

TMSSequoia File ActiveX Control

User's Guide

IMAGE  *BASIC*

Diamond Head Software, Inc.
1217 Digital Drive Ste. 125
Richardson, Texas 75081
(972) 479-9205

COPYRIGHT NOTICES

Under the copyright laws, neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of Diamond Head Software, Inc., except in the manner described in the documentation.

This software product contains proprietary software components developed by a number of different software companies, referred herein as "Third Party Licensors". This documentation and the software that you purchased are protected by one or more of the following copyright notices:

Portions of this product,© 1994, 1995, 1996, 1997 Diamond Head Software, Inc. All rights reserved.
Portions of this product,© 1995, 1996 TMSSequoia, Inc. All rights reserved.

Company and product names mentioned in this documentation are trademarks or registered trademarks of their respective companies. Windows is a trademark and Microsoft is a registered trademark of Microsoft Corporation.

DIAMOND HEAD SOFTWARE INC. AND ITS THIRD PARTY LICENSORS MAKE NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE SOFTWARE. DIAMOND HEAD SOFTWARE, INC. AND ITS THIRD PARTY LICENSORS DO NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS OR OTHERWISE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY YOU. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME JURISDICTIONS. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

IN NO EVENT WILL DIAMOND HEAD SOFTWARE INC. OR ITS THIRD PARTY LICENSORS AND/OR THEIR DIRECTORS, OFFICERS, EMPLOYEES OR AGENTS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, AND THE LIKE) ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE EVEN IF DIAMOND HEAD SOFTWARE INC. OR ITS THIRD PARTY LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU. Diamond Head Software Inc.'s and its Third Party Licensors' liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort (including negligence), product liability or otherwise), will be limited to \$50.

Contents

Chapter 1 : Introduction	1
Introduction to the TMSSequoia File Control.....	1
Purpose of the TMSSequoia File Control.....	1
Definitions of Commonly Used Terms.....	2
Supported File Formats.....	2
Linking ImageBASIC Controls.....	3
Licensing Configuration and Verification.....	5
Chapter 2 : Creating, Modifying and Saving Documents	9
Creating and Modifying a Document.....	9
Loading an Image from File.....	10
Loading an Image from Memory.....	11
Modifying Pages in an Existing Document.....	12
Document and Page Information.....	13
Image Data Output.....	14
Sending Images to Other Controls.....	14
Saving Images to File.....	15
Chapter 3 : Reference	23
Properties, Methods and Events.....	23
Appendix A : File Formats and Color Files	49
File and Compression Formats.....	49
Image File Formats.....	50
Compression Types.....	51
Color Limitations of File Formats.....	52
Comparative Charts of Colors Supported by File Format.....	52
Index	55

Chapter 1 : Introduction

Introduction to the TMSSequoia File Control

Purpose of the TMSSequoia File Control

Each ImageBASIC component performs some specialized action such as image display, scanner control, or OCR. The TMSSequoia File control performs the file input and output. Because most other ImageBASIC controls cannot directly access images saved to file, the ultimate source of images will usually be either a File control or a Scan control.

The TMSSequoia File control is designed to provide the application developer the ability to create, manipulate and save documents formed from one or more separate image files or image sources. After its formation by the File control, the document can then be referenced as a single source by other ImageBASIC controls. The formation of the virtual document includes the options to insert, delete and reposition pages within the document.

Image Input Options

The File control can form a virtual document by several methods. For all methods, the source of the incoming images may be either a file or another ImageBASIC control:

- The incoming image may be used to form a new document after clearing any existing document.
- The image may be appended to the existing document
- The image may be inserted at an arbitrary point within the existing document.

Image Output Options

Once a document is formed, its constituent pages can be output either to another ImageBASIC control or written directly to file.

- The **SaveDocument** and **SavePage** methods write the current document and page, respectively, to file.
- The **Save** and **SaveRegion** methods save the image page and region, respectively, from the **ImageDataSource** without changing the current document.

- Other ImageBASIC controls can access document pages by setting the recipient control's **ImageDataSource** property to the TMSSequoia File control, as illustrated here for the Nestor ICR control:

```
Nestor1.ImageDataSource = TMSFile1.Link
```

After an image has been read into memory by the File control, any other ImageBASIC component can then use that image, even if the underlying engine could not otherwise understand the file format.

Definitions of Commonly Used Terms

The following terms are used throughout this manual with these definitions:

<i>document</i>	One or more image pages currently referenced by the File control; a document may be composed of pages from one or more sources including image files and other ImageBASIC controls
<i>file</i>	A valid image file on a Windows-accessible storage medium
<i>page</i>	A single image contained within a file or document; once added to a document, each page is no longer directly associated with its original source; as pages from each additional image source are added to an existing document, the entire collection of pages, regardless of their original source, can be referenced as a single source

Supported File Formats

The following file formats are supported by the TMSSequoia File control. Some of these file types are supported only for reading, while some are supported for both reading and writing. The read-only file formats are indicated in the list below:

BMP (read only)
 CALS Type1
 JPEG (read only)
 PCX
 TIFF Group 3 with 1-D Option (Standard Group 3)
 TIFF Group 3 with 2-D Option
 TIFF Group 3 Modified Huffman
 TIFF Group 4
 TIFF Packbits
 TIFF Uncompressed

BMP is a common Windows format for storage of relatively small, low resolution color images. Because BMP files do not generally perform any compression, the files can be very large and have long access times if the image is complex or large.

CALS is a U.S. government file format. It is similar to TIFF in that the file is composed of header information followed by the compressed image data. CALS Type I supports only single page bitonal files and uses the CCITT Group 4 algorithm to compress the image data.

PCX is a commonly used Windows format for storing paletted images. PCX supports only single page files of up to 256 colors (paletted).

The *TIFF* formats are the most common in document imaging. Group 4 typically offers the best compression ratios for bitonal images and is the default for output from the TMSSequoia File control.

More description of these file formats and the level of color that each supports may be found in 'Appendix A : File Formats and Color File's on page 49.

Linking ImageBASIC Controls

Types of Inter-Control Communication

The ImageBASIC controls communicate amongst themselves in a unique manner to get information and data to the control that needs it. There are two times that this communication is performed:

- 1) Image data is routed to another control
- 2) Services are performed

Each ImageBASIC control can be either a source or a recipient of one or both of these types of communication. The act of specifying the source and recipient of this communication is called *linking* the controls.

All ImageBASIC controls that can accept image data from another ImageBASIC control have a property called **ImageDataSource**. The **ImageDataSource** property specifies the ImageBASIC control that is the source for all incoming images.

- For example, Display and OCR controls cannot typically read image files directly from disk.
- The TMSSequoia File control can read the image files into memory where they can be retrieved by the Display control.

- Therefore, the source of images for the Display control is the TMSSequoia File control.

The communication between the controls is enabled by setting the Display control's **ImageDataSource** property to specify the TMSSequoia File control.

Some ImageBASIC controls offer services other than image processing. The communication for these controls is enabled in a similar manner, but through properties other than the **ImageDataSource** property.

- For example, the Annotation control does not process image data and, therefore, does not have an **ImageDataSource** property.
- Instead, the Annotation control performs a service -- creating and maintaining annotations. The Annotation control must have a Display control on which to show its annotations.
- The Annotation control's **DisplaySource** property must specify the Display control that will be performing this function.

Linking at Design Time

As each ImageBASIC control is added to a Form at design time, its **ImageDataSource** property will be automatically set to an ImageBASIC control, if one already exists on that Form.

The assignment may be changed at design time by selecting from the drop-down list of available ImageBASIC components. This list is shown with the **ImageDataSource** property in the Properties Window or Object Inspector. The same list of ImageBASIC controls is shown for the **DisplaySource** property, the **RegionSource** property, and all other...**Source** properties.

Linking at Runtime

As each instance of an ImageBASIC control is created, a unique identification string is calculated for that control. This string, or Link ID, is stored in the **Link** property of each control. This property is not visible at design time and cannot be changed by the application. The unique Link ID is calculated each time a control is created, whether it is created statically at design time or dynamically at runtime.

The assignment of a source control is made by setting the Source property of the receiving control -- perhaps the **ImageDataSource** of a Display control -- to the **Link** property of the source control. For example, for a Display control to accept image data from a TMSSequoia File control, set the **ImageDataSource** property as shown here for Visual Basic:

```
TMSDispl1.ImageDataSource = TMSFile1.Link
```


Other forms of inter-control communication are established in the same manner. For example, allowing an Annotation control to display its objects on a TMSSequoia Display control is performed by setting the Annotation control's **DisplaySource** property as shown here:

```
Annotel.DisplaySource = TMSDispl.Link
```

Licensing Configuration and Verification

In order to run, each ImageBASIC control must be able to verify the presence of a valid license token. These license tokens are stored on either a hardware key (shipped with each toolkit) or in a licensing database (the standard runtime distribution format).

Utilizing these tokens, licensing in ImageBASIC is based on enabling a set number of concurrent seats. Each license token allows a single PC to run any number of instances of a single control. For example, a single token for a Display will allow one workstation to concurrently run multiple applications, each of which employs any number of Display controls.

A unique token is required for each different type of ImageBASIC component that you have licensed:

- There is a special type of token for the TMSSequoia Display control, another for the TextBridge control, another for the ScanFix control, and yet more token types for each additional control. The 16-bit and 32-bit versions of each control are also licensed separately.
- Each one of these licenses also comes in two varieties: *runtime* and *development*.

As suggested by the names, runtime licenses are necessary for an executable to function, and development licenses are necessary to develop an application using ImageBASIC.

