1

Introduction

Chapter Contents

Need for Consistency

With the introduction of scanners, digital cameras, and other image acquisition devices, users eagerly discovered the value of incorporating images into their documents and other work. However, supporting the display and manipulation of this raster data placed a high cost on application developers. They needed to create user interfaces and build in device control for the wide assortment of available image devices. Once their application was prepared to support a given device, they faced the discouraging reality that devices continue to be upgraded with new capabilities and features. Application developers found themselves continually revising their product to stay current.

Developers of both the image acquisition devices and the software applications recognized the need for a standard communication between the image devices and the applications. A standard would benefit both groups as well as the users of their products. It would allow the device vendors' products to be accessed by more applications and application vendors could access data from those devices without concern for which type of device, or particular device, provided it. TWAIN was developed because of this need for consistency and simplification.

Elements of TWAIN

TWAIN defines a standard software protocol and API (application programming interface) for communication between software applications and image acquisition devices (the source of the data).

The three key elements in TWAIN are:

- **Application software**

  An application must be modified to use TWAIN.

- **Source Manager software**

  This software manages the interactions between the application and the Source. This code is provided in the TWAIN Developer's Toolkit and should be shipped for free with each TWAIN application and Source.

- **Source software**

  This software controls the image acquisition device and is written by the device developer to comply with TWAIN specifications. Traditional device drivers are now included with the Source software and do not need to be shipped by applications.

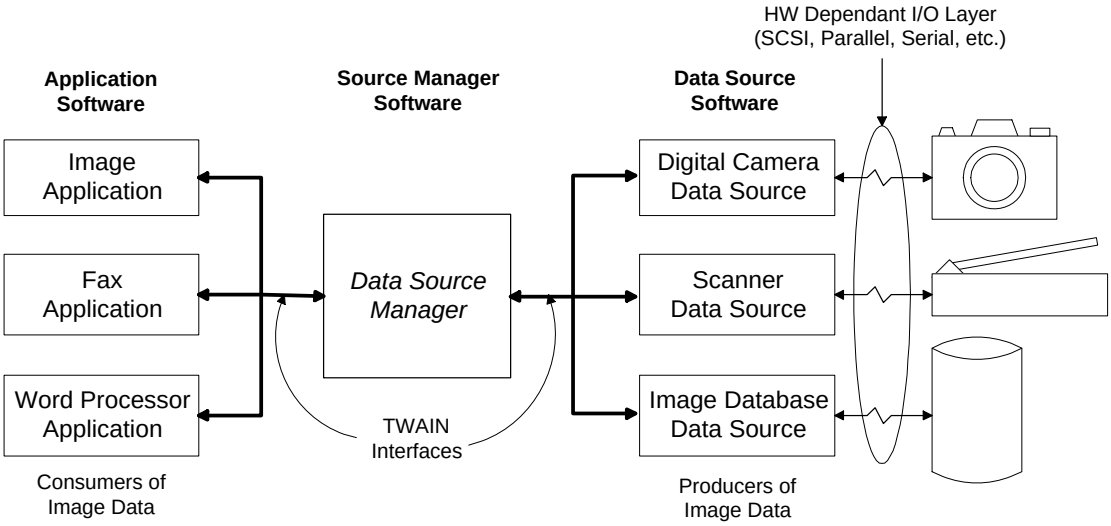

Figure 1-1 TWAIN Elements

Benefits of Using TWAIN

For the Application Developer

- Allows you to offer users of your application a simple way to incorporate images from any compatible raster device without leaving your application.
- Saves time and dollars. If you currently provide low-level device drivers for scanners, etc., you no longer need to write, support, or ship these drivers. The TWAIN-compliant image acquisition devices will provide Source software modules that eliminate the need for you to create and ship device drivers.
- Permits your application to access data from any TWAIN-compliant image peripheral simply by modifying your application code once using the high-level TWAIN application programming interface. No customization by product is necessary. TWAIN image peripherals

can include desktop scanners, hand scanners, digital cameras, frame grabbers, image databases, or any other raster image source that complies to the TWAIN protocol and API.

