32 bit ActiveX Image Acquisition Control

- **Fully TWAIN compliant**
- **Easy to use Automatic Mode**
- **Full control over TWAIN capabilities**
- **Royalty FREE runtime distribution**

CompleteControl™

What is iTWAIN

More About iTWAIN

iTWAIN Reference

Properties
Methods
Events

System Requirements

Sample Applications

Examples

Contact Information

How to contact us
Obtaining Technical Support
Obtaining more information about TWAIN

Registration

Registering your purchase

Licensing and Distribution

License Agreement
Distributing iTWAIN
UnInstalling iTWAIN

iTWAIN is a 32 bit ActiveX Control designed for easy acquisition of high quality images from digital input devices such as scanners and cameras. As the name suggests **iTWAIN** is fully TWAIN compliant.

With **iTWAIN**, you can build image capture functionality into your applications in a matter of minutes using the Automatic Mode.   However, for complete control, Manual Mode gives you access to ALL of the TWAIN capabilities.

iTWAIN is completely royalty free. You may distribute unlimited copies of your runtime application without any further license fee or royalty fee payments.

Everything you need to start using this control immediately is here including the TWAIN Source Manager, TWAIN sample sources, and Visual Basic sample applications.

iTWAIN supports properties, methods and events to enable you to …

Initialize, control and interrogate an input device

Acquire images from the input device as a device independent bitmap (DIB) or any graphics files supported by your imaging device

Display the acquired images

Transfer the acquired image to other picture controls and applications via the clipboard or the saved file

Get information about images

Build sophisticated image capture applications with control over any features such as document feeding, image layout, multiple frames per page, and alternative user interafaces, which are supported by your imaging device.

Events are also supported that allow you to take appropriate actions before, during and after image transfer.

iTWAIN is just one of many ActiveX components in our CompleteControl range.

iTWAIN is supported under the following 32 bit Microsoft® Windows operating systems on Intel® architecture machines.

Windows 95
Windows NT Server and Workstation 4.0
and future versions of these operating systems.

Note:

The current version of iTWAIN is not supported on beta versions of the above operating systems

The current version of iTWAIN is not supported on win32s.

The current version of iTWAIN is not supported on non-Intel® architecture implementations of Microsoft® Windows.

For information about future versions and updates please register with Imagine IT Ltd

To register your copy of iTWAIN please print out this page and send it to us after completing the details. Alternatively you can also E-Mail the required information to us at `registration@imagineit.co.uk`

Registered users automatically receive information about updates, new releases and future products from Imagine IT and are eligible for technical support.

Product: iTWAIN OCX 2.0

Last Name:	**First Name:**
--	--
Company Name:	
--	
Address:	
--	
--	
--	
--	
Telephone Number:	**Fax Number:**
--	--
E-Mail Address:	**Web URL:**
--	--
Date of Purchase:	**Purchased From:**
--	--

----

please post to: Imagine IT Limited, 3rd Floor, Hygeia Building, 66 College Road, Harrow HA1 1BE, United Kingdom
or E-Mail to `registration@imagineit.co.uk`

iTWAIN is designed and developed by Imagine IT Limited, United Kingdom.

Imagine IT specialises in object technologies and component software development.

Apart from creating great components we also help customers develop their own line-of-business objects and applications.

To contact Imagine IT -

Write to us at:

```
Imagine IT Limited
3rd Floor, Hygeia Building
66 College Road
Harrow, HA1 1BE
United Kingdom
```

Telephone:

from within the United Kingdom:	`0181 324 1240`
from other countries:	`international access code + 44 + 181 324 1240`

Fax:

from within the United Kingdom:	`0181 324 1752`
from other countries:	`international access code + 44 + 181 324 1752`

E-Mail:

for general inquiries:	`info@imagineit.co.uk`
for technical support inquiries:	`support@imagineit.co.uk`
for registration:	`registration@imagineit.co.uk`

World Wide Web:

find us at:	`http://www.imagineit.co.uk`

All our products are available for free evaluation form our web site. Please do visit regularly and download the latest components.

For the fastest possible response please direct all technical support enquiries by E-Mail to `support@imagineit.co.uk` whenever possible. Alternatively fax your enquiry to the above fax number marked for the attention of “Technical Support”.

If you do not have access to E-Mail or fax then please call us at the above telephone number and ask for “Technical Support”

NOTE: We can only provide technical support to registered users.

Getting information on TWAIN:

HPFirst is a fax reply system at Hewlett-Packard. This system contains two TWAIN documents. To receive these documents call from a touch tone phone or fax machine and information will be faxed to you.

Inside the US or Canada	800 333-1917
Outside the US or Canada	208 344-4809

Using HPFirst you can receive:

3130	The TWAIN toolkit order form
3019	The TWAIN White Paper

The white paper and FAQ guide are also available through CompuServe and an ftp site.

In CompuServe: GO HPPERIPH and look under TWAIN.

Anonymous ftp: `ftp.twain.org(/pub/users/twain)`

Find the TWAIN Working Group on the world wide web:
`http://www.twain.org/`

Information about the TWAIN Developer's Mailing List:

TWAIN developers are invited to join in discussion about TWAIN and problem exchanges on the TWAIN developer's mailing list.

To be added or removed to the mailing list, please write to:
`twain-request@caere.com`

To post questions or other information of interest to TWAIN developers write to:
`twain@caere.com`

How to Order the TWAIN Toolkit:

US/Canada
Call (800) 722-0379 and order the TWAIN Toolkit.

International
Call (303) 739-4067 and order the TWAIN Toolkit.

All
Fax:(970) 330-7655

NOTE:
We have provided the above information to you for your guidance only. The information is subject to change and you should first check with the above mentioned companies and organisations before downloading files, taking part in on-line discussions and mailing lists and before placing any purchase orders.

IMPORTANT

THE USE OF THIS SOFTWARE IS SUBJECT TO THE TERMS OF THE LICENSE AGREEMENT PRINTED BELOW. PLEASE READ THE LICENSE AGREEMENT CAREFULLY. IF YOU DO NOT AGREE WITH ALL OF THE TERMS, THEN YOU SHOULD NOT USE THE SOFTWARE IN ANY WAY WHATSOEVER INCLUDING COPYING OR DISTRIBUTING THE SOFTWARE OR ANY PART OF THE SOFTWARE TO ANY THIRD PARTY. IF YOU USE THE SOFTWARE IN ANY WAY WHATSOEVER IT WILL BE DEEMED TO INDICATE YOUR ACCEPTANCE OF ALL THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT.