A design time license will function as a runtime license, removing the need to add runtime tokens for application testing during development.

Where the Licenses Tokens are Kept

ImageBASIC will find licenses stored in either one of two locations -- in a licensing database or on a hardware key.

- The hardware key is plugged into a parallel port on the computer using ImageBASIC and will be automatically found each time an ImageBASIC component is used.

Note: When using Windows NT, the parallel port must be configured using the Sentinel drivers that are shipped with ImageBASIC.

- The licensing database must be created on the site where it will be used and may not be moved from the location in which it is installed.

The inability to move a licensing database is one of its protection features. This attachment to a single location is formed when the licensing database is first created. Although it may not be moved once created, the licensing database can be written to a network drive to which all the machines running ImageBASIC have access.

A file called IMGBASIC.INI must be on each of these networked machines. This file contains an entry pointing to the location of the licensing database. ImageBASIC can now search the licensing database for an available copy of each license it needs to run.

The licensing database may also be created on a local drive, activating ImageBASIC on only that one machine. The same INI file is still necessary to pinpoint the location of the database.

How the License Tokens are Used

As each application that uses ImageBASIC is initiated, or the control is loaded into the development environment, the ImageBASIC licensing server attempts to find the proper token for each component.

For example, if an executable has been developed that uses ImageBASIC to display existing images, and then calls on TextBridge to perform OCR on that image, that application must be able to find one *Display runtime license*, one *File runtime license* and one *TextBridge runtime license*.

- If the application is successful in finding these licenses, it will load normally and function exactly as it was programmed.

When the application finds these licenses and is thereby informed that the correct licenses are available, it simultaneously locks the licenses so that no other application can use the same licenses to run at the same time.

If two copies of these same licenses are available, another computer will be able to run the same application and will then lock the second license.

- If the application cannot verify the presence of the required tokens, the ImageBASIC components will not operate.

The **Active** property of the controls that did not find license tokens will be set to False. The state of the **Active** property can be verified during Form Load.

A runtime error that can be trapped during Form Load will also occur.

Licensing Token Release

When an application that has locked one or more tokens terminates normally, the tokens are released and can immediately be taken by another user if a network licensing database is being used. This is the process by which concurrent licensing for any number of seats may be enabled.

- If the application ends abnormally -- the user might reboot or a concurrently running Windows application might lock up -- then the release of the licenses is conditional on the network or disk operating system.
- For Novell networks, the default time for releasing a file lock after a connection is lost is 5 (five) minutes.

For other networks, your system administrator should be able to tell you how long the NOS takes to release the lock.

The system administrator should be able to change the default release time to satisfy your particular needs.

- If the licensing database has been installed on a stand-alone machine, the locks are immediately released and will again be available when the application is started again.

Chapter 2 : Creating, Modifying and Saving Documents

Creating and Modifying a Document

Making a Document

In order for the File control to output any image information either to file or to another ImageBASIC control, a document must be created. The next few pages detail the creation and modification of documents.

- Creating a document can be as simple as loading a single file or a single page from another ImageBASIC control.
- Creating a document can also be as complex as loading several files, appending some to the end and inserting others in the middle of the document, and then replacing some of those pages with new pages, perhaps from a scanner, followed by rearranging the pages in this complex document.

Refer to the following sections titled "Loading an Image from File" and "Loading an Image from Memory" for details on creating a document.

For details on the output options once a document is available, refer to "Image Data Output" on page 14.

Selecting the Active Page of a Document

After a document is created, you can select a single page as the current, or active, page:

- At all times, the **PageCount** property reports the total number of pages in the current document.
- Select a single active page of the current document by setting the **PageIndex** property to an integer value from 1 to **PageCount**, inclusive.
- The active page, as selected through the **PageIndex** property, will be the page that is available to any other ImageBASIC control that is linked to the document control. The active page is also the point at which new images will be inserted and may be individually saved.

To link another control to the File control, set the recipient control's **ImageDataSource** property to specify the link ID of the File control.

The Link ID is a unique string calculated for each control as it is created. The Link ID is reported in the **Link** property.

For example, to display the active page in a TMSSequoia Display control that is on a different Form, link the controls together as illustrated here:

```
TMSDispl1.ImageDataSource = Form1!TMSFile1.Link
```

Several properties report information about the current page. This information ranges from the resolution and dimensions of the image to its color definition. Refer to Document and Page Information on page 13 for a complete list of these informational properties.

Saving Changes to a Document

While a document is held in memory, no changes made to that document will be saved to file. All changes such as appending, moving or deleting pages will be maintained in memory until the document or its constituent pages are explicitly saved. The entire current document can be saved through the **SaveDocument** method; the current page can be saved individually through the **SavePage** method.

Loading an Image from File

An existing image file may be opened and its contents made available through several methods. Depending upon whether you wish to start a new document or add to an existing document, use one of these methods to load the file:

Clear	Removes all current document information from memory and resets all document and page description properties to defaults. This method accepts no parameters: <code>TMSFile1.Clear</code>
-------	---

Before executing this method, it is generally advisable to query the **Dirty** property to determine if the image has been modified since loaded.

LoadFile	Clears the current document, if any, and starts a new document containing all pages from the specified file. This method accepts a single parameter of the name of the file to load. The filename may include a relative or absolute path: <code>TMSFile1.LoadFile "c:\images\img001.tif"</code>
----------	---

AppendFile	Adds all pages in the specified file to the end of the current document. This method accepts a single parameter of the name of the file to load. The filename may include a relative or absolute path: <code>TMSFile1.AppendFile "c:\images\img001.tif"</code>
------------	---

InsertFile Inserts all pages of the specified file beginning at the current **PageIndex** in the current document. This method accepts a single parameter of the name of the file to load. The filename may include a relative or absolute path:

```
TMSFile1.InsertFile "c:\images\img001.tif"
```

Note: When opening multiple-page files, images are loaded into memory only when explicitly instructed. For example, suppose you are reading a 4 (four) page file and set **PageIndex** to 3. The only pages actually read into memory are page 1 (one), because it is automatically loaded when the file is selected, and page 3 (three). Pages 2 (two) and 4 (four) are not read into memory until necessary.

Note: After a new Document is started, any changes to that document such as appending or inserting additional pages will be maintained in memory. The changes must be explicitly saved through the **SaveDocument** or **SavePage** method.

Note: To open a large format image page, an additional license is required. The determination of whether a given image page is large format is based on the size of the uncompressed image data as calculated by the following formula:

$$(\text{Pixel_Width} * \text{Pixel_Height} * \text{Bits_Per_Pixel}) / 8$$

If the size of the uncompressed image data exceeds 2,103,750 bytes, a large format license is required to open the image. This size limitation calculates to a bitonal double letter page (11"x17") at 300 DPI.

This formula is applied to each page individually as it is loaded. Therefore, if the first page loaded from a file is too large, the file cannot be loaded. The size of the compressed page or the size of the disk file are not relevant to this calculation.

Loading an Image from Memory

The File control can accept images from both an existing image file and from another ImageBASIC control through the **ImageDataSource** property. The File control's **ImageDataSource** property can specify any other ImageBASIC control that can serve as a source. The controls that typically supply image data to the File control are Display controls and Scan controls.

The **ImageDataSource** property is set as illustrated below, linking the File control to a Display control that is on another Form:

```
TMSFile1.ImageDataSource = Form2!TMSDispl.Link
```

Any image shown in the specified Display control will be available to the File control. The image from one of these controls may be added to the existing document by executing one of the following methods:

Clear	Removes all current document information from memory and resets all document and page description properties to defaults. No parameters are required for this method: <code>TMSFile1.Clear</code>
LoadPage	Clears the current document, if any, and starts a new document with the single page available from the control specified in the ImageDataSource property. No parameters are required for this method: <code>TMSFile1.LoadPage</code>
AppendPage	Adds the page available from the control specified in the ImageDataSource property to the end of the current document. No parameters are required for this method: <code>TMSFile1.AppendPage</code>
InsertPage	Inserts the page available from the control specified in the ImageDataSource property at the current PageIndex . No parameters are required for this method: <code>TMSFile1.InsertPage</code>

Note: After a new Document is started, any changes to that document such as appending or inserting additional pages will be maintained in memory. The changes must be explicitly saved through the **SaveDocument** or **SavePage** method.

Modifying Pages in an Existing Document

The TMSSequoia File control can arrange, delete, copy, and move pages within the current document. For information on creating a new document rather than arranging the pages of the current document, refer to the following sections:

"Loading an Image from File" on page 10

"Loading an Image from Memory" on page 11

Page management operations are performed through the following methods:

CopyPage	Makes a copy of the page at the current PageIndex and inserts the copy at the specified index position. The index of all subsequent pages is modified and the PageCount property is updated. This method accepts a single integer parameter specifying the position of the new page in the document: <code>TMSFile1.CopyPage index</code>
DeletePage	Deletes the document page at the current PageIndex . No parameters are required for this method: <code>TMSFile1.DeletePage</code>

MovePage	Moves the document page at the current PageIndex to the specified position. This method accepts a single integer parameter specifying the position of the new page in the document: <code>TMSFile1.MovePage <i>index</i></code>
ReplacePage	Accepts the image from the ImageDataSource and replaces the current document page with the new page. No parameters are required for this method: <code>TMSFile1.ReplacePage</code>
ReplaceRegion	Accepts only the current Working Region from the control specified in the ImageDataSource property and replaces the current document page with the region as a new page. No parameters are required for this method: <code>TMSFile1.ReplaceRegion</code>

Note: After a new Document is started, any changes to that document such as appending or inserting additional pages will be maintained in memory. The changes must be explicitly saved through the **SaveDocument** or **SavePage** method.

