BitMap (.bmp)	StuffIt Lossless	PNG	TIFF
Kodim01	1,179,702	395,732	736,501	847,605
Kodim02	1,179,702	321,394	617,995	726,568
Kodim03	1,179,702	278,725	502,888	578,584
Kodim04	1,179,702	344,155	637,432	741,834
Kodim05	1,179,702	446,769	785,610	917,568
Kodim06	1,179,702	373,849	618,959	703,938
Kodim07	1,179,702	304,547	566,322	642,958
Kodim08	1,179,702	304,547	788,470	981,245
Kodim09	1,179,702	316,997	582,899	659,494
Kodim10	1,179,702	328,366	593,463	685,355
Kodim11	1,179,702	358,035	621,023	716,280
Kodim12	1,179,702	305,127	531,024	607,782
Kodim13	1,179,702	491,848	822,712	973,990
Kodim14	1,179,702	411,739	692,201	793,285
Kodim15	1,179,702	331,895	612,582	730,701
Kodim16	1,179,702	312,594	534,247	605,267
Kodim17	1,179,702	324,963	602,078	680,780
Kodim18	1,179,702	450,052	780,947	914,233
Kodim19	1,179,702	370,363	671,476	767,113
Kodim20	1,179,702	294,238	492,462	552,468
Kodim21	1,179,702	372,878	637,051	725,661
Kodim22	1,179,702	414,250	701,970	836,188
Kodim23	1,179,702	335,196	557,596	665,477
Kodim24	1,179,702	391,132	706,397	827,587
Total Image Data	28,312,848	8,579,391	15,394,305	17,881,961

Figure 13: StuffIt Lossless Compression vs PNG and TIFF (Size in bytes)

Whether the files are taken individually, or the total image data is taken together, StuffIt provides a significant benefit over existing formats. In each case, the PNG, TIFF, or StuffIt image contains exactly the same image data as the uncompressed bitmap. The StuffIt Lossless Image provides the most efficient way to store this image data.