THE AGREEMENT

THIS DOCUMENT IS A LEGAL AGREEMENT BETWEEN IMAGINE IT LIMITED, UNITED KINGDOM, HEREAFTER REFERRED TO AS "IMAGINE IT" AND THE USER OF THE SOFTWARE HEREIN REFERRED TO AS "LICENSEE"

DEFINITIONS

"SOFTWARE" MEANS THE SOFTWARE KNOWN AS "iTWAIN OCX" AND ALL ITS COMPONENTS, PARTS AND DOCUMENTATION, WHICH HAVE BEEN DEVELOPED BY IMAGINE IT

"END USER APPLICATION" MEANS ANY APPLICATION DEVELOPED WITH THE HELP OF OR BY USING THE SOFTWARE

PURPOSE

THE PURPOSE OF THIS AGREEMENT IS TO DEFINE THE RELATIONSHIP BETWEEN IMAGINE IT AND LICENSEE, THE TERMS, CONDITIONS, AND LIMITATIONS OF USE, THE RIGHTS OF BOTH PARTIES AND THE LIMITATION OF LIABILITY ARISING FROM THE USE OF THE SOFTWARE

EFFECTIVE DATE

THIS AGREEMENT IS DEEMED TO HAVE BEEN MADE EFFECTIVE ON THE FIRST DATE AT WHICH LICENSEE OPENS THE PACKAGE CONTAINING THE SOFTWARE AND ITS ASSOCIATED DOCUMENTATION, OR IF THE SOFTWARE IS DISTRIBUTED ELECTRONICALLY, THE FIRST DATE ON WHICH THE LICENSEE OBTAINS THE SOFTWARE

GRANT OF LIMITED LICENSE

I. EVALUATION LICENSE

IF THE LICENSEE HAS NOT PURCHASED A FULL PRODUCT KEY FOR THE SOFTWARE THEN IMAGINE IT GRANTS THE LICENSEE A NON EXCLUSIVE, NON TRANSFERABLE, PERSONAL LICENSE TO USE THE SOFTWARE AND ITS DOCUMENTATION FOR EVALUATION PURPOSES ONLY AND FOR NO OTHER PURPOSE. THE LICENSE DOES NOT PERMIT THE DEVELOPMENT AND DISTRIBUTION OF ANY END USER APPLICATIONS USING THE SOFTWARE

II. FULL DEVELOPMENT LICENSE

IF THE LICENSEE HAS PURCHASED A FULL PRODUCT KEY FOR THE SOFTWARE THEN IMAGINE IT GRANTS THE LICENSEE A NON EXCLUSIVE, NON TRANSFERABLE, PERSONAL LICENSE TO USE THE SOFTWARE AND ITS DOCUMENTATION FOR THE PURPOSE OF DEVELOPING AND DISTRIBUTING ANY NUMBER OF END USER APPLICATIONS AND TO COPY AND DISTRIBUTE ANY PARTS OF THE SOFTWARE DEFINED BELOW UNDER REDISTRIBUTABLE COMPONENTS TOGETHER WITH AND AS PART OF THE END USER APPLICATION, SUBJECT TO ALL THE TERMS AND CONDITIONS AND RESTRICTIONS IN THIS AGREEMENT.

RESTRICTIONS

A USER OF ANY END USER APPLICATIONS MAY NOT FURTHER USE PARTS OF THE SOFTWARE FOR SOFTWARE DEVELOPMENT, COPYING OR DISTRIBUTION. THE LICENSEE MUST ENFORCE THIS IN A SEPARATE AGREEMENT WITH THE USER OF ANY END USER APPLICATIONS.

THE LICENSEE MAY USE ONLY ONE COPY OF THE SOFTWARE AT ANY TIME ON ONE IBM PC OR COMPATIBLE SYSTEM.

ADDITIONAL PRODUCT KEYS FOR THE SOFTWARE MUST BE PURCHASED IF IT IS REQUIRED TO BE USED IN A MULTI USER NETWORKED ENVIRONMENT IN QUANTITIES OF ONE KEY FOR EACH PERSON HAVING ACCESS TO AND USING THE SOFTWARE.

THE LICENSEE MAY MAKE A SINGLE BACKUP COPY OF THE SOFTWARE.

THE LICENSEE MAY NOT USE, COPY, MODIFY, REVERSE ENGINEER, DISASSEMBLE, SELL, TRANSFER, HIRE, LEND OR OTHERWISE DISTRIBUTE THE SOFTWARE OR ANY OF ITS DOCUMENTATION OR COMPONENTS IN WHOLE OR IN PART, EXCEPT AS EXPRESSLY PROVIDED FOR IN THIS AGREEMENT.

THE END USER APPLICATION MUST NOT BE ANOTHER DEVELOPMENT TOOL DESIGNED TO BE USED FOR CREATING OTHER END USER APPLICATIONS USING THE SOFTWARE. THE LICENSEE MUST ENFORCE THIS IN A SEPARATE AGREEMENT WITH THE USER OF THE END USER APPLICATION.

THE LICENSEE HAS NO RIGHTS TO THE ORIGINAL SOURCE CODE OF THE SOFTWARE WHATSOEVER.

THE LICENSEE MUST INFORM IMAGINE IT OF THE NATURE OF EACH AND EVERY NEW END USER APPLICATION IT

DEVELOPS WITH THE USE OF THE SOFTWARE.

THE LICENSEE SHALL COMPLY WITH ALL LOCAL AND INTERNATIONAL LAWS AND EXPORT / IMPORT REGULATIONS WHEN DISTRIBUTING PARTS OF THIS SOFTWARE AS PROVIDED FOR BELOW ALONG WITH ANY END USER APPLICATIONS

REDISTRIBUTABLE COMPONENTS

THE SOFTWARE, BY ITS NATURE, INCLUDES COMPONENTS WHICH ARE REQUIRED TO BE DISTRIBUTED WITH ANY END USER APPLICATIONS IN ORDER FOR THAT END USER APPLICATION TO FUNCTION.

I. EVALUATION LICENSE

IF THE LICENSEE HAS NOT PURCHASED A FULL PRODUCT KEY FOR THE SOFTWARE THEN THERE ARE NO DISTRIBUTABLE COMPONENTS. THE LICENSEE SHALL NOT DISTRIBUTE ANY COMPONENTS OF THE SOFTWARE

II. FULL DEVELOPMENT LICENSE

IF THE LICENSEE HAS PURCHASED A FULL PRODUCT KEY FOR THE SOFTWARE THENTHE FOLLOWING FILES DEFINE THE COMPONENTS WHICH THE LICENSEE MAY DISTRIBUTE PROVIDED THAT THEY ARE DISTRIBUTED AS PART OF THE END USER APPLICATION AND PROVIDED THAT THEY ARE DISTRIBUTED WITHOUT ANY MODIFICATION AND ARE COPIED FROM THE ORIGINAL DISKS OR CDS. ALL FILES LISTED BELOW MUST BE DISTRIBUTED WITH THE END USER APPLICATION.

1. **iTWAIN.OCX**
2. **iTWAIN.RTL**

NO OTHER FILES SUPPLIED WITH THE SOFTWARE MAY BE DISTRIBUTED WITH THE END USER APPLICATION.