Document and Page Information

As each image page is loaded into memory, several read-only properties are populated with information about that image or the document. In a multiple-page document, any given page is selected by setting the **PageIndex** property. The informational properties are as follows:

PageCount	Reports the total number of pages in the document. The PageIndex property can be set to load any of these pages.
Dirty	Reports True if the document has been modified since the initial image or file was loaded. Reports False if no changes have been made to the document.
InputFileFormat	Reports the type of the input file. Will report a string similar to one of the following values: BMP CALS JPEG PCX TIFF

PageBitsPerSample	Reports the bits per sample value for the currently loaded image.
PageByteWidth	Reports the number of bytes of data describing the width of the image.
PageHeight	Reports the pixel height of the image page specified in PageIndex
PagePhotoInterp	Reports the photometric interpretation value of the image page specified in PageIndex
PageSamplesPerPixel	Reports the samples per pixel value of the image page specified in PageIndex
PageWidth	Reports the pixel width of the image page specified in PageIndex
PageXRes	Reports the horizontal resolution in DPI of the image page specified in PageIndex
PageYRes	Reports the vertical resolution in DPI of the image page specified in PageIndex

Image Data Output

The File control can supply image data to other ImageBASIC controls, or it can write the images to file, possibly accompanied by annotations. Annotations can be saved only to TIFF files.

Sending Images to Other Controls

When any document is defined within the File control, the individual pages of that document are available for output either to file or to another ImageBASIC control. Refer to 'Creating and Modifying a Document' on page 9 for a description of this process.

- 1) To select a single page of a document as the active page, set the **PageIndex** property to any integer value between 1 and the total number of image pages in the file as reported in the **PageCount** property.
- 2) The specified page will then be sent to any other ImageBASIC control that is linked to the File control. The ImageBASIC controls that can accept this image page must have their **ImageDataSource** property set equal to the **Link** property of the File control.

For example, the following code segment will link a Pixel Display control, load a file and then display the second page of the newly created document:

```
PixDisp1.ImageDataSource = TMSFile1.Link
TMSFile1.LoadFile "c:\images\form1.tif"
TMSFile1.PageIndex = 2
```

Saving Images to File

When saving to file, the image that is written may be obtained from either of two locations:

1) The current document

Saving image pages from the current document is the preferred technique for using the File control. Any changes made to an image page by another ImageBASIC control should be updated in the File control. These modifications may include annotations or cut and paste operations.

The modified page should be placed in the document that is being maintained by the File control. Adding the modified pages may require inserting the new pages or replacing an existing page.

For details on inserting and replacing pages, refer to 'Loading an Image from Memory' on page 11 and 'Modifying Pages in an Existing Document' on page 12.

Pages from the updated document can then be saved using either of these two methods:

SaveDocument Saves all pages of the current document.

Important: All pages must be of the same color definition; e.g., all must be bitonal, or all must be 256-level gray.

SavePage Saves only the image page at the current **PageIndex**

2) The current **ImageDataSource**

In order to maintain compatibility with code written for earlier revisions of the ImageBASIC File controls, two methods will save the image page currently available through the **ImageDataSource** without modifying the document itself. This behavior is identical to the behavior of these methods in prior revision of the controls:

Save Saves the entire image page available from the ImageBASIC control specified in the **ImageDataSource** property.

SaveRegion	Saves only the Working Region of the image page available from the ImageBASIC control specified in the ImageDataSource property. Does not affect the current document.
------------	---

Save the Current Document to an Image File

After creating a document of one or more pages, the File control can save either individual pages or the entire document as a single file. For details on creating a document, refer to 'Creating and Modifying a Document' on page 9.

To save a page or an entire document to file, follow these steps:

- 1) Specify the Page to Save
- 2) Set Output Parameters
- 3) Write the File to Disk

Sample code illustrating the updating of a document with an annotated page and then saving the document may be found in the section titled 'Example: Saving an Image File with Annotations in the File Header' on page 18.

1) Specify the Page to Save

If saving a single page, select a single page from the existing document by setting the **PageIndex** property. This property may be set to any integer value between 1 and the number of pages in the document as reported in the **PageCount** property. For example, to select the second page of the document, set **PageIndex** as shown here:

```
TMSFile1.PageIndex = 2
```

When saving the entire document, it is not necessary to select a page. The entire document will be saved without regard to the value of the **PageIndex** property.

2) Set Output Parameters

When saving a file, the following properties must be set before calling one of the save methods:

OutputFileFormat	Specifies the type of file to create. The enumerated options for this property are as follows:
2	CALS Type1
12	PCX
18	TIFF Packbits
20	TIFF Group 3 1-D
21	TIFF Group 3 2-D

	23	TIFF Group 3 Modified Huffman
	24	TIFF Group 4
	28	TIFF Uncompressed
OutputFileAppend		<p>If True, and the file already exists, the output will be appended after the last page in the file. Of the supported file types, TIFF allows the storage of multiple pages in a single file.</p> <p>If False, and the output file already exists, the existing file will be overwritten.</p>
OutputScalingDen		<p>Specifies the source half of a ratio defining the input pixel dimensions relative to output pixel dimensions. Used in conjunction with OutputScalingNum to scale output data. Defaults to one (1).</p> <p>For example, setting OutputScalingDen to 2 and OutputScalingNum to 1 will create an output file of one-half the resolution and pixel dimensions.</p>
OutputScalingNum		<p>Specifies the destination half of a ratio defining the input pixel dimensions relative to the output pixel dimensions of an image as it is saved. Used in conjunction with OutputScalingDen to scale output data. Defaults to one (1).</p>
OutputBitsPerSample		<p>Specifies the bits per sample of the output file; if 0, the current value of the image data is used.</p>
OutputSamplesPerPixel		<p>Specifies the samples per pixel of the output file; if 0, the current value of the image data is used.</p>
OutputPhotoInterp		<p>Specifies the photometric interpretation of the output file. Refer to 'Color Limitations of File Formats' on page 52 for information on the color levels supported by each file format and the definition of color using bits per sample, samples per pixel, and photometric interpretation.</p>

3) Write the File to Disk

Depending upon whether you wish to save the entire document or a single page of the document, one of the following methods must be executed to write the image file.

SaveDocument Saves all pages of the current document, applying the same output parameters listed above to all pages. If the specified output file type does not support multiple page files, an error

will be generated. Accepts a single parameter specifying the file name, with or without path, of the file to write:

```
TMSFile1.SaveDocument  
"c:\images\document.tif"
```

Note: All pages of the document must be of the same color definition; i.e., bits per sample, samples per pixel and photometric interpretation must be identical for all pages.

SavePage Saves only the page specified in the **PageIndex** property. All output parameter properties, as listed above, are applied to this single page. Accepts a single parameter specifying the file name, with or without path, of the file to write:

```
TMSFile1.SavePage "c:\images\page.tif"
```

Example: Saving an Image File with Annotations in the File Header

Each image page and the associated data with that page, such as any annotation objects, are all managed within ImageBASIC as a single object. All information that is maintained in addition to the image itself is referred to as sideband data.

Each control that accepts an image maintains its own copy of the image data and its associated sideband data. Therefore, changes made to this image structure by one control must be explicitly updated in any recipient control.

The sideband data is an integral part of the image data structure and is always available with the image.

- Any change to either the image or the sideband data is considered a new image page. Therefore, when a Display control accepts a page from an File control, and the Annotation control is used to add annotations to that page, the Display control is now holding a page that is different from the one held by the File control.
- To save the new annotations, the page held by the File control must be replaced with the page held by the Display control.

For this purpose, the File control has a method called **ReplacePage**. The **ReplacePage** method accepts the page from an ImageBASIC control, in this case a Display control, and replaces the current page.

The File control's document has now changed and should be saved as any other changed image.

The following code snippet provides an example of the general flow of image data as it relates to saving annotations in a TIFF header:

```
' link the Display control to a File control to  
' display an image
```

```

TMSDispl.ImageDataSource = TMSFile1.Link

    ' link the Annotation control to the Display
    control;
    ' this Display control will show all annotations
    made
    ' by the Annotation control
Annotel.DisplaySource = TMSDispl.Link

    ' load an image; for this example, assume a multi-
    ' page TIFF; the first page of the file will be
    ' displayed
TMSFile1.LoadFile "c:\images\multpage.tif"

    ' load and display the second page of the file
TMSFile1.PageIndex = 2

    ' by some technique, add one or more annotation
    ' objects to the image; for this example, add a new
    ' Text object and refresh the Display control
Annotel.CreateText 100, 100, "This is a Text annotation"
TMSDispl.Refresh

    ' link the File control back to the Display control
    ' so that the modified page can be put back into the
    ' original document; the annotation sideband data
    ' will be supplied to the File control along with
    the
    ' image data
TMSFile1.ImageDataSource = TMSDispl.Link

    ' replace the current page of the File control's
    ' document with the new page from the Display
    control
TMSFile1.ReplacePage

    ' save the new (changed) document; the new
    annotation
    ' object will be written to the TIFF header so that
    ' it is available for future display or modification
TMSFile1.OutputFileFormat = 21      ' TIFF Group 3 2-D

    ' add the document pages to the existing output file
TMSFile1.OutputFileAppend = True

    ' save all pages of the document along with any

```

```
' annotations on each page  
TMSFile1.SaveDocument "c:\newfile.tif"
```


Chapter 3 : Reference

Properties, Methods and Events

AboutBox Method

Definition:	Displays a message box showing version and copyright information when queried.
Syntax:	<code>TMSFile1.AboutBox</code>
Data Type:	Long
Return Values:	None
Comments:	The message box is application modal and has a single OK button on it. The message box is unloaded when the button is clicked.

