

---

**White Paper:**  
**TWAIN as an API for High Throughput Image Capture**

The TWAIN Working Group  
March 26, 1999

---



---

Contributors:  
Mark McLaughlin, Eastman Kodak Corporation

Jon Harju, JFL Peripheral Solutions

Chuck Mayne, Hewlett-Packard Company

Ron Vogel, Canon Information Systems

Dan Young, Xerox Corporation

---

# **TWAIN as an API for High Throughput Image Capture**

TWAIN Working Group  
25-Mar-1999

---

---

## **Abstract**

---

TWAIN is well suited as an API for image capture systems that produce large images, large numbers of images or both.

---

## **History**

---

TWAIN has been the API of choice for low volume image capture on Microsoft Windows and Apple Macintosh platforms since the early 90's; a time when personal computers had scant resources, compared to today (typical systems clocking around 25MHz, had 4MB of RAM and about 80MB of disk). An uncompressed bitonal 8.5" x 11" image at 100dpi, which uses about 117K of space, was a lot for these systems to handle. Yet people wanted to scan and work with images like this, as well as grayscale and color, and TWAIN was designed to help by providing efficient mechanisms for transferring images between the Source and the Application.

High volume image capture devices also existed in the early 90's, with some machines capable of delivering well over one hundred images per minute. These devices demanded fast systems capable of managing large numbers of images. The personal computers of the day were not capable of keeping up without specialized hardware support. Since UNIX was the platform of choice for workstations (for instance one popular machine ran at 50MHz, had 32MB of RAM and 400MB of disk space), they were typically selected to manage these high volume image capture devices.

Today, personal computers have sufficient performance to meet high volume capture needs and so does TWAIN. Since 1997 the TWAIN Working Group has expanded the TWAIN Specification to provide better programmatic control of high volume capture devices and digital cameras. This work has progressed because of the recognition that TWAIN performs well with these classes of devices.

---

## **Large Image Sizes**

---

TWAIN was designed from the start to manage large images; this was an essential requirement since even grayscale images can grow to be many megabytes in size.

Because of the small amount of RAM available to personal computers in the early 90's, TWAIN developed two ways of dealing with this amount of data. One way was for the TWAIN Data Source to write the images to disk. This method, called "File Transfer Mode," has the Source create an image file directly on disk, eliminating the need for transfer buffers between the Application and the Source. The Application then opens the file for viewing or processing, usually after the Source had been closed, freeing up even more memory for the Application.

The other method of transferring large images is "Buffered Memory Mode". This method is capable of moving the image in strips between the Application and the Source. If done properly it does not require any additional memory besides that needed to contain the image itself. Of course it has the advantage that it does not access the disk, which can result in a measurable performance benefit. Buffered memory mode has two other advantages: it is mandatory for all Sources, unlike file mode, which is optional, and Buffered memory mode also supports compression, which can be used to reduce the size of the images being transferred.

---

## Large Numbers of Images

---

High volume scanners and digital cameras are both capable of delivering large number of images in a single session. The limitations that used to prevent personal computers from supporting this kind of throughput no longer exist. Given TWAIN's popularity as an image capture API, it seems natural to turn to it when seeking to drive these kinds of devices. TWAIN is fully able to meet this need.

A TWAIN session is divided into two phases: configuration and image capture. The configuration phase identifies the capabilities of the Source and selects attributes of the capture session. Most of the TWAIN API is devoted to configuration, and most of the coding effort is here too, but the main part of a batch session is in the capture and transfer of images.

The image capture phase is composed of three basic steps: transfer, information collection and completion. Information collection may take place before or after the image has been transferred, or in some cases both before and after the transfer has occurred. The actual image transfer (DAT\_IMAGEMEMXFER), may occur in one or more steps. The completion step is required to tell the Source that all activity with the current image is concluded. Some samples follow:

This sequence is appropriate if ICAP\_UNDEFINEDIMAGE\_SIZE is FALSE, meaning that the Source knows the exact size of the image that will be generated prior to its capture. This example captures one image along with its associated attributes:

DG\_IMAGE / DAT\_IMAGEINFO / MSG\_GET

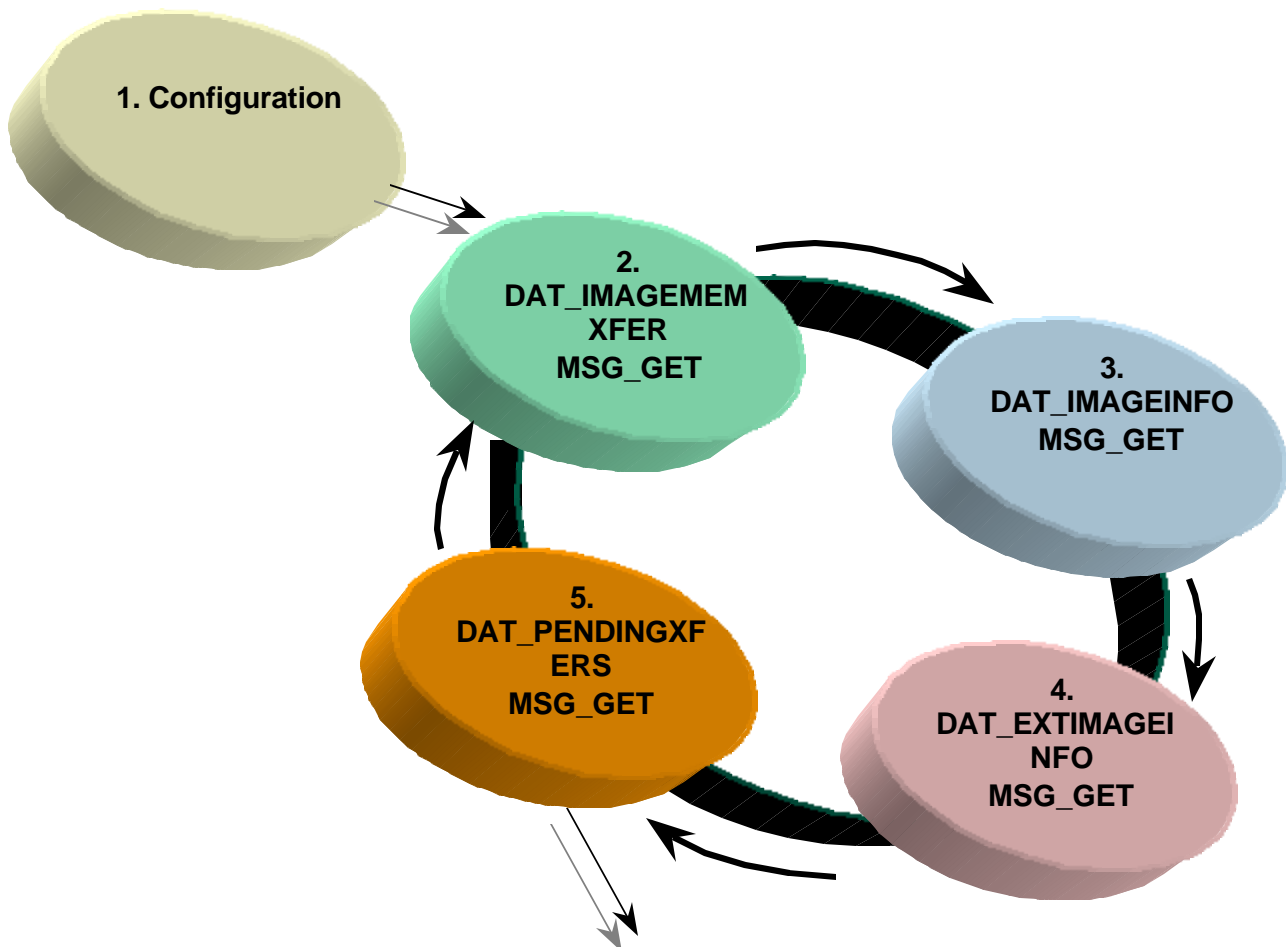
do

```
    rc = DG_IMAGE / DAT_IMAGEMEMXFER / MSG_GET
while (rc != TWRC_XFERDONE);
DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER
```