THE SOFTWARE BY ITS NATURE ALSO REQUIRES SOME FILES WHICH ARE DISTRIBUTED BY MICROSOFT(R) TO BE PRESENT ON THE END USER'S COMPUTER FOR THE END USER APPLICATION TO FUNCTION. THIS LICENSE AGREEMENT DOES NOT IN ANY WAY PROVIDE FOR LICENSEE TO DISTRIBUTE THESE COMPONENTS WITH THE END USER AGREEMENT. THE LICENSEE MUST VERIFY THEIR RIGHTS TO DISTRIBUTE THESE COMPONENTS, SEPARATELY WITH MICROSOFT(R) BEFORE DISTRIBUTING THESE COMPONENTS WITH THE END USER APPLICATION

1. **MFC42.DLL**
2. **OLEPRO32.DLL**
3. **REGSVR32.EXE**

THE SOFTWARE BY ITS NATURE ALSO REQUIRES SOME FILES WHICH ARE DISTRIBUTED BY THE TWAIN WORKING GROUP TO BE PRESENT ON THE END USER'S COMPUTER FOR THE END USER APPLICATION TO FUNCTION.

THIS LICENSE AGREEMENT DOES NOT IN ANY WAY PROVIDE FOR LICENSEE TO DISTRIBUTE THESE COMPONENTS WITH THE END USER AGREEMENT. THE LICENSEE MUST VERIFY THEIR RIGHTS TO DISTRIBUTE THESE COMPONENTS, SEPARATELY WITH THE TWAIN WORKING GROUP BEFORE DISTRIBUTING THESE COMPONENTS WITH THE END USER APPLICATION

1. **TWAIN.DLL**
2. **TWAIN_32.DLL**
3. **TWUNK_16.EXE**
4. **TWUNK_32.DLL**
5. **TWSRC_16.DS**
6. **TWSRC_32.DS**

TECHNICAL SUPPORT

ONLY THOSE LICENSEES WHO HAVE PURCHASED A FULL PRODUCT KEY AND REGISTERED WITH IMAGINE IT BY FULLY COMPLETING AND RETURNING ALL THE INFORMATION ASKED FOR IN THE REGISTRATION FORM OR RELEVANT SECTION IN THE ON-LINE HELP DOCUMENTATION ARE QUALIFIED FOR RECEIVING TECHNICAL SUPPORT HELP ON THE SOFTWARE.

TECHNICAL SUPPORT IS RESTRICTED TO ANSWERING QUESTIONS ABOUT THE USE OF THE SOFTWARE OR ERRORS IN THE SOFTWARE IF ANY.

TECHNICAL SUPPORT DOES NOT INCLUDE DEBUGGING AND OTHER PROBLEM SOLVING TASKS ASSOCIATED WITH THE DEVELOPMENT OF ANY END USER APPLICATIONS.

USERS OF THE END USER APPLICATION ARE NOT ELIGIBLE FOR OBTAINING TECHNICAL SUPPORT FROM IMAGINE IT UNDER ANY CIRCUMSTANCES.

THIS SERVICE WILL BE PROVIDED ACCORDING TO IMAGINE IT'S TECHNICAL SUPPORT POLICIES FROM TIME TO TIME, AND IS LIMITED TO SUCH TIME AS IMAGINE IT SUPPORTS THE SOFTWARE OR OWNS THE SOFTWARE.

TECHNICAL SUPPORT MAY BE PROVIDED BY MEANS OF E-MAIL, FAX, BULLETIN BOARDS, ON-LINE SERVICES, WRITTEN CORRESPONDENCE OR TELEPHONE AT IMAGINE IT'S SOLE DISCRETION.

OWNERSHIP

IMAGINE IT SHALL REMAIN THE OWNERS OF ALL RIGHTS, TITLE AND INTEREST IN THE SOFTWARE

THIS LICENSE DOES NOT CONFER ANY OWNERSHIP RIGHTS TO THE LICENSEE

THE LICENSEE AGREES NOT TO USE IMAGINE IT'S NAME IN ANY WAY OR FORM

THE LICENSEE AGREES NOT TO IMPLY THAT ANY END USER APPLICATIONS IT DEVELOPS ARE EITHER APPROVED OR OTHERWISE SANCTIONED BY IMAGINE IT

THE LICENSEE AGREES TO CARRY A COPYRIGHT NOTICE IN THE ABOUT BOX OR HELP FILE OF EACH AND EVERY COPY OF ANY END USER APPLICATIONS IT DEVELOPS WITH THE HELP OF THIS SOFTWARE, WITH THE FOLLOWING WORDS :-

"PORTIONS OF THIS APPLICATION PROVIDED BY IMAGINE IT LIMITED, UNITED KINGDOM, COPYRIGHT (C) 1996-1997 IMAGINE IT LIMITED. ALL RIGHTS RESERVED"

TERMINATION

THIS AGREEMENT SHALL BE TERMINATED IN WRITING BY IMAGINE IT AT ANY TIME IF THE LICENSEE BREACHES ANY TERMS AND CONDITIONS CONTAINED IN THIS AGREEMENT OR CARRIES OUT ANY ACTIONS EXPRESSLY PROHIBITED IN THIS AGREEMENT, AND CONTINUES TO BE IN BREACH FOR 30 DAYS AFTER WRITTEN NOTICE IS GIVEN TO THE LICENSEE BY IMAGINE IT OR ITS REPRESENTATIVES.

UPON TERMINATION LICENSEE SHALL RETURN ALL COPIES OF THE SOFTWARE, ITS COMPONENTS AND DOCUMENTATION TO IMAGINE IT.

ALL RIGHTS GRANTED TO THE LICENSEE IN THIS AGREEMENT SHALL CEASE UPON TERMINATION OF THIS AGREEMENT EXCEPT FOR THOSE APPLYING TO ANY COPIES OF ANY END USER APPLICATIONS ALREADY PROPERLY DISTRIBUTED AND LICENSED WITHIN THE TERMS OF THIS AGREEMENT, PRIOR TO TERMINATION OF THIS AGREEMENT.

CONFIDENTIALITY

THE LICENSEE ACKNOWLEDGES THAT THE SOFTWARE AND ITS DOCUMENTATION AND DESIGN CONSTITUTE CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO IMAGINE IT.

THE LICENSEE AGREES NOT TO DISCLOSE ANY OF THIS INFORMATION TO OTHER THIRD PARTIES EXCEPT FOR ITS EMPLOYEES WHO ARE ENGAGED IN USING THE SOFTWARE AND ARE BOUND BY THESE SAME TERMS THROUGH A SEPARATE WRITTEN AGREEMENT WITH LICENSEE, OR EXCEPT WHERE REQUIRED TO DISCLOSE THE INFORMATION BY LAW.

WARRANTIES AND DISCLAIMERS

THE SOFTWARE IS PROVIDED "AS IS". IMAGINE IT DOES NOT WARRANT THAT THE SOFTWARE WILL OPERATE WITHOUT ERRORS, OR THAT IT WILL MEET ALL OF THE LICENSEE'S REQUIREMENTS.