Active Property

Definition:	<p>If set to True at design time, the control will fully initialize and verify licensing immediately upon initialization of the runtime application.</p> <p>If set to False at design time, full initialization of the control will be delayed at initialization of the runtime application. In this case, this property must be explicitly set to True at runtime before the control is used.</p>
Data Type:	Boolean
Design Access:	Read/Write
Runtime Access:	Read/Write (see limits below)
See Also:	"Licensing Configuration and Verification on page 5"
Comments:	<p>If this property is set to True (the default) at design time, the control is fully initialized and licensing is verified immediately upon initialization of the application at runtime. The related technology libraries are loaded and the control is ready to be used.</p> <p>If this property is set to False at design time, the control will only partially initialize when the application loads at runtime. By performing the two actions listed below, the application should be able to load more quickly:</p> <ol style="list-style-type: none">1) The related technology libraries for the control will not be loaded.

- 2) The licensing server will not verify an available token for the control.

If the control initializes with **Active** set to False, this property must be explicitly set to True by the application. Until **Active** is set to True, the control will ignore all instructions to it.

When **Active** is set to True, the licensing server will attempt to verify the availability of a license token, and the related technology libraries will be loaded. These actions will cause a short delay.

If the control fails to find a license token, the **Active** property will be automatically set to False. The application can check this value on Form Load or later to determine if each control is licensed and can be used.

AppendFile Method

Definition: Reads the specified image file and appends its pages to the end of the current document.

Parameters: *filename* Fully qualified path and file name of the image file to

append to the current document

Syntax: TMSFile1.AppendPage *filename*

Data Type: Long

Return Values: None

See Also: AppendPageMethod, LoadFileMethod, InsertFileMethod

Comments: When successfully executed, the **PageCount** property is updated, and the current **PageIndex** remains unchanged. If a multiple-page file is specified, all of the pages in the file will be appended to the document.

The file name specified should generally include a fully qualified path. The drive specification in this property may include either a mapped drive letter or a UNC path.

The following file formats are supported for reading by the TMSSequoia File control:

BMP (read only)

CALS Type1

JPEG (read only)

PCX

TIFF Group 3 with 1-D Option

TIFF Group 3 with 2-D Option

TIFF Group 3 Modified Huffman

TIFF Group 4

TIFF Packbits

TIFF Uncompressed

Note: To open a large format image page, an additional license is required. The determination of whether a given image page is large format is based on the size of the uncompressed image data as calculated by the following formula:

$$(\text{Pixel_Width} * \text{Pixel_Height} * \text{Bits_Per_Pixel}) / 8$$

If the size of the uncompressed image data exceeds 2,103,750 bytes, a large format license is required to open the image. This size limitation calculates to a bitonal double letter page (11"x17") at 300 DPI.

AppendPage Method

Definition: Accepts the current image from the control specified in the **ImageDataSource** property and appends that page to the end of the current document.

Parameters: None

Syntax: TMSFile1.AppendPage

Data Type: Long

Return Values: None

See Also: AppendFileMethod, LoadPageMethod, InsertPageMethod

Comments: When successfully executed, the **PageCount** property is updated to reflect the additional page, but the **PageIndex** property remains unchanged.

If the new image page included additional information such as the annotations, this information will be maintained by the File control. Any annotations associated with the image will be saved into the file header if the output file format is TIFF.

Clear Method

Definition: Removes all references to the current document from memory and resets all informational properties to defaults.

Parameters: None

Syntax: TMSFile1.Clear

Data Type: Long

Return Values: None

See Also: DeletePageMethod, LoadFileMethod, LoadPageMethod

Comments: If any of the pages in the current document have changed since they were originally loaded, the **Dirty** property will be True.

This property may be queried at any time to determine if the current document has been changed and should be saved.

CopyPage Method

Definition: Copies the page at the current**PageIndex**to the specified index position.

Parameters: *index* Page index of the destination for the copied page

Syntax: `TMSFile1.CopyPage index`

Data Type: Long

Return Values: None

See Also: MovePageMethod, DeletePageMethod, PageIndexProperty

Comments: A copy of the current page is placed at the index position specified as a parameter to the method call. The**PageCount** property will be updated, but the**PageIndex**will not change.

DeletePage Method

Definition: Deletes the page at the current**PageIndex**

Parameters: None

Syntax: `TMSFile1.DeletePage`

Data Type: Long

Return Values: None

See Also: Clear Method, MovePageMethod, PageIndexProperty

Comments: Only the page at the current**PageIndex**will be deleted. The **PageCount**property will be updated to reflect the new number of pages in the document.

The **PageIndex**will be changed only if necessary, as when the last page of the document is deleted. In this case, the PageIndex will be automatically updated to point to the new last page in the document.

Dirty Property

Definition: Reports True if the current document has changed in any way since the initial page or file was originally loaded.

Data Type: Boolean

Design Access: Read-only

Runtime Access: Read-only

Comments: Any change to the document will cause this property to report True, indicating that a change has occurred. The change to the document include addition or deletion of pages, annotating a

page, cut and paste to a page, and any other action that changes the image data.

Successful execution of any of the following methods will cause the **Dirty** property to be True:

- AppendFile
- AppendPage
- InsertFile
- InsertPage
- InsertRegion
- ReplacePage
- ReplaceRegion

Error Event

Definition:	Occurs for each error internal to the control.	
Parameters:	Number	A long error code that identifies the error
	Description	Descriptive string of the error
	SCode	A composite long number indicating the severity of the error, the facility code, the origin of the error, and the native error code
	Source	Descriptive string of the source of the error
	HelpFile	Suggested help file name that should have a detailed explanation of the error
	HelpContext	Context ID in the help file
	CancelDisplay	If set to True during this event, the standard error dialog will not be displayed
Comments:	Any time an error occurs inside the TMSSequoia File control, the Error event is triggered. As of this writing, if a runtime error that generated this event is trapped through the <i>On Error</i> statement, the built-in error dialog will not be displayed.	

ImageDataChanged Event

Definition:	Occurs each time the image data supplied to this control is changed.
Parameters:	None
See Also:	ImageDataSourceProperty
Comments:	When the image data supplied by the control specified in the ImageDataSource property changes, this event is fired. Typically, saving is triggered in this event when two or more

ImageBASIC components are linked in a pipe to perform a series of processes.

ImageDataSource Property

- Definition:** Specifies the ImageBASIC control that will supply image data to this control.
- Data Type:** String
- Design Access:** Read/Write
- Runtime Access:** Read/Write
- See Also:** LoadPageMethod, ImageDataChangedEvent
- Comments:** This property may be set to the Link ID of any ImageBASIC control that is capable of outputting image data. Each time the data supplied by that control changes, the TMSSequoia File control is notified in the **ImageDataChanged** event.
- For example, to receive image data from a TMSSequoia Display control and then save that image, the TMSSequoia File control must be linked as shown here:

```
TMSFile1.ImageDataSource = TMSDispl.Link
```

The image data available through this property is used when the following methods are executed:

- AppendPage
- InsertPage
- InsertRegion
- LoadPage
- ReplacePage
- ReplaceRegion

InputFileFormat Property

- Definition:** Reports the format of the most recently loaded image file.
- Data Type:** String
- Design Access:** Not Available
- Runtime Access:** Read-only
- See Also:** InputFileName Property
- Comments:** This property will always report the file type of the most recently opened image file, even if additional image pages have been loaded through the **ImageDataSource** property.
- When an existing image file is opened by the control, this property reports the format of the file. This property will report a value similar to one of the following:

BMP
CALS
PCX
TIFF

The following file formats are supported for reading by the TMSSequoia File control:

BMP (read only)
CALS Type1
JPEG (read only)
PCX
TIFF Group 3 with 1-D Option (Standard Group 3)
TIFF Group 3 with 2-D Option
TIFF Group 3 Modified Huffman
TIFF Group 4
TIFF Packbits
TIFF Uncompressed

InputFileName Property

Definition: When set to a valid image file name, the current document is cleared and the specified file is loaded as a new document.
Note: This property has been maintained for backward compatibility and has been superseded by the **LoadFile** method.

Data Type: String

Design Access: Read/Write

Runtime Access: Read/Write

See Also: LoadFileMethod, PageIndexProperty

Comments: The fully qualified path and file name should be supplied when setting this property. A relative path can be used, but care should be taken when doing so because of possible confusion due to changes in the active directory.