- Allows you to determine the features and capabilities that an image acquisition device can provide. Your application can then restrict the Source to offer only those capabilities that are compatible with your application's needs and abilities.
- Eliminates the need for your application to provide a user interface to control the image acquisition process. There is a software user interface module shipped with every TWAIN-compliant Source device to handle that process. Of course, you may provide your own user interface for acquisition, if desired.

For the Source Developer

- Increases the use and support of your product. More applications will become image consumers as a result of the ease of implementation and breadth of device integration that TWAIN provides.
- Allows you to provide a proprietary user interface for your device. This lets you present the newest features to the user without waiting for the applications to incorporate them into their interfaces.
- Saves money by reducing your implementation costs. Rather than create and support various versions of your device control software to integrate with various applications, you create just a single TWAIN-compliant Source.

For the End User

- Gives users a simple way to incorporate images into their documents. They can access the image in fewer steps because they never need to leave your application.

Note: TWAIN is supported on all versions of Microsoft Windows and Apple Mac OS X. TWAIN 2.x and higher includes support for Linux and 64-bit operating systems. Information about supporting TWAIN on 16-bit operating systems and older versions of the Apple Macintosh OS are no longer described in the current TWAIN specification. Please refer to version 1.9 of the Specification for support of older operating systems.

Creation of TWAIN

TWAIN was created by a small group of software and hardware companies in response to the need for a proposed specification for the imaging industry. The Working Group's goal was to provide an open, multi-platform solution to interconnect the needs of raster input devices with application software. The original Working Group was comprised of representatives from five companies: Aldus, Caere, Kodak Alaris, Hewlett-Packard, and Logitech. Three other companies, Adobe, Howtek, and Software Architects also contributed significantly.

The design of TWAIN began in January, 1991. Review of the original TWAIN Developer's Toolkit occurred from April, 1991 through January, 1992. The original Toolkit was reviewed by the TWAIN Coalition. The Coalition includes approximately 300 individuals representing 200 companies who continue to influence and guide the future direction of TWAIN.

The current version of TWAIN was written by members of the TWAIN Working Group including Adobe, Kodak Alaris, Inc., Fujitsu Computer Products of America, Hewlett-Packard Company, JFL Peripheral Solutions Inc., Ricoh Corporation, Xerox Corporation, and Lizardtech Corporation.

In May, 1998, an agreement was announced between Microsoft and the TWAIN Working Group which provided for the inclusion of the TWAIN Data Source Manager in Microsoft Windows 98 and Microsoft Windows NT 5.0.

During the creation of TWAIN, the following architecture objectives were adhered to:

- **Ease of Adoption —** Allow an application vendor to make their application TWAIN-compliant with a reasonable amount of development and testing effort. The basic features of TWAIN should be implemented just by making modest changes to the application. To take advantage of a more complete set of functionality and control capabilities, more development effort should be anticipated.
- **Extensibility —** The architecture must include the flexibility to embrace multiple windowing environments spanning various host platforms (Mac OS X, Microsoft Windows, Linux with KDE or Gnome, etc.) and facilitate the exchange of various data types between Source devices and destination applications. Currently, only the raster image data type is supported but suggestions for future extensions include text, facsimile, vector graphics, and others.
- **Integration —** Key elements of the TWAIN implementation "belong" in the operating system. The agreement between Microsoft and the TWAIN Working Group indicates that this integration into the operating system is beginning. TWAIN must be implemented to encourage backward compatibility (extensibility) and smooth migration into the operating system. An implementation that minimizes the use of platform-specific mechanisms will have enhanced longevity and adoptability.
- **Easy Application <-> Source Interconnect —** A straight-forward Source identification and selection mechanism will be supplied. The application will drive this mechanism through a simple API. This mechanism will also establish the data and control links between the application and Source. It will support capability and configuration communication and negotiation between the application and Source.
- **Encapsulated Human Interface —** A device-native user interface will be required in each Source. The application can optionally override this native user interface while still using the Source to control the physical device.