ALL WARRANTIES, EXPRESSED OR IMPLIED, ARE EXCLUDED FROM THIS AGREEMENT INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE BUT NOT EXCLUDING ANY STATUTARY RIGHTS PROVIDED FOR BY LAW.

IMAGINE IT SHALL NOT BE LIABLE FOR ANY DAMAGES INCLUDING BUT NOT LIMITED TO INCIDENTAL OR CONSEQUENTIAL LOSS, LOSS OF PROFIT, LOSS OF SALES, INJURY, DEATH, LOSS OF OR DAMAGE TO PROPERTY OR ANY OTHER LOSS ARISING DIRECTLY OR INDIRECTLY FROM THE USE OF THE SOFTWARE.

IMAGINE IT'S SOLE REMEDY TO THE LICENSEE, AT IMAGINE IT'S SOLE DISCRETION SHALL BE LIMITED TO EITHER THE REPAIR OR REPLACEMENT OF FAULTY SOFTWARE OR THE REPAYMENT OF THE LICENSE FEE PAID BY THE LICENSEE UPON RETURN OF ALL COPIES OF THE SOFTWARE. IMAGINE IT'S TOTAL LIABILITY SHALL NOT EXCEED THE AMOUNTS PAID BY LICENSEE FOR THE PURCHASE OF THIS LICENSE.

COPYRIGHT (C) 1996-1997   IMAGINE IT LIMITED, UNITED KINGDOM, ALL RIGHTS RESERVED.

MICROSOFT IS A REGISTERED TRADEMARK OF MICROSOFT CORPORATION.

IMAGINE IT, THE IMAGINE IT LOGO AND iTWAIN,   ARE TRADEMARKS OF IMAGINE IT LIMITED

APPLICABLE LAW

THIS AGREEMENT SHALL BE GOVERNED BY TO THE LAWS OF ENGLAND.

END

In order to use **iTWAIN** in your applications you need to distribute the OCX with your application. In addition you also need to ship some Microsoft® shared libraries which the OCX uses at run-time. If you also want to supply the TWAIN source manager with your application then you will need to ship some files which are distributed by the TWAIN Working Group

Before distributing any files which have been shipped with this product please read the license agreement and ensure that you comply with all its terms and conditions. Note that you do not have a license to distribute any Microsoft® files or TWAIN Working Group files as part of this license and you should ensure that you have the appropriate licenses from Microsoft® and the TWAIN Working Group. You should not distribute these third party components from the disks supplied with this product.

Shipping iTWAIN

To ship **iTWAIN** to your customers you MUST ship the following two files ONLY - you may NOT ship any other files belonging to this product under any circumstances.

1. iTWAIN.OCX
2. iTWAIN.RTL

You are advised to install these files on the target system in the Windows System Directory:-

<WindowsDirectory> \ System

After installing these files and the other Microsoft® files identified below, you should register iTWAIN on the target system,. By running the following command :-

```
REGSVR32.EXE /s iTWAIN.OCX
```

Shipping Microsoft® Components

The following files are required on the target system :-

1. MFC42.DLL
2. OLEPRO32.DLL
3. REGSVR32.EXE

These should be installed in the target system's Windows System Directory but only if these files are either not already installed or they are a later version than those already on the target system.

When you install and register a control, you should also register OLEPRO32.DLL. Using the following command :-

```
REGSCR32.EXE /s OLEPRO32.DLL
```

Perform this registration step only if you need to install OLEPRO32.DLL. If the DLL is installed already, you should assume that it has been registered.

You should also register MFC40.DLL. Unlike OLEPRO32.DLL, you should always register this DLL, even if it is already installed. To register this DLL run the following command :-

```
REGSVR32.EXE /s MFC40.DLL
```

Shipping TWAIN files
To install TWAIN on the target system , you will need to install the following TWAIN components:-

1. TWAIN.DLL
2. TWAIN_32.DLL
3. TWUNK_16.EXE
4. TWUNK_32.DLL
5. TWSRC_16.DS
6. TWSRC_32.DS

These should be installed in the target system's Windows Directory as shown below :-

<Windows Directory> \ TWAIN.DLL
<Windows Directory> \ TWAIN_32.DLL
<Windows Directory> \ TWUNK_16.EXE
<Windows Directory> \ TWUNK_32.EXE
<Windows Directory> \ TWAIN \ TWSRC_16.DS
<Windows Directory> \ TWAIN_32 \ TWSRC_32.DS

You do not have to register any of the TWAIN files.

To UnInstall **iTWAIN** from your development system please follow these instructions :-

1. Un-Register the OCX by running the “Unregister iTWAIN OCX” command in the iTWAIN program group or by running the following command :-

```
<SystemDir> \ REGSVR32.EXE /u   <SystemDir> \ iTWAIN.OCX
```

where <SystemDir> is your windows system directory.

1. From the Control Panel, select “Add/Remove Programs”, select the iTWAIN entry in the list and press the Add/Remove button.

iTWAIN supports the following properties:

Stock Properties

BackColor	sets the control's background color
Visible	whether the control is visible or not. MUST NOT be set to FALSE

Custom Properties

AutoFeeder	whether to use an automatic page feeder on the input device (if available)
EnableTransferCancelledEvent	set this property to receive notification when an image transfer has been canceled by the user
EnableTransferCompleteEvent	set this property to receive notification when an image transfer is completed
EnableTransferReadyEvent	set this property to receive notification when image transfer is ready
EnableSourceClosedEvent	set this property to receive notification when the source is closed
ImageCount	the number of images you wish to acquire at once. Useful for acquiring multiple images or when using an autofeeder
Mode	whether to use the control in manual or automatic mode
ShowSourceProgress	whether you wish to display the input device's progress indicator.
ShowSourceUIF	whether you wish to display the input device's own user interface to the end user.
TransferMode	controls how you wish to transfer images - in Native mode or through file transfer.

See Also

Methods
Events
Examples

Use this property to specify the background color of the control.

Syntax

controlname.**BackColor** [= *color*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Color value

Setting

The **BackColor** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
color	N/A	An integer expression evaluating to a color value. Select the required color using the drop down list in the visual basic properties window.

Remarks

Use this property to set the background color of the control.   This is the color the control will display when it has no image.

This is a standard stock property for specifying whether the control is visible or not. However due to nature of the interaction between the TWAIN source manager and the control, this property should never be set to FALSE. If the control is made invisible, the TWAIN source manager will not be able to interact with the control.

Syntax

controlname.**Visible** [= *visible*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Boolean

Setting

The **Visible** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
visible	N/A	A Boolean expression (TRUE or FALSE) indicating whether the control is visible or not

Remarks

Although this property is provided, it must not be set to FALSE during a TWAIN session.   This is because TWAIN relies on the controls window handle to display the image.