When set to a valid image file name, the following properties are updated:

- InputFileFormat
- PageCount

The **PageIndex** property is set to 1 and the first page of the specified file is loaded into memory. The following properties are populated with information about the currently loaded page:

- PageBitsPerSample
- PageByteWidth

PageHeight
PagePhotoInterp
PageSamplesPerPixel
PageWidth
PageXRes
PageYRes

The following file formats are supported for reading by the TMSSequoia File control:

BMP (read only)
CAL5 Type1
JPEG (read only)
PCX
TIFF Group 3 1-D, Group 3 2-D
TIFF Group 3 Modified
TIFF Group 4
TIFF Packbits
TIFF Uncompressed

InsertFile Method

- Definition:** Inserts the specified file starting at the page at the current **PageIndex**
- Parameters:** *filename* Fully qualified path and file name of the file to insert
- Syntax:** TMSFile1.InsertFile *filename*
- Data Type:** Long
- Return Values:** None
- See Also:** InsertPageMethod, AppendFileMethod, MovePageMethod, PageIndexProperty
- Comments:** If a path is not specified with the file name, or if the path is relative, the file will be opened based on the current directory. The drive specification in this property may include either a mapped drive letter or a UNC path.
- All pages of the specified file will be inserted starting at the current **PageIndex**. The first page of the newly loaded file will be the active page; i.e., it will be the document page specified by the **PageIndex** property. The **PageCount** property will be updated after the new images are added.
- The following file formats are supported for reading by the TMSSequoia File control:
- BMP (read only)
CAL5 Type1

JPEG (read only)
 PCX
 TIFF Group 3 1-D, TIFF Group 3 2-D
 TIFF Group 3 Modified
 TIFF Group 4
 TIFF Packbits
 TIFF Uncompressed

InsertPage Method

Definition: Accepts the page available from the control specified in the **ImageDataSource** property and inserts the new page at the current **PageIndex**

Parameters: None

Syntax: `TMSFile1.InsertPage`

Data Type: Long

Return Values: None

See Also: ImageDataSourceProperty, PageIndexProperty, InsertFile Method, InsertRegionMethod, AppendPageMethod, LoadPage Method

Comments: Upon successful completion of the method, the **PageCount** property will be incremented by one, and the newly inserted page will be the active page; i.e., it will be the page specified by the **PageIndex** property.

InsertRegion Method

Definition: Inserts the Working Region from the **ImageDataSource** at the current **PageIndex**

Parameters: None

Syntax: `TMSFile1.InsertRegion`

Data Type: Boolean

Return Values: True on success
False on error

See Also: ImageDataSourceProperty, InsertPageMethod, InsertFile Method

Comments: Upon successful completion of the method, the **PageCount** property will be incremented by one, and the newly inserted page will be the active page; i.e., it will be the page specified by the **PageIndex** property.

Note: As of this writing, only the ImageBASIC Display controls may be used to specify a Working Region, therefore this method will be applicable only when the **ImageDataSource** property specifies an ImageBASIC Display control. An attempt to insert a region from a control that does not support region specification will result in an error.

Link Property

Definition: Reports the Link ID calculated for this control at its creation.

Data Type: String

Syntax: `TMSDispl.ImageDataSource = TMSFile1.Link`

Design Access: Not Available

Runtime Access: Read-only

Comments: Each ImageBASIC control is assigned a unique Link ID at its creation. This Link ID can be specified in the **ImageDataSource**, **DisplaySource**, **RegionSource**, and **AnnoteSource** properties of various ImageBASIC controls. These source properties specify the ImageBASIC control that is supplying information or services to a control.

Note: The TMSSequoia File control may be specified only for the **ImageDataSource** and **AnnoteSource** property of other ImageBASIC controls.

LoadFile Method

Definition: Clears the current document and loads the specified file to begin a new document.

Parameters: *filename* Fully qualified path and file name of the image file to load

Syntax: `TMSFile1.LoadFile filename`

Data Type: Long

Return Values: None

See Also: AppendFileMethod, InsertFileMethod, LoadPageMethod, Clear Method

Comments: The file name specified should generally include a fully qualified path. The drive specification in this property may include either a mapped drive letter or a UNC path.

Before this method is executed, it is advisable to query the **Dirty** property. This property will report if the current document has been modified from its original form and should be saved.

The following file formats are supported for reading by the TMSSequoia File control:

- BMP (read only)
- CALS Type1
- JPEG (read only)
- PCX
- TIFF Group 3 1-D
- TIFF Group 3 2-D
- TIFF Group 3 Modified
- TIFF Group 4
- TIFF Packbits
- TIFF Uncompressed

Note: To open a large format image page, an additional license is required. The determination of whether a given image page is large format is based on the size of the uncompressed image data as calculated by the following formula:

$$(\text{Pixel_Width} * \text{Pixel_Height} * \text{Bits_Per_Pixel}) / 8$$

If the size of the uncompressed image data exceeds 2,103,750 bytes, a large format license is required to open the image. This size limitation calculates to a bitonal double letter page (11"x17") at 300 DPI.

LoadPage Method

Definition:	Clears the current document and loads the image available from the control specified in the ImageDataSource property to begin a new document.
Parameters:	None
Syntax:	TMSFile1.LoadPage
Data Type:	Boolean
Return Values:	True on success False on error
See Also:	ImageDataSourceProperty, AppendPageMethod, InsertPageMethod, LoadFileMethod
Comments:	When a new page from another ImageBASIC control is loaded through this method, the application should generally save the existing document. For this purpose, the Dirty property will report if the document has been modified since the original page or file was loaded.

MovePage Method

- Definition:** Moves the page at the current **PageIndex** to the specified index.
- Parameters:** *index* Page index of the destination position for the page that is being moved
- Syntax:** `TMSFile1.MovePage index`
- Data Type:** Boolean
- Return Values:** True on success
False on error
- See Also:** PageIndex Property, CopyPageMethod, DeletePageMethod
- Comments:** The current page will be moved to the specified index, changing the index value for subsequent pages. The **PageIndex** property will not change, so the new page that is at that index will be the active page.

OutputBitsPerSample Property

- Definition:** Specifies the bits per pixel for the file that is to be saved.
- Data Type:** Integer
- Design Access:** Read/Write
- Runtime Access:** Read/Write
- See Also:** OutputSamplesPerPixelProperty, OutputPhotoInterpProperty, SaveDocumentMethod
- Comments:** If the color level of the output image is being reduced – e.g., converted from grayscale to bitonal – the threshold level for the color conversion is not user-definable.

The color level of an image is defined by the **OutputBitsPerSample**, **OutputSamplesPerPixel** and **OutputPhotoInterp** properties. Refer to "Color Limitations of File Formats" on page 52 for a discussion of color and file types.

OutputFileAppend Property

- Definition:** If True, the image will be saved as the last page in the output file if it already exists. If the file does not exist, it will be created.
- Data Type:** Boolean
- Design Access:** Read/Write
- Runtime Access:** Read/Write
- See Also:** SaveDocumentMethod, OutputFileFormatProperty

Comments: Of the supported file types, only TIFF files support the creation of multiple pages in a single file. If another file format is selected, this property will be ignored and the specified output file will be overwritten if it exists.

OutputFileFormat Property

Definition: Specifies the file format of the output file.
Data Type: Enumerated
Design Access: Read/Write
Runtime Access: Read/Write
See Also: OutputFileAppendProperty, SaveDocumentMethod
Possible Values: See below
Comments: The following list shows the file formats supported by the TMSSequoia File control and the valid values for this property:

- 2 CALS Type1
- 12 PCX
- 18 TIFF Packbits
- 20 TIFF Group 3 with 1-D Option (Standard Group 3)
- 21 TIFF Group 3 with 2-D Option
- 23 TIFF Group 3 Modified Huffman
- 24 TIFF Group 4
- 28 TIFF Uncompressed

Note: Different file formats are limited in the level and type of color images that they support. Files saved by the TMSSequoia File control will be of the color definition specified in the following properties:

OutputBitsPerSample
OutputPhotoInterp
OutputSamplesPerPixel

Ensure that the file format selected supports the color format of the file. To change the color format of the file. Refer to Color Limitations of File Format\$ on page 52 for a detailed chart.

OutputPhotoInterp Property

Definition: Specifies the photometric interpretation for the saved file.
Data Type: Enumerated
Design Access: Read/Write
Runtime Access: Read/Write

See Also: OutputBitsPerSampleProperty, OutputSamplesPerPixelProperty, SaveDocumentMethod

Comments: The possible values for this property are as follows:

- 0 White0
- 1 White1
- 2 RGB
- 3 Paletted

Refer to 'Color Limitations of File Formats' on page 52 for a more detailed discussion of color and file formats.

If the color level of the output image is being reduced – e.g., converted from grayscale to bitonal – the threshold level for the color conversion is not user-definable.

OutputSamplesPerPixel Property

Definition: Specifies the samples per pixel for the file that is being output.

Data Type: Integer

Design Access: Read/Write

Runtime Access: Read/Write

See Also: OutputBitsPerSampleProperty, OutputPhotoInterpProperty, SaveDocumentMethod

Comments: If the color level of the output image is being reduced – e.g., converted from grayscale to bitonal – the threshold level for the color conversion is not user-definable.

The color type of an image is defined by the **OutputBitsPerSample**, **OutputSamplesPerPixel** and **OutputPhotoInterp** properties. Refer to 'Appendix A : File Formats and Color File' on page 49 for a discussion of color and file types.

OutputScalingDen Property

Definition: Specifies the scaling factor applied to the image data as it is saved.

Data Type: Integer

Design Access: Read/Write

Runtime Access: Read/Write

See Also: OutputScalingNumProperty, SaveDocumentMethod

Comments: Specifies one-half of the scaling ratio that is applied to image data as it is output to file. This property specifies the source half of a ratio defining the number of original pixels relative to

output pixels. Used in conjunction with **OutputScalingNum** to scale output data for saving.

$$\text{OutputScalingNum} / \text{OutputScalingDen} = \text{OutputSize} / \text{InputSize}$$

Scaling the image data will not alter the resolution or color definition of the data. Instead, only the width and height are modified.

For example, to write an image file with one-half the dimensions of the original image, use these parameters:

```
TMSFile1.OutputScalingDen = 1  
TMSFile1.OutputScalingNum = 2
```

OutputScalingNum Property

Definition: Specifies the scaling factor applied to the image data as it is saved.