This sequence is appropriate if ICAP\_UNDEFINEDIMAGESIZE is TRUE, meaning that the Source does not know the exact size of the image that will be generated prior to its capture. This situation might arise with deskew correction or border removal. This example captures one image with its associated attributes, along with a request for extended information (such as barcode, deskew, or border removal information):

do

```
    rc = DG_IMAGE / DAT_IMAGEMEMXFER / MSG_GET
while (rc = TWRC_XFERDONE);
DG_IMAGE / DAT_IMAGEINFO / MSG_GET
DG_IMAGE / DAT_EXTIMAGEINFO / MSG_GET
DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER
```



Note that DAT\_EXTIMAGEINFO is not a required operation, and may not be supported by some Sources. Also note that these examples assume CAP\_AUTOSCAN being TRUE (which is an optional capability). If CAP\_AUTOSCAN is FALSE or not supported then we either intend to re-negotiate capabilities in the midst of the batch, or we are limited to page-on-demand mode. In either case the session cannot be expected to produce the same image throughput numbers as one where CAP\_AUTOSCAN is supported and set to TRUE.

---

## Large Numbers of Large Size Images

---

Given the two previous sections nothing special is needed to accommodate a device that is capable of producing large numbers of large images, besides acknowledging that such a device requires a higher end hardware system to properly support it.

---

## Optimizing Performance

---

TWAIN developers can code efficient transfer mechanisms for Sources and Applications without a great deal of effort. The following items suggest a number of guidelines:

- Native mode on a PC may not be the most efficient transfer mechanism, since scanned images typically need to be flipped about the x-axis in order to match their 0,0 origin to the BITMAP format. In the case of color images it the Source may also need to convert from RGB format to BGR. Also, native mode images must be transferred in an uncompressed format, which in the case of large color images can require a prohibitive amount of memory.
- Buffered memory mode is flexible enough to work with both memory poor and memory rich machines. This should be the transfer mechanism of choice.
- File mode is optional, and is not supported by the majority of Sources. It is, however, a potentially useful transfer mode for Sources that are capable of generating extremely large images, since the Source is in complete control of the process and can perform it in the most efficient manner possible.
- File mode may not be a good choice for some high volume capture Applications, since the impact of disk I/O is significant compared to operations performed entirely in memory. An example of this is an Application that performs inline image processing: the Source writes the image to disk, the Application reads it from disk, applies its image processing, and writes the final result to disk. The end result is three disk accesses with File mode, where with Buffered memory mode transfer there would have been only one.

- If using buffered memory mode, then use buffers sized to the Source, the system, and the desired throughput. DAT\_SETUPMEMXFER returns information about the most efficient way to use the Source. In general, a larger buffer needs fewer calls to perform the transfer, resulting in a faster operation. A small system might not have enough memory to support the needed throughput, in which case the Application has to decide if it has enough memory available to work efficiently.
- Minimize the number of TWAIN calls in the scanning loop. This is important if the Source is expected to transfer several images, either as a result of batch scanning, or as a dump of a digital camera's captured images. For example:
  - DG\_IMAGE / DAT\_IMAGEMEMXFER / MSG\_GET
  - DG\_IMAGE / DAT\_IMAGEINFO / MSG\_GET
  - DG\_IMAGE / DAT\_EXTENDEDIMAGEINFO / MSG\_GET (optional)
  - DG\_CONTROL / DAT\_PENDINGEXFERS / MSG\_ENDXFER
- For extremely large images, avoid use of native mode transfer. Use either buffered memory mode with the preferred transfer size, or file transfer mode, if it is supported. If compression is available, then use it.
- For high volume capture devices, try to transfer the entire image with as few calls as possible. A 24-bit 8.5"x11" image at 100dpi will use about 2.8MB of RAM. Transferring this image with a buffered memory size of 64K will take more than forty calls to DAT\_IMAGEMEMXFER to transfer the whole thing.
- For high volume capture devices, try to use compression if it is available. A reduction in size not only reduces the transfer time, it also reduces the amount of data that needs to be written to disk. Less disk I/O can result in a significant performance improvement, especially on today's machines, which can read and decompress an image faster than the time it takes to read an uncompressed image from disk.
- For high volume capture devices, check if the Source supports CAP\_AUTOSCAN, and if it does then set it to TRUE for optimal performance. In some cases this may only be permitted if CAP\_XFERCOUNT is equal to -1 (infinite).

\* The TWAIN name, logo and phrase "TWAIN- Linking Images with Applications" are trademarks of the TWAIN Working Group. All rights reserved.