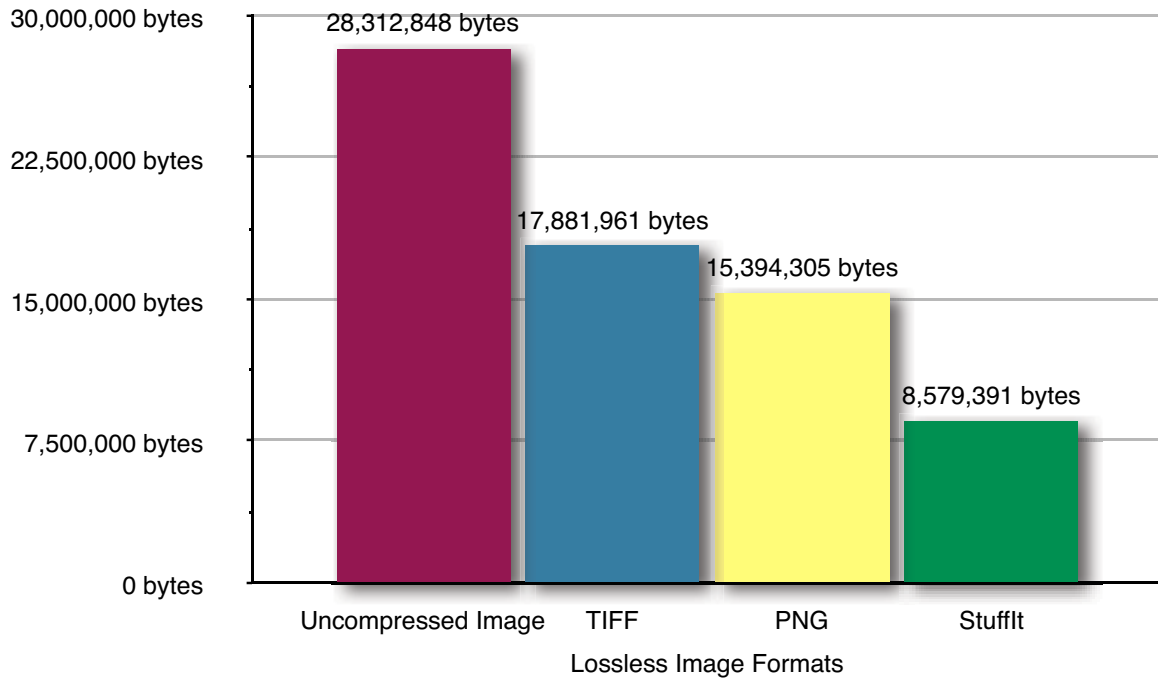


Figure 14: Total Image Data Comparison Between StuffIt, BMP, TIFF, and PNG

Figure 14 compares four image formats in terms of the total number of bytes required to store --with no loss-- the image data for all the pictures in the Kodak image set. Compared to the size of the PNG --the second best format-- StuffIt's lossless compression offers approximately a 45% advantage.

StuffIt Lossless Compression Performance: Variation by File Size

The following table illustrates how the efficiency of StuffIt's lossless compression method changes as the size of the original bitmap increases. In order to measure this, a single image was progressively reduced in size from an original image size of 1200x1024 to a final size of 640x480, or the dimensions of a VGA image. At each stop on the scale, the image was compressed with StuffIt's lossless compression method.


Dimension	BMP	
640x480	921,656 bytes	
800x600	1,440,056 bytes	
1024x768	2,362,424 bytes	
1200x1024	3,932,216 bytes	
	StuffIt Lossless	% Saved over BMP
640x480	406,428 bytes	56
800x600	652,090 bytes	55
1024x768	1,059,434 bytes	55
1200x1024	1,728,275 bytes	56

Figure 15: StuffIt Lossless Compression Variation by File Size

Figure 15 demonstrates how the compression ratio achieved by StuffIt does not significantly change as the size of the source bitmap file is scaled up or down. The benefit derived from deploying StuffIt Wireless, therefore, can be demonstrated in a variety of different environments; whether the source images are large or small StuffIt's image compression technology is able to deliver a substantial reduction in size.

StuffIt Lossless Compared to JPEG

The data in the above table provides an indication of how StuffIt’s lossless compression method compares to other 24-bit lossless image formats such as TIFF and PNG. In this section, the performance of StuffIt Lossless will be measured against JPEG. To obtain these results, a single image was again resized at various dimensions and then compressed. In this case, JPEG images were created in Adobe Photoshop. At each size, JPEGs were created using Photoshop’s default “Max” and “Medium” JPEG quality settings.

Dimension	BMP	StuffIt Lossless	JPEG Max	JPEG Med
640x480	921,656	406,428	321,612	81,190
800x600	1,440,056	652,090	502,354	112,437
1024x768	2,362,424	1,059,434	798,889	162,547
1200x1024	3,932,216	1,728,275	1,272,267	241,352

Figure 16: StuffIt Lossless Compression Compared to JPEG (sizes in bytes)