Data Type: Integer

Design Access: Read/Write

Runtime Access: Read/Write

See Also: **OutputScalingDenProperty**, **SaveDocumentMethod**

Comments: Specifies one-half of the scaling ratio that is applied to image data as it is output to file. This property specifies the destination half of a ratio defining the number of original pixels relative to output pixels. Used in conjunction with **OutputScalingDen** to scale output data for saving.

$$\text{OutputScalingNum} / \text{OutputScalingDen} = \text{OutputSize} / \text{InputSize}$$

Scaling the image data will not alter the resolution or color definition of the data. Instead, only the width and height are modified.

For example, to write an image file with one-half the dimensions of the original image, use these parameters:

```
TMSFile1.OutputScalingDen = 1  
TMSFile1.OutputScalingNum = 2
```

PageBitsPerSample Property

Definition: Reports the bits per sample of the document page that is specified in the **PageIndex** property.

Data Type: Enumerated

Design Access: Not Available

Runtime Access: Read-only

See Also: PageIndexProperty, OutputBitsPerSampleProperty, PagePhotoInterpProperty, PageSamplesPerPixelProperty

Comments: Used in conjunction with the samples per pixel and photometric interpretation, this value defines the level and type of color in the image page. Refer to "Appendix A : File Formats and Color Files" on page 49 for details on supported color types.

PageByteWidth Property

Definition: Reports the number of bytes required in the image file to describe the width of the current document page.

Data Type: Integer

Design Access: Not Available

Runtime Access: Read-only

See Also: PageWidthProperty, PageIndexProperty

Comments: This property is meaningful only when the file type and compression method of the image file requires the storage of image data in byte segments, such as files that use Group 3 and Group 4 compression.

PageCount Property

Definition: Reports the total number of pages in the current document.

Data Type: Integer

Design Access: Not Available

Runtime Access: Read-only

See Also: PageIndexProperty, "Creating and Modifying a Document" on page 9

Comments: Any of the pages in the current document may be made the active page by setting the **PageIndex** property to an integer value between 1 (one) and **PageCount**

PageHeight Property

Definition: Reports the pixel height of the image page specified in the **InputFileName** and **PageIndex** properties.

Data Type: Long

Design Access: Not Available

Runtime Access: Read-only

See Also: PageWidthProperty, PageYResProperty, PageIndexProperty

Comments: The height of an image is always reported in original image pixels without regard for any scaling performed on the image for display or saving (until the new image is loaded).

PageIndex Property

Definition:	Specifies the active page in the current document.
Data Type:	Long
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	PageCountProperty, See additional properties below
Comments:	<p>The PageCount property reports the total number of pages that are in the current document. Any of the available pages may be made the active page by setting the PageIndex property to a value between 1 and PageCount.</p> <p>When the PageIndex is set to a valid value, the specified image page is loaded into memory, and the following descriptive properties are updated:</p> <ul style="list-style-type: none">PageBitsPerSamplePageByteWidthPageHeightPagePhotoInterpPageSamplesPerPixelPageWidthPageXResPageYRes <p>Because they act on the currently active page, the results of the following methods are affected by the PageIndex value:</p> <ul style="list-style-type: none">CopyPageDeletePageInsertFileInsertPageInsertRegionMovePageSavePage

PagePhotoInterp Property

Definition:	Reports the photometric interpretation of the document page that is specified in the PageIndex property.
Data Type:	Enumerated
Design Access:	Not Available
Runtime Access:	Read-only

See Also: PageIndexProperty, OutputPhotoInterpProperty, PageBitsPerSampleProperty, PageSamplesPerPixelProperty

Comments: The possible values for this property are as follows:

- 0 White0
- 1 White1
- 2 RGB
- 3 Paletted

0--White0 indicates that white image pixels are represented in the binary image data as zeros (0). Darker colors are represented by higher values. Applies to bitonal and grayscale images.

1--White1 indicates that black image pixels are represented in the binary image data as zeros (0). Lighter colors are represented by higher values. Applies to bitonal and grayscale images.

2--RGB indicates that each color component (red, green, blue) is defined separately for each pixel. Requires a samples per pixel of three (3).

3--Paletted indicates that a color palette is available to define the pixel colors in this image. Applies to 16-color and 256-color images.

PageSamplesPerPixel Property

Definition: Reports the samples per pixel of the document page that is specified in the **PageIndex** property.

Data Type: Enumerated

Design Access: Not Available

Runtime Access: Read-only

See Also: PageIndexProperty, OutputSamplesPerPixelProperty, PageBitsPerSampleProperty, PagePhotoInterpProperty

Comments: Used in conjunction with the bits per sample and photometric interpretation, this value defines the level and type of color in the image page. Refer to "Appendix A : File Formats and Color Files" on page 49 for details on supported color types.

PageWidth Property

Definition: Reports the pixel width of the current document page.

Data Type: Integer

Design Access: Not Available

Runtime Access: Read-only

See Also: PageHeightProperty, PageXResProperty, PageIndexProperty
Comments: The width of an image is always reported in original image pixels without regard for any scaling performed on the image for display or saving (until the new image is loaded).

PageXRes Property

Definition: Reports the horizontal resolution in DPI of the current document page.
Data Type: Integer
Design Access: Not Available
Runtime Access: Read-only
See Also: PageYResProperty, PageWidthProperty, PageIndexProperty
Comments: The resolution of the image file is typically the same horizontally and vertically. The most common exceptions to this rule are facsimile documents received electronically rather than printed. The resolution reported in this property is in the units specified in the file, typically dots per inch.

PageYRes Property

Definition: Reports the vertical resolution in DPI of the current document page.
Data Type: Integer
Design Access: Not Available
Runtime Access: Read-only
See Also: PageXResProperty, PageHeightProperty, PageIndexProperty
Comments: The resolution of the image file is typically the same horizontally and vertically. The most common exceptions to this rule are facsimile documents received electronically rather than printed. The resolution reported in this property is in the units of dots per inch.

ReplacePage Method

Definition: Replaces the document page at the current **PageIndex** with the entire page from the **ImageDataSource**
Parameters: None
Syntax: TMSFile1.ReplacePage
Data Type: Boolean
Return Values: True on success
False on error
See Also: ReplaceRegionMethod, InsertPageMethod, DeletePageMethod

Comments: The newly inserted page will be the active page (i.e., it will be specified by the current**PageIndex**).

ReplaceRegion Method

Definition: Replaces the document page at the current**PageIndex** with only the Working Region from the**ImageDataSource**

Parameters: None

Syntax: TMSFile1.ReplaceRegion

Data Type: Boolean

Return Values: True on success
False on error

See Also: ReplacePageMethod, InsertRegionMethod, DeletePageMethod

Comments: The newly inserted page will be the active page (i.e., it will be specified by the current**PageIndex**).

Save Method

Definition: Saves the page available from the**ImageDataSource** without modifying the current document.

Note: This method is maintained for backward compatibility but has been superseded by the following methods:

SaveDocument

SavePage

Parameters: *filename* Fully qualified path and file name of the image file to
write

Syntax: TMSFile1.Save *filename*

Data Type: Long

Return Values: 0 on failure
1 on success

See Also: SaveDocument Method

Comments: Several properties set parameters to the saving of any image file. These properties are as follows:

OutputFileFormat	Specifies the type of file to create
OutputFileAppend	If True and the file already exists, the image will be saved as the last page in the file
OutputScalingDen	Specifies the source half of a ratio defining the input pixel dimensions relative to output pixel

	dimensions. Used in conjunction with OutputScalingNum to scale output data. Defaults to one (1).
OutputScalingNum	Specifies the destination half of a ratio defining the input pixel dimensions relative to the output pixel dimensions of an image as it is saved. Used in conjunction with OutputScalingDen to scale output data. Defaults to one (1).
OutputBitsPerSample	Specifies the bits per sample of the output file
OutputSamplesPerPixel	Specifies the samples per pixel of the output file
OutputPhotoInterp	Specifies the photometric interpretation of the output file

SaveDocument Method

Definition:	Saves all pages of the current document to the single, specified file.						
Parameters:	<i>filename</i> Fully qualified path and file name of the image file to write						
Syntax:	TMSFile1.SaveDocument <i>filename</i>						
Data Type:	Boolean						
Return Values:	True on success False on error						
See Also:	SavePageMethod, SaveRegionMethod						
Comments:	Several properties set parameters to the saving of any image file. These properties are as follows: <table> <tr> <td>OutputFileFormat</td><td>Specifies the type of file to create</td></tr> <tr> <td>OutputFileAppend</td><td>If True and the file already exists, the image will be saved as the last page in the file</td></tr> <tr> <td>OutputScalingDen</td><td>Specifies the source half of a ratio defining the input pixel dimensions relative to output pixel dimensions. Used in conjunction with OutputScalingNum to scale output data. Defaults to one (1).</td></tr> </table>	OutputFileFormat	Specifies the type of file to create	OutputFileAppend	If True and the file already exists, the image will be saved as the last page in the file	OutputScalingDen	Specifies the source half of a ratio defining the input pixel dimensions relative to output pixel dimensions. Used in conjunction with OutputScalingNum to scale output data. Defaults to one (1).
OutputFileFormat	Specifies the type of file to create						
OutputFileAppend	If True and the file already exists, the image will be saved as the last page in the file						
OutputScalingDen	Specifies the source half of a ratio defining the input pixel dimensions relative to output pixel dimensions. Used in conjunction with OutputScalingNum to scale output data. Defaults to one (1).						