If you wish to hide the control at run time, you should move its position off screen.

Use this property to specify whether to use the input device auto-feeder or not.

Syntax

controlname.**AutoFeeder** [= *autofeeder*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Boolean

Setting

The **AutoFeeder** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
autofeeder	Check the AutoFeeder box to use the auto-feeder, un-check it otherwise.	A Boolean expression (TRUE or FALSE) indicating whether to use the auto-feeder or not

Remarks

An auto-feeder, if attached to the input device, allows you to capture many images in one go.   Auto-feeders are normally available with scanners where multiple pages can be scanned automatically.

Only available in Automatic Mode.
Not Guaranteed to succeed - **iTWAIN** will TRY to negotiate this capability in Automatic Mode

Set this property if you wish to receive the TransferCancelled Event in your application.

Syntax

controlname.**EnableTransferCancelledEvent** [= *enableTCE*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Boolean

Setting

The **EnableTransferCancelledEvent** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
enableTCE	Check the Transfer Canceled box to receive the transfer canceled event, un-check it otherwise.	A Boolean expression (TRUE or FALSE) indicating whether your application should receive the transfer canceled event or not

Remarks

The TransferCanceled Event is useful especially when the input device's user interface (Source UIF) is to be displayed - see the ShowUIF property.

This means that the user could cancel the image transfer from the Source UIF and your application would wait indefinitely for the image to be acquired.   The TransferCanceled Event allows your application to recognize that an image transfer process was canceled by the user.

Set this property if you wish to receive the TransferComplete Event in your application.

Syntax

controlname.**EnableTransferCompleteEvent** [= *enableTCE*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Boolean

Setting

The **EnableTransferCompleteEvent** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
enableTCE	Check the Transfer Complete box to receive the transfer complete event, un-check it otherwise.	A Boolean expression (TRUE or FALSE) indicating whether your application should receive the transfer complete event or not

Remarks

The TransferComplete Event tells your application that a single image or frame has been acquired and the image data is ready.   Normally you would save the image data in the handler for this event.

Note that the TransferComplete Event is fired after every image or frame that is acquired - this is especially important when capturing multiple images.

Set this property if you wish to receive the TransferReady Event in your application.

Syntax

controlname.**EnableTransferReadyEvent** [= *enableTRE*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Boolean

Setting

The **EnableTransferReadyEvent** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
enableTRE	Check the Transfer Ready box to receive the transfer ready event, un-check it otherwise.	A Boolean expression (TRUE or FALSE) indicating whether your application should receive the Transfer Ready event or not

Remarks

The TransferReady Event tells your application that the input device is ready to transfer image data.

In the handler for this event, you can interrogate the control for more details about the image about to be acquired. You can also specify a filename and file type for the image when using the file transfer mode.

Set this property if you wish to receive the SourceClosed Event in your application.

Syntax

controlname.**EnableSourceClosedEvent** [= *enableSCE*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Boolean

Setting

The **EnableSourceClosedEvent** property uses the following setting.

Setting	Property Sheet	Visual Basic
enableTRE	Check the Source Closed box to receive the source closed event, un-check it otherwise.	A Boolean expression (TRUE or FALSE) indicating whether your application should receive the source closed event or not

Remarks

The SourceClosed Event tells your application that the source was closed automatically or by the CloseSource method.

Allows you to specify the number of images to acquire

Syntax

controlname.**ImageCount** [= *n*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Integer

Setting

The **ImageCount** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
n	Enter the number of images to acquire in the Image Count Edit Box. Enter -1 for unlimited images. 1 for single image >1 for n images 0 Invalid <-1 Invalid	An integer expression to specify the number of images to acquire. Set to -1 for unlimited images. 1 for single image >1 for n images 0 Invalid <-1 Invalid

Remarks

This property is useful for acquiring multiple images. If you know the number of images you wish to acquire set this property to that number.

If you do not know in advance how many images are to be acquired - for example when using an automatic sheet feeder - then set this property to -1. Your application will receive the TransferReady Event before transferring data for each and every image or frame.

Only available in Automatic Mode.
Not Guranteed to succeed - **iTWAIN** will TRY to negotiate this capability in Automatic Mode

Sets the **iTWAIN** control to Manual or Automatic Mode.

Syntax

controlname.**Mode** [= *mode*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Integer

Setting

The **Mode** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
mode	Select the Manual Radio Button for Manual Mode. Select the Automatic Radio Button for Automatic Mode (default)	An integer expression indicating the Mode : 0 - Manual 1 - Automatic (default)

Remarks

The **iTWAIN** Control can be made to operate in two modes

Manual Mode
Allows you to control the input device at a low level and select and modify the device's capabilities.   See the Initialization Methods for more information.

Automatic Mode
Allows you to capture images with a minimum of work using the default settings.

Allows you to specify whether the input device's progress indicator is to be displayed.

Syntax

controlname.**ShowSourceProgress** [= *ShowProgress*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Boolean

Setting

The **ShowSourceProgress** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
ShowProgress	Check the Show Source Progress Box to display the device progress indicator	A Boolean expression (TRUE or FALSE) indicating whether to display the device progress indicator

Remarks

When the input device is active (transferring an image) you may wish to display a progress indicator. Use this property to show the indicator

Note that all devices are not guaranteed to implement a progress indicator.   If one is not available this property has no effect.

Only available in Automatic Mode.
Not Guaranteed to succeed - **iTWAIN** will TRY to negotiate this capability in Automatic Mode

Allows you to specify whether the input device's User Interface should be displayed.

Syntax

controlname.**ShowSourceUIF** [= *ShowUIF*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Boolean

Setting

The **ShowSourceUIF** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
ShowUIF	Check the Show Source UIF Box to display the device UIF	A Boolean expression (TRUE or FALSE) indicating whether to display the device UIF

Remarks

Many manufacturers supply a customised user interface or dialog that can be used for fine tuning the settings for their input device.

You can choose to display this UIF using this property. This allows the end user to specify image capture parameters more precisely. Examples of the kind of settings that might be available are image cropping, image enlargement and reduction, multiple sheet feeders and so forth.

This property is valid in both Automatic Mode and Manual Mode

Specifies how you wish to receive the image data.

Syntax

controlname.**TransferMode** [= *TransferMode*]

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Type

Integer

Setting

The **TransferMode** property uses the following setting.

Setting	**Property Sheet**	**Visual Basic**
TransferMode	Select the Native Radio Button for Native Mode Transfers Select the File Radio Button for File Transfers	An integer expression indicating the Transfer Mode: 0 - Native Mode (Default) 1 - File Mode

Remarks

The **iTWAIN** Control is capable of transferring image data in the following modes :

Native Mode
This mode is the easiest to use. The image is stored in a block of memory that is allocated by the TWAIN source. The format it is stored in is DIB on Windows and PICT on Macintosh. Native Mode is ideal for small images. In this mode your application cannot know whether there will be enough memory until the image transfer starts. The source may then need to clip the image due to lack of memory.