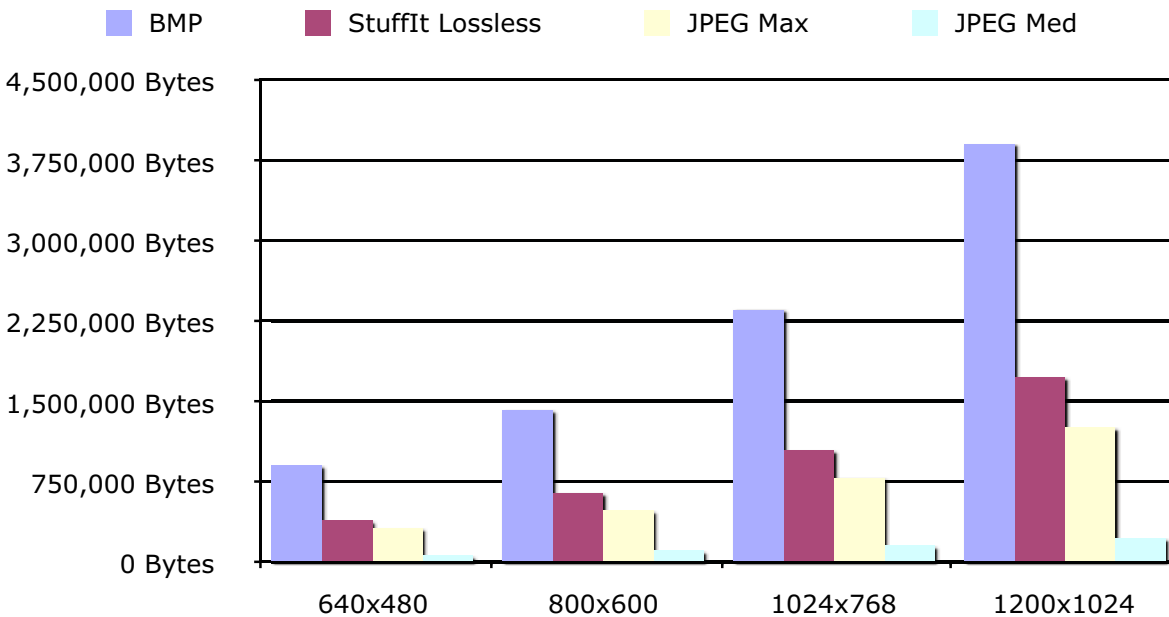


Figure 17: StuffIt Lossless Compared to JPEG

The figures above illustrate how the original size of the image has little impact on the compression ratios achieved: in all cases, the size of the high-quality JPEG image was ultimately an average of 66% smaller than the original bitmap (compared to an average of 55% for StuffIt lossless). Reducing the quality of the JPEG produced an average compression ratio of 93 percent compared to the original bitmap. Of course, it should be noted that the efficiency of StuffIt in each case can be improved by further compressing the JPEG images with StuffIt JPEG compression.

While, in each case, the performance of JPEG produces a smaller image than StuffIt's lossless compression method, it is worth noting that even at maximum quality the JPEG contains steps that alter the quality of the image. StuffIt is able to produce an image whose size is equivalent to that of a high quality JPEG while preserving exactly the image data of the original bitmap. The StuffIt version of the image contains exactly the same pixels as the bitmap, a statement that cannot be made even of the highest quality JPEG.

The StuffIt Image Format

The StuffIt Image Format best is described as the “container” used to store StuffIt compressed images. A StuffIt Image Format (or SIF) file can be used to store data resulting from the compression of a JPEG image or data from a compressed 24-bit image. A SIF reader, or a SIF-enabled application, therefore, is able to read the contents of a SIF file regardless of whether the SIF file was created from an existing JPEG file or created by directly compressing data from a camera sensor with a lossless algorithm.

A SIF file will contain a number of elements in addition to the compressed image data. An extensible set of metadata tags will allow information about the image to be available to a SIF reader without requiring the image data to be decompressed. These tags might include technical information about the image (such as the type of camera on which it was taken, or the particular aperture and shutter speed settings), or it might be user-created content such as a description or searchable key words. SIF images will also provide support for storing a thumbnail of the image as an attribute that can be accessed without requiring image data to be decompressed.

Compatibility and Interoperability

A JPEG compressed in StuffIt format cannot be opened by an existing JPEG implementation. In addition to providing embedded compression on handsets, StuffIt Wireless will also provide a solution that allows carriers to build a compatibility layer for StuffIt messages into their Multimedia Messaging Service (MMS) system. Currently, as MMS traffic passes through the carriers network, the job of formatting (transcoding) JPEG content so that it is viewable on the destination handset is performed by an MMS Center (MMSC) server. The StuffIt Wireless project proposes to insert an additional, but very similar, task at the MMSC level; when StuffIt-compressed content arrives as part of an MMS message, the server will first use the target profile to determine whether the destination client is StuffIt enabled. If it is not, the image will be converted back to a standard JPEG, leaving the message to be passed on in uncompressed format.