OutputScalingNum	Specifies the destination half of a ratio defining the input pixel dimensions relative to the output pixel dimensions of an image as it is saved. Used in conjunction with OutputScalingDen to scale output data. Defaults to one (1).
OutputBitsPerSample	Specifies the bits per sample of the output file
OutputSamplesPerPixel	Specifies the samples per pixel of the output file
OutputPhotoInterp	Specifies the photometric interpretation of the output file

SavePage Method

Definition: Saves the Working Region from the **ImageDataSource** without modifying the existing document.

Parameters: *filename* Fully qualified path and file name of the image file to write

Syntax: TMSFile1.SavePage *filename*

Data Type: Boolean

Return Values: True on success
False on error

See Also: SaveDocumentMethod, SaveRegionMethod, PageIndex Property

Comments: The current page is selected by setting the **PageIndex** property. Several properties set parameters to the saving of any image file. These properties are as follows:

OutputFileFormat	Specifies the type of file to create
OutputFileAppend	If True and the file already exists, the image will be saved as the last page in the file
OutputScalingDen	Specifies the source half of a ratio defining the input pixel dimensions relative to output pixel dimensions. Used in conjunction with OutputScalingNum to scale output data. Defaults to one (1).
OutputScalingNum	Specifies the destination half of a ratio defining the input pixel

	dimensions relative to the output pixel dimensions of an image as it is saved. Used in conjunction with OutputScalingDen to scale output data. Defaults to one (1).
OutputBitsPerSample	Specifies the bits per sample of the output file
OutputSamplesPerPixel	Specifies the samples per pixel of the output file
OutputPhotoInterp	Specifies the photometric interpretation of the output file

SaveRegion Method

Definition:	Saves the Working Region of the ImageDataSource to the specified file without affecting the current document.								
Parameters:	<i>filename</i> Fully qualified path and file name of the image file to write								
Syntax:	<code>TMSFile1.SaveRegion <i>filename</i></code>								
Data Type:	Long								
Return Values:	0 on failure 1 on success								
See Also:	SaveDocumentMethod, SavePageMethod, PageIndexProperty								
Comments:	Several properties set parameters to the saving of any image file. These properties are as follows: <table> <tr> <td>OutputFileFormat</td><td>Specifies the type of file to create</td></tr> <tr> <td>OutputFileAppend</td><td>If True and the file already exists, the image will be saved as the last page in the file</td></tr> <tr> <td>OutputScalingDen</td><td>Specifies the source half of a ratio defining the input pixel dimensions relative to output pixel dimensions. Used in conjunction with OutputScalingNum to scale output data. Defaults to one (1).</td></tr> <tr> <td>OutputScalingNum</td><td>Specifies the destination half of a ratio defining the input pixel dimensions relative to the output pixel dimensions of an image as it is saved. Used in conjunction with</td></tr> </table>	OutputFileFormat	Specifies the type of file to create	OutputFileAppend	If True and the file already exists, the image will be saved as the last page in the file	OutputScalingDen	Specifies the source half of a ratio defining the input pixel dimensions relative to output pixel dimensions. Used in conjunction with OutputScalingNum to scale output data. Defaults to one (1).	OutputScalingNum	Specifies the destination half of a ratio defining the input pixel dimensions relative to the output pixel dimensions of an image as it is saved. Used in conjunction with
OutputFileFormat	Specifies the type of file to create								
OutputFileAppend	If True and the file already exists, the image will be saved as the last page in the file								
OutputScalingDen	Specifies the source half of a ratio defining the input pixel dimensions relative to output pixel dimensions. Used in conjunction with OutputScalingNum to scale output data. Defaults to one (1).								
OutputScalingNum	Specifies the destination half of a ratio defining the input pixel dimensions relative to the output pixel dimensions of an image as it is saved. Used in conjunction with								

	OutputScalingDenom to scale output data. Defaults to one (1).
OutputBitsPerSample	Specifies the bits per sample of the output file
OutputSamplesPerPixel	Specifies the samples per pixel of the output file
OutputPhotoInterp	Specifies the photometric interpretation of the output file

Appendix A : File Formats and Color Files

File and Compression Formats

Most of us do not have a lot of experience with image data. We may know that our favorite fax program produces PCX files, and that other programs produce files called TIFF files, but we may not know much beyond that. Many of the features in ImageBASIC will be much more easily understood if we present a short primer on how image data is captured, stored and maintained.

When a black and white image is captured at the scanner, it is held as raster data, or a stream of pixels that map together to form a picture. This data stream is relatively large; e.g., the pixels making up a bitonal 8.5" X 11" image scanned at 300 DPI (dots per inch) resolution will take up about 1 megabyte. The scanner usually sends this data either to the CPU or to a special board for compression.

As the data reaches the CPU there are numerous compression options. Each compression option presents different trade-offs with respect to compactness and decompression speed. For data that is being sent to the screen for display, we use a form of compression called Run Length Encoding (RLE). Run Length Encoding is a form of compressing the data that provides for very rapid decompression for quick display. RLE reduces the file size from 1 megabyte to about 250K, so it is somewhat more costly in terms of memory size than some other options.

By default, Diamond Head Software uses TIFF Group 4 compression in files and RLE in memory. Other options are available, however, and an overview of the various compression methods is given in the next section.

Image File Formats

BMP File Format

A common Windows graphics format, BMP files can be read by almost any Windows-based image viewer. BMP files are paletted, and can represent up to 24-bit color. However, these files are not compressed and can therefore be very large when storing high color or high resolution images.

CALS File Format

CALS is a raster format for compressing bitonal image data. It was developed by the U.S. Department of Defense and is now in wide use throughout federal applications. The image data is either uncompressed or compressed using CCITT Group 4 algorithm. The image data is appended to a raster header block, very much like TIFF files. Although common in U.S. government document imaging applications, CALS is uncommon elsewhere.

GIF File Format

GIF has been popularized by years of use on CompuServe and other Internet resources. Typically grayscale or paletted, GIF images are widely supported and provide reasonable compression -- a typical GIF file will be 20% of the uncompressed data size. This file format can encode an image up to 64K X 64K pixels using from 1 to 8 bits of color.

GIF files use LZW compression and are, therefore, not recommended for use unless you have entered into a licensing agreement with UNISYS Corp., holders of rights to LZW.

JPEG File Format

JPEG is a compression method originally developed to compress photographs. As might be expected, most JPEG files are color or grayscale, not bitonal. JPEG is a "lossy" compression, meaning that some resolution and details are lost when an image undergoes compression. However, the loss of resolution is usually not visible and does not discourage the use of JPEG.

JPEG can encode an image of up to 24-bit color to a maximum size of 64K X 64K pixels. Many image files can be compressed to just 10% of their original size with very little net loss.

PCX / DCX File Format

PCX was developed and distributed by Microsoft and ZSoft and is common throughout all Windows installations. Image data is compressed using an RLE scheme, and is therefore quick but not markedly efficient, particularly when storing high color images.

PCX can encode bitonal, 4-bit, 8-bit, or 24-bit color images up to 64K X 64K pixels. Typical compression for images of 16 colors or fewer is approximately 50%, while high color and complex images can actually be increased in size.

DCX is simply a version of PCX that allows for the storage of multiple images in a single file.

TIFF File Format

TIFF was initially developed by Aldus Corporation to standardize the storage of bitonal images in document imaging and desktop publishing applications. TIFF is a versatile format composed of a header record listing details of the image followed by the compressed image data. Image data in a TIFF can be compressed using a number of different algorithms, and the exact compression scheme is recorded in the header to allow for easier decompression.

TIFF Group 4, the default format for ImageBASIC, compresses typical documents to approximately 5% of the original size. Group 4 is a bitonal standard and will not reliably store any grayscale or color images.

Compression Types

CCITT Group 3 Compression

Group 3 compression is a standard developed by CCITT, a standards organization committee responsible for the sanctioning of many file compression methods. This technique uses a combination of RLE, differential, and Huffman encoding.

CCITT Group 4 Compression

CCITT Group 4 compression is similar to Group 3 except that Group 3 limits its compression to one dimension, while Group 4 compresses both horizontally and vertically. Group 4 compression results in the smallest file size of all the options available in ImageBASIC, which is why it is the default compression method for saving files.

LZW Compression

LZW is a dictionary-based compression algorithm first developed in the late 1970's. LZW compression substitutes portions of image data that have been seen before with shorter strings stored in its dictionary.

Run Length Encoding (RLE)

RLE (Run Length Encoding) is a simple compression algorithm that encodes a run of identical pixels as the pixel color plus the number of pixels in the run. This compression will generally result in a file size approximately one-quarter the original size. RLE compression is very fast but relatively inefficient; however, because of its speed advantages, it is used as the compression method for images held in memory by ImageBASIC and also in BMP files.

Uncompressed Image Files

Uncompressed images are simply that -- no compression has been applied to the raw image data. Therefore, the image files tend to be very large. For example, the raw data to describe a bitonal letter size image at 300 DPI is about 1 MB. Adding to the area of the image or to the level of color can drastically increase the file size and the time required to transfer and display it.

Color Limitations of File Formats

The level of color that can be saved in each file format is a function of the specification of that format. For example, TIFF Group 4 is a bitonal standard, but JPEG can store up to 24-bit color. The next section introduces the definition of color in image files and contains tables that shown the level of color that can be saved in each file type.

Comparative Charts of Colors Supported by File Format

In the following tables, the entries in the top row of each table indicate the color format of an image file. The three numbers indicate Bits Per Sample, Samples Per Pixel, and Photometric Interpretation, respectively.