File Mode
This mode transfers data directly to a disk file. This is likely to be the most convenient since most applications probably want to save the image data once acquired. For this mode you MUST set the EnableTransferReadyEvent Property to TRUE. See the SetTransferFile Method for further information.

Memory Mode
This TWAIN mode is not supported in this version of **iTWAIN**.

This property is valid in both Automatic Mode and Manual Mode

iTWAIN provides a comprehensive range of methods for controlling input devices, acquiring images and getting image information.   There are high level methods for ease of use and lower level methods for complete control.

iTWAIN methods are broadly classified as:

Initialization	Methods for initializing the Source Manager and Sources
Source Selection	Methods for selecting an input source and acquiring images from it
Image Information	Methods for getting information about an image to be acquired
Status Information	Methods for getting state and status information
High Level Capabilities	Methods for getting common capabilities
Capability Negotiation	Methods for negotiating capabilities directly with the source giving complete control

See Also

Properties
Events
Examples

InitializeSession	Initializes the TWAIN Source Manager and begins a TWAIN session
ResetSession	Resets the TWAIN Session
LoadSourceManager	Loads the TWAIN Source Manager
UnloadSourceManager	Unloads the TWAIN Source Manager
OpenSourceManager	Opens the TWAIN Source Manager
CloseSourceManager	Closes the TWAIN Source Manager
OpenSource	Opens the selected Source
CloseSource	Closes an open Source
EnableSource	Enables the Source with negotiated capabilities
DisableSource	Disables the Source and resets any negotiated capabilities

See Also

Source Selection

Image Information

Status Information

High Level Capabilities

Capability Negotiation

Begins a session between your application and the TWAIN Source Manager.

Syntax

controlname.**InitializeSession** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

Must be called once before starting the TWAIN session. This method loads the Source Manager and opens it, ready to accept the remaining method calls.   If successful, you must call ResetSession eventually to close the Source Manager.

See Also

SelectSource, Acquire, ResetSession

Ends the session between your application and the TWAIN Source Manager

Syntax

controlname.**ResetSession** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure -

Note: once the session has been reset, you cannot use GetStatus to get error information

Parameters

None

Remarks

Use this method to finally close the Source Manager.

See Also

InitializeSession

Loads the TWAIN Source Manager

Syntax

controlname.**LoadSourceManager** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure

Note: You cannot use GetStatus to get error information until the Source manager has been opened by calling OpenSourceManager()

Parameters

None

Remarks

In Automatic Mode there is no need to call this method. Use InitializeSession instead.

In Manual Mode call this to load the TWAIN Source Manager.   You must then call OpenSourceManager immediately after.

See Also

UnLoadSourceManager,   OpenSourceManager

Unloads the TWAIN Source Manager

Syntax

controlname.**UnLoadSourceManager** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure

Note: once the Source Manager has been unloaded you cannot use GetStatus to get error information

Parameters

None

Remarks

In Automatic Mode there is no need to call this method. Use ResetSession instead.

In Manual Mode call this to unload the TWAIN Source Manager.   You MUST have called LoadSourceManager earlier.

See Also

LoadSourceManager

Opens the Source Manager

Syntax

controlname.**OpenSourceManager** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

In Automatic Mode there is no need to call this method. Use InitializeSession instead.

In Manual Mode call this to open the TWAIN Source Manager.   You MUST first call LoadSourceManager.

After calling this method you would select the Source using one of the Source Selection Methods

See Also

LoadSourceManager,   CloseSourceManager

Closes an open connection with the TWAIN Source Manager

Syntax

controlname.**CloseSourceManager** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure

Note: after closing the Source Manager you cannot use GetStatus to get error information

Parameters

None

Remarks

In Automatic Mode there is no need to call this method. Use ResetSession instead.

In Manual Mode call this to close the TWAIN Source Manager.   You MUST have called OpenSourceManager first.

See Also

OpenSourceManager

Opens the selected Source

Syntax

controlname.**OpenSource** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

In Automatic Mode you need not call this method.

In Manual Mode you would call this after selecting a source using one of the Source Selection Methods. After calling OpenSource you would carry out Capability Negotiation with the source prior to calling EnableSource.

See Also

CloseSource,

Closes an Open Source

Syntax

controlname.**CloseSource** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

In Automatic Mode you need not call this method.

In Manual Mode you would not normally need to use this method. The control will automatically Disable the source and Close the Source after it has completed the acquiring all images or the process has been canceled by the user. However this method is provided so that you can chose to selectively close down the source at any time during the image acquisition process. You MUST call the DisableSource() method first before calling this method.

See Also

OpenSource

Prepares a Source with newly negotiated capabilities

Syntax

controlname.**EnableSource** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

In Automatic Mode there is no need to call this method.

In Manual Mode you would call this after calling OpenSource and Negotiating Capabilities.

Calling this function triggers off the image acquisition process in both Automatic Mode and Manual Mode. After all images have been transferred the control automatically calls DisableSource() followed by CloseSource() and TWAIN is placed back in state 3 ready for selecting another source or for negotiating properties for the next acquisition process.

See Also

DisableSource

Resets any capabilities negotiated with the Source

Syntax

controlname.**DisableSource** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

In Automatic Mode there is no need to call this method.

In Manual Mode you would not normally need to use this method. The control will automatically Disable the source and Close the source after it has completed the acquiring all images or the process has been canceled by the user. However this method is provided so that you can chose to selectively close down the source at any time during the image acquisition process. You MUST call the CloseSource() method after calling this method.

See Also

EnableSource

SelectSource	Select a Source from a list of installed TWAIN Sources using the standard TWAIN select source user interface
SelectSourceUIF	Same as SelectSource
SelectSourceDefault	Select the default TWAIN Source
SelectSourceFirst	Select the first TWAIN Source
SelectSourceNext	Select the next TWAIN Source
GetSourceName	Get the name of the currently selected Source
SelectSourceByIndex	Select the nth TWAIN Source
Acquire	Get the image(s)

See Also

- Initialization
- Image Information
- Status Information
- High Level Capabilities
- Capability Negotiation

Select a TWAIN Source from a list of available sources

Syntax

controlname.**SelectSource** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

The user is presented with a list of available TWAIN Sources. When the user selects one it becomes the selected source. This method is synonymous with SelectSourceUIF and is provided for TWAIN compliance.

See Also

SelectSourceUIF

Select a TWAIN Source from a list of available sources

Syntax

controlname.**SelectSourceUIF** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

The user is presented with a list of available TWAIN Sources. When the user selects one it becomes the selected source. This method is synonymous with SelectSource

See Also

SelectSource

Select the default TWAIN Source without displaying a list of available sources.

Syntax

controlname.**SelectSourceDefault** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

Selects the default TWAIN Source.   The default source is the first available source or the last selected source.