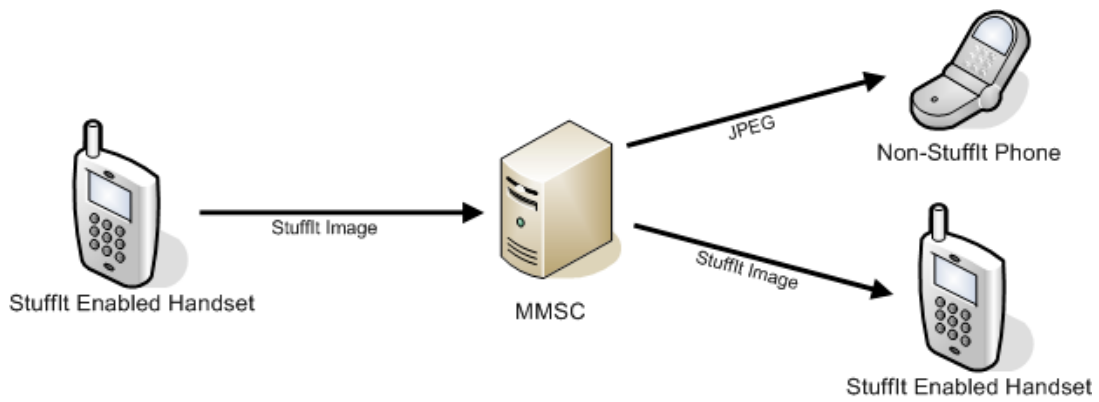
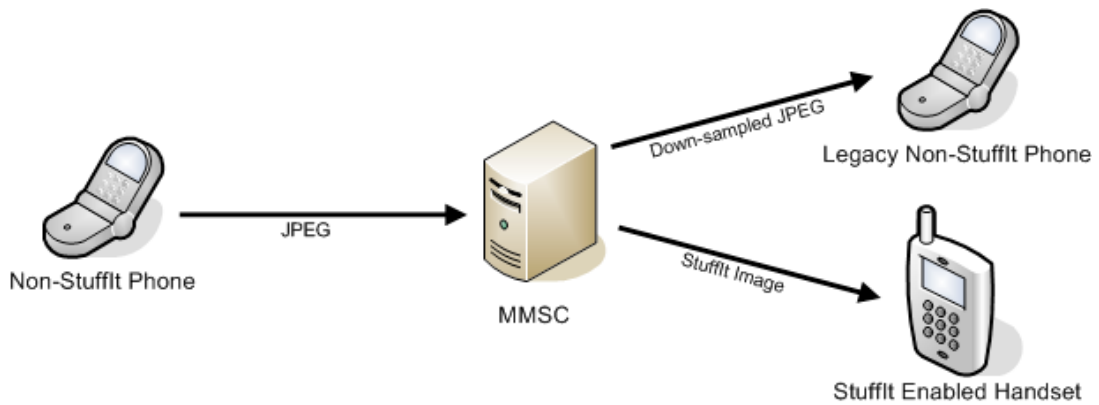


Figure 18: MMS messages from a Stuffit Phone

The MMSC infrastructure at the carrier level can also be used to compress MMS traffic flowing from a non-Stuffit phone to a Stuffit enabled device.



Summary

StuffIt Wireless provides a unique, cost-saving solution for both handset manufacturers and wireless carriers. In the face of a growing volume of MMS traffic, StuffIt Wireless provides a solution that enables carriers to get the most efficient possible use out of existing network infrastructure. The 20 to 30 percent size saving offered by StuffIt's JPEG compression component can be translated directly into a 20 to 30 percent bandwidth saving.

For handset manufacturers, StuffIt Wireless provides a range of benefits focused on the most efficient use of memory possible. StuffIt's image compression technology mitigates the increase in image file size driven by higher quality cameras. StuffIt allows more--and/or higher-quality--images to be stored within the same memory footprint, and StuffIt's ability to compress handset resources frees up more space for a users' files.

Contact Information

For more information about StuffIt Wireless contact:

W. Rick Wyand
VP of Sales - Wireless and OEM
Smith Micro Software, Inc.
51 Columbia
Aliso Viejo, CA 92656

949-362-2347 - Office
949-362-5723- Fax
913-220-5244-Mobile
rwyand@smithmicro.com