The leftmost column indicates the file format. By finding the intersection of the file type and color format, you can find the most appropriate, fully functional combination for your needs.

BitsPerSample(BPS) indicates how many bits define the different values for each sample (see **SamplesPerPixel**) that defines a pixel's color. The only valid values are 1, 4, and 8.

SamplesPerPixel(SPP) indicates how many individual values are used to define a pixel's color. The only valid values are 1 and 3. A value of 1 indicates that this image is not RGB, leaving bitonal, gray scale, and paletted as possibilities. BitsPerSample and PhotometricInterpretation will indicate which of these possible formats is actually used.

PhotometricInterpretation(PI) indicates the interpretation of color values for display. The valid values of PhotometricInterpretation are as follows:

- 0 White 0 Typical bitonal/gray interpretation in which 0 indicates white and higher numbers are darker shades.
- 1 White 1 The inverse of White 0, in which 0 is interpreted as black and higher numbers are lighter shades.
- 2 RGB Red, Green, and Blue values of each pixel are specified; this setting requires a value of 3 for SamplesPerPixel
- 3 Paletted An array is constructed with a color assigned to each value in than array. This value is generally only applied to color images and limits the image to fewer colors than RGB, typically 16-color or 256-color.

The key to the results codes in the tables is as follows:

- Y The combination file type and color definition is fully functional
- ~ Recognizable image but palette is not converted properly
- N This combination does not work at all

Bitonal and Grayscale Image File Formats

	Binary BPS 1 SPP 1 PI 0/1	16-level gray BPS 4 SPP 1 PI 0/1	256-level gray BPS 8 SPP 1 PI 0/1
TIFF Group 4	Y	N	N
TIFF Group 3	Y	N	N
TIFF Packed	Y	Y	Y
TIFF Uncompressed	Y	Y	Y
CALS	Y	N	N
BMP	Y	N	Y
JPEG	N	N	N

Paletted and RGB Image File Formats

	16-color palette BPS 4 SPP 1 PI 3	256-color palette BPS 8 SPP 1 PI 3	24-bit RGB BPS 8 SPP 3 PI 2
TIFF Group 4	N	N	N
TIFF Group 3	N	N	N
TIFF Packed	Y	Y	Y
TIFF Uncompressed	Y	Y	Y
CALS	N	N	N
BMP	Y	Y	N
JPEG	N	N	Y

Index

A

- AboutBox Method 21
- Active Property 21
- Adding to a Document
 - AppendFile Method 10, 22
 - AppendPage Method 12, 23
 - InsertFile Method 11, 28
 - InsertPage Method 12, 29
 - InsertRegion Method 29
 - ReplacePage Method 13, 39
 - ReplaceRegion Method 13, 39
- Annotation Control 4
- AppendFile Method 10, 22
- AppendPage Method 12, 23
- Arranging Pages of a Document
 - AppendFile Method 10, 22
 - AppendPage Method 12, 23
 - CopyPage Method 12, 24
 - DeletePage Method 12, 24
 - InsertFile Method 11, 28
 - InsertPage Method 12, 29
 - InsertRegion Method 29
 - MovePage Method 13, 31
 - ReplacePage Method 13, 39
 - ReplaceRegion Method 13, 39

B

- BMP File Format 3, 46

C

- CALS File Format 3, 46
- Changes to Document
 - Dirty Property 24
- Clear Method 10, 12, 23
- Color Definition 48
 - Bits Per Sample 49
 - OutputBitsPerSample Property 17, 32
 - OutputPhotoInterp Property 17, 33
 - OutputSamplesPerPixel Property 17, 34
 - Photometric Interpretation 49
 - Samples Per Pixel 49
- Compression Options 45

- Control Communication 3
- CopyPage Method 12, 24

D

- Definitions of Commonly Used Terms 2
- DeletePage Method 12, 24
- Dirty Property 24

E

- Error Event 25
- Events
 - Error 25
 - ImageDataChanged 25

F

- File Format
 - OutputFileFormat Property 16, 32
- File Formats Supported 2
- File Information
 - InputFileFormat 13, 26
 - PageBitsPerSample Property 35
 - PageCount 13
 - PageCount Property 36
 - PageHeight Property 36
 - PagePhotoInterp Property 37
 - PageSamplesPerPixel Property 38
 - PageWidth Property 38
 - PageXRes Property 38
 - PageYRes Property 39
 - Property 35
- Formats Supported 2

G

- GIF File Format 46
- Group 3 Compression 47
- Group 4 Compression 47

I

- Image Data Output 14
- Image Page Too Large 11, 23, 31
- ImageDataChanged Event 25
- ImageDataSource 3, 4
- ImageDataSource Property 9, 26
- Input Options 1

InputFileFormat Property 13, 26
InputFileName Property 27
InsertFile Method 11, 28
InsertPage Method 12, 29
InsertRegion Method 29
Introduction to the Document Control 1

J

JPEG File Format 46

L

Large Format Images 11, 23, 31
Licensing
 Active Property 21
Link ID 4
Link property 4, 30
Linking Controls 3, 4
 ImageDataChanged Event 25
 ImageDataSource Property 26
Load Time Improvement
 Active Property 21
LoadFile Method 10, 30
Loading a Multiple Page File
 PageIndex Property 36
Loading an Image
 InputFileName 27
Loading an Image from File 10
LoadPage Method 12, 31
LZW Compression 48

M

Methods

AboutBox 21
AppendFile 10, 22
AppendPage 12, 23
Clear 10, 12, 23
CopyPage 12, 24
DeletePage 12, 24
InsertFile 11, 28
InsertPage 12, 29
InsertRegion 29
LoadFile 10, 30
LoadPage 12, 31
MovePage 13, 31
ReplacePage 13, 39
ReplaceRegion 13, 39
Save 17, 18, 40
SaveDocument 1, 41

SavePage 1, 42
SaveRegion 43
Modifying a Document
 AppendFile Method 10, 22
 AppendPage Method 12, 23
 CopyPage Method 12, 24
 DeletePage Method 12, 24
 InsertFile Method 11, 28
 InsertPage Method 12, 29
 MovePage Method 13, 31
 ReplacePage Method 13, 39
 ReplaceRegion Method 13, 39
MovePage Method 13, 31
Multi-Page Files
 OutputFileAppend Property 17, 32

N

New Document LoadFile 10
New Document LoadPage 12, 31

O

Ouput to File 16
Ouput of Image Data
 Link Property 30
Ouput Options 1
Output to Other ImageBASIC Controls
 14
OutputBitsPerSample Property 17, 32
OutputFileAppend Property 17, 32
OutputFileFormat Property 16, 32
OutputPhotoInterp Property 17, 33
OutputSamplesPerPixel Property 17, 34
OutputScalingDen Property 17, 34, 40,
 41, 42, 43
OutputScalingNum Property 17, 35, 40,
 41, 42, 43

P

PageBitsPerSample Property 35
PageByteWidth Property 35
PageCount Property 13, 36
PageHeight Property 36
PageIndex Property 36
PagePhotoInterp 37
PageSamplesPerPixel Property 38
PageWidth Property 38
PageXRes Property 38
PageYRes Property 39

PCX / DCX File Format 47

PCX File Format 3

Properties

Active 21

Dirty 24

ImageDataSource 9, 26

InputFileFormat 13, 26

InputFileName 27

Link 30

OutputBitsPerSample 17, 32

OutputFileAppend 17, 32

OutputFileFormat 16, 32

OutputPhotoInterp 17, 33

OutputSamplesPerPixel 17, 34

OutputScalingDen 17, 34, 40, 41,
42, 43

OutputScalingNum 17, 35, 40, 41,
42, 43

PageBitsPerSample 35

PageByteWidth 35

PageCount 13, 36

PageHeight 36

PageIndex 36

PagePhotoInterp 37

PageSamplesPerPixel 38

PageWidth 38

PageXRes 38

PageYRes 39

Purpose of the Document Control 1

R

Reading a File 10

Removing a Document

Clear Method 10, 12, 23

ReplacePage Method 13, 39

ReplaceRegion Method 13, 39

Routing Data between Controls 3

Run Length Encoding (RLE) 48

Runtime Linking 4

S

Save Method 17, 18, 40

SaveDocument Method 1, 41

SavePage Method 42

SaveRegion Method 43

Saving Color Files 48

Saving Files

ImageDataSource Property 26

OutputBitsPerSample 17, 32

OutputFileAppend Property 17, 32

OutputFileFormat Property 16, 32

OutputPhotoInterp 17, 33

OutputSamplesPerPixel 17, 34

OutputScalingDen 17, 34, 40, 41,
42, 43

OutputScalingNum 17, 35, 40, 41,
42, 43

Save Method 17, 18, 40

SaveDocument Method 41

SaveDocumentMethod 1

SavePage Method 1, 42

SaveRegion Method 43

Saving Images to File 16

Scaling Output Files

OutputScalingDen Property 17, 34,
40, 41, 42, 43

OutputScalingNum Property 17, 35,
40, 41, 42, 43

Sending Images to Other Controls 14

Size Limitation, Image Page 11, 23, 31

Source of Image Data 4

ImageDataSource Property 9

Starting a Document

LoadFile Method 10, 30

LoadPage Method 12, 31

Supported Color Formats 48

Supported File Formats 2

T

TIFF File Format 3, 47

Timing of Licensing Verification

Active Property 21

U

Uncompressed Image Files 48

V

Version Information

AboutBox Method 21