See Also

SelectSource

Select the first TWAIN Source without displaying a list of available sources.

Syntax

controlname.**SelectSourceFirst** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

Selects the first available TWAIN Source

See Also

SelecteSourceNext

Select the next TWAIN Source without displaying a list of available sources.

Syntax

controlname.**SelectSourceNext** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

Selects the next available TWAIN Source

See Also

SelectSourceFirst

Get the name of the currently selected source

Syntax

controlname.**GetSourceName** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

String containing name of the currently selected source

Parameters

None

Remarks

Returns the name of the currently selected source

See Also

SelectSource

Selects a source by index value without displaying a list of available sources

Syntax

controlname.**SelectSourceByIndex** (Short index)

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

index	index of the source to select

Remarks

Selects the source identified by the index parameter.   For example index = 1 would select the first source, index = 2 would select the second source and so on.

See Also

SelectSource

Tells the currently selected source to begin image acquisition

Syntax

controlname.**Acquire** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

None

Remarks

This method should be used in Automatic Mode. In this mode, the control opens the source, automatically negotiates the capabilities specified in the control's properties and then enables the source to start the image acquisition process.

You can call this method in manual mode if you do not want to negotiate any capabilities with the source. However, if you want to set one or more capabilities, you must call OpenSource() to place TWAIN in state 4, then carry out any capability negotiations you want, followed by calling EnableSource() to start the image acquisition process.

In both modes, as soon as the source has acquired the image but before the image data is transferred to your application, the TransferReady Event is fired. This event is fired before each and every image or frame data transfer.

In both modes, as soon as the source completes transferring the image data the TransferComplete Event is fired.   This event is fired after each and every image or frame that is acquired.

See Also

TransferComplete, TransferReady

Set the filename for saving an image in File Transfer Mode.

Syntax

controlname.**SetTransferFile** (String filename, Short filetype)

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

1 indicates success
0 indicates failure - use GetStatus to get further error information

Parameters

filename	Full file specification including drive, path, filename and extension
filetype	Type of image file. One of

0	TWFF_TIFF	Tagged Image File Format
1	TWFF_PICT	Macintosh PICT Format
2	TWFF_BMP	Windows Bitmap Format
3	TWFF_XBM	X-Windows Bitmap Format
4	TWFF_JFIF	JPEG File Interchange Format

Remarks

This method can only be used from the TransferReady event handler and only when the File transfer mode has been selected by setting the TransferMode property to 1 (TWSX_FILE).

For each and every image or frame that is about to be acquired, this event gives you the chance to provide a filename and file type for the image to be saved by the source. You can access the file during the TransferComplete event which will be fired after the file has been saved.

See Also

x

These methods are only available within the TransferReady Event and are used to get detailed information about the image or frame about to be transferred.

ImageGetFrameTop	Get the top (y) coordinate of a frame
ImageGetFrameBottom	Get the bottom (y) coordinate of a frame
ImageGetFrameLeft	Get the left (x) coordinate of a frame
ImageGetFrameRight	Get the right (x) coordinate of a frame
ImageGetDocumentNumber	Get the current document number
ImageGetPageNumber	Get the current page number
ImageGetFrameNumber	Get the current frame number
ImageGetXResolution	Get the X resolution of the image in pixels per unit
ImageGetYResolution	Get the Y resolution of the image in pixels per unit
ImageGetWidth	Get the width of the image in pixels
ImageGetLength	Get the length (height) of the image in pixels
ImageGetSamplesPerPixel	Gets the number of samples taken for each pixel in the image. For example, 1 for monochrome images and 3 for RGB color images.
ImageGetBitsPerPixel	Gets the pixel depth for the image's pixel type. For example if the image is grey then whether it is 4 bit grey or 8 bit grey
ImageGetBitsPerSample	Gets BitsPerPixel divided by Samples PerPixel
ImageGetPixelType	Gets the type of pixel data in the image. For example black and white, grey, RGB, CMYK etc
ImageGetCompression	Only applies to memory mode transfers which are not supported in the current version of the control. Gets the type of compression to be used during the memory transfer.
ImageIsPlanar	Whether the image is planar or chunky. Planar images are coded with the entire red plane of data first followed by the green plane data, followed by the entire blue plane. Non planar (chunky) image data interlaces each pixel from each color plane.
CopyToClipboard	Copies an acquired image to the windows clipboard

See Also

Initialization

Source Selection

Status Information

High Level Capabilities

Capability Negotiation

Use this method to determine the top co-ordinate of the rectangular frame describing the part of the image about to be acquired

Syntax

f = controlname.**ImageGetFrameTop** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Float

Parameters

None

Remarks

Returns the co-ordinate in the current units used by the source.

See Also

ICAP_UNITS capability in the TWAIN documentation

Use this method to determine the bottom co-ordinate of the rectangular frame describing the part of the image about to be acquired

Syntax

f = controlname. **ImageGetFrameBottom**()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Float

Parameters

None

Remarks

Returns the co-ordinate in the current units used by the source.

See Also

ICAP_UNITS capability in the TWAIN documentation

Use this method to determine the left co-ordinate of the rectangular frame describing the part of the image about to be acquired

Syntax

f = controlname. **ImageGetFrameLeft** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Float

Parameters

None

Remarks

Returns the co-ordinate in the current units used by the source.

See Also

ICAP_UNITS capability in the TWAIN documentation

Use this method to determine the right co-ordinate of the rectangular frame describing the part of the image about to be acquired

Syntax

f = controlname. **ImageGetFrameRight** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Float

Parameters

None

Remarks

Returns the co-ordinate in the current units used by the source.

See Also

ICAP_UNITS capability in the TWAIN documentation

Get the document number of the image to be acquired, if this ability is provided by a device using an auto feeder.

Syntax

l = controlname.**ImageGetDocumentNumber** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Long Integer

Parameters

None

Remarks

Some devices using an autofeeder can group a number of images into a single document.   This method allows you to get the document number.

See Also

x

Gets the page number of the image to be acquired, if this ability is provided by a device using an auto feeder.

Syntax

l = controlname. **ImageGetPageNumber** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Long Integer

Parameters

None

Remarks

x

See Also

x

This method returns the frame number of the frame to be acquired, if this ability is provided by a device. Some devices are capable of providing multiple rectangular frames on a single page. This method lets you know which frame is about to be acquired.

Syntax

l = controlname. **ImageGetFrameNumber** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Long Integer

Parameters

None

Remarks

x

See Also

x

This method returns the image resolution along the horizontal axis of the image. The resolution is measured in pixels per unit. For example if the source is using Inch units, then the resolution represents the number of pixels per inch

Syntax

f = controlname. **ImageGetXResolution** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Float

Parameters

None

Remarks

See Also

ICAP_XRESOLUTION capability in the TWAIN documentation
ICAP_YRESOLUTION capability in the TWAIN documentation
ICAP_UNITS capability in the TWAIN documentation

This method returns the image resolution along the vertical axis of the image. The resolution is measured in pixels per unit. For example if the source is using Inch units, then the resolution represents the number of pixels per inch

Syntax

f = controlname. **ImageGetYResolution** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Float

Parameters

None

Remarks
x

See Also
ICAP_XRESOLUTION capability in the TWAIN documentation
ICAP_YRESOLUTION capability in the TWAIN documentation
ICAP_UNITS capability in the TWAIN documentation

This method returns the number of pixels in the entire width of the image to be acquired.

Syntax

l = controlname. **ImageGetWidth** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Long Integer

Parameters

None

Remarks

x

See Also

None

This method returns the number of pixels in the entire length (height) of the image to be acquired.

Syntax

l = controlname.**ImageGetLength** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Long Integer

Parameters

None

Remarks

x

See Also

None

Returns the number of samples in each pixel
For example this is 1 for black and white and 3 for RGB images.

Syntax

s = controlname.**ImageGetSamplesPerPixel** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Short Integer

Parameters

None

Remarks

See Also
ICAP_PIXELTYPE capability in the TWAIN documentation

Returns the number of bits of data representing one pixel. Applies to the PixelType.

Syntax

s = controlname.**ImageGetBitsPerPixel** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Short Integer

Parameters

None

Remarks
This represents the bitdepth for the image's pixel type, see ImageGetSamplesPerPixel() and ImageGetPixelType()

For example if the PixelType is TWPT_GRAY then this method returns 4 for 4 bit grey and 8 for 8 bit grey images

See Also
ICAP_BITDEPTH capability in the TWAIN documentation
ICAP_PIXELTYPE capability in the TWAIN documentation

Gets the number of bits per sample for an image.

Syntax

s = controlname.**ImageGetSamplesPerSample** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Short Integer

Parameters

None

Remarks

This is equivalent to ImageGetBitsperPixel() divided by ImageGetSamplesPerPixel()

See Also
ImageGetBitsPerPixel()
ImageGetSamplesPerPixel
ICAP_BITDEPTH capability in the TWAIN documentation

Gets the type of pixel data in the image to be acquired.

Syntax

s = controlname.**ImageGetPixelType** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Short Integer

Parameters

None

Remarks

Valid values are:-

0	TWPT_BW	Black and White
1	TWPT_GRAY	Grey
2	TWPT_RGB	RGB color image
3	TWPT_PALETTE	Image containing a palette of colours
4	TWPT_CMY	CMY color image
5	TWPT_CMYK	CMYK color image
6	TWPT_YUV	YUV color image
7	TWPT_YUVK	YUVK color image
8	TWPT_CIEXYZ	CIE color image

See Also
ICAP_PIXELTYPE capability in the TWAIN documentation

Gets the type of compression that will be used for transferring the image using memory mode transfer.

Syntax

l = controlname.**ImageGetCompression** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Long Integer

Parameters

None

Remarks
Memory mode transfer is not supported in the current version of this control.

Valid return values are

0	TWCP_None
1	TWCP_PACKBITS
2	TWCP_GROUP31D
3	TWCP_GROUP31DEOL
4	TWCP_GROUP32D
5	TWCP_GROUP34
6	TWCP_JPEG
7	TWCP_LZW

See Also
ICAP_COMPRESSION capability in the TWAIN documentation

Checks whether the image is planar or chunky.

Syntax

b = controlname.**ImageIsPlanar** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Boolean

Parameters

None

Remarks

Returns TRUE if the image color data is planar and FALSE if it is chunky.

Planar data is presented as an entire plane of data for one color followed by an entire plane for the next color. For example, if the image is RGB color and planar, then the image data is presented first as the entire red plane, then the green plane followed by the blue plane. On the other hand if the data is chunky, each pixel is represented by the red, green and blue values.

Planar data is organized like this:-
    RRRRRR, GGGGGGGG, BBBBBBBB

Chunky data is organized like this:-
    RGBRGBRGBRGBRGBRGBRGBRGB

Usually single pass scanners generate chunky data and multiple pass scanners generate planar data

See Also
ICAP_PLANARCHUNKY capability in the TWAIN documentation

Copies an acquired image to the windows clipboard

Syntax

i = controlname.CopyToClipboard ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Short Integer

Parameters

None

Remarks
Use this method to copy image data to the windows clipboard.   Your appication can then use clipboard functions to transfer the bitmap data as required.

Valid return values are

0	Could not copy to clipboard or no image to copy
1	Copied successfully. The clipboard now contains the image data

See Also
None

These methods allow you to get the current TWAIN State information and the status of the last operation.

GetTwainState	Get the current TWAIN State
GetStatus	Get the current Status Code

See Also

- Initialization
- Source Selection
- Image Information
- High Level Capabilities
- Capability Negotiation

Gets the current TWAIN State

Syntax

s = controlname.**GetTwainState** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Short Integer	current TWAIN State

Parameters

None

Remarks

TWAIN is implemented as a state machine - each operation puts TWAIN into one of seven states:

State 1	Initial state and after calling ResetSession
State 2	After calling LoadSourceManager or CloseSourceManager
State 3	After calling InitializeSession, OpenSourceManager, CloseSource or automatically after all transfers are completed
State 4	After calling OpenSource or DisableSource
State 5	After calling EnableSource
State 6	Automatically when an image is ready to be transferred
State 7	Automatically during the image transfer process

When using Manual Mode it is useful to always know which state TWAIN is in at any time

See Also

None

Get the current TWAIN status code

Syntax

s = controlname.**GetStatus** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Short Integer	current TWAIN Status Code

Parameters

None

Remarks

Gets the status of the latest TWAIN Operation.   After every operation this status code is set as follows:

Value	Code	Description
0	TWCC_SUCCESS	It worked !
1	TWCC_BUMMER	Failure due to unknown causes
2	TWCC_LOWMEMORY	Not enough memory to perform operation
3	TWCC_NODS	No Data Source
4	TWCC_MAXCONNECTIONS	DS is connected to max possible apps
5	TWCC_OPERATIONERROR	DS or DSM reported error, app shouldn't
6	TWCC_BADCAP	Unknown capability
9	TWCC_BADPROTOCOL	Unrecognized MSG DG DAT combination
10	TWCC_BADVALUE	Data parameter out of range
11	TWCC_SEQERROR	DG DAT MSG out of expected sequence
12	TWCC_BADDEST	Unknown destination App/Src in DSM_Entry

See Also

None

These methods are provided for ease of use - they allow common capabilities to be checked

IsDeviceOnLine	Checks to see if device is on-line
IsFeederEnabled	Checks to see if an auto-feeder is enabled
IsUIControllable	Checks to see if the Source UIF can be controlled
GetCurrentUnits	Gets the current unit of measurement of the source
GetPhysicalHeight	Gets the physical height of the source
GetPhysicalWidth	Gets the physical width of the source
GetTimeDate	Gets the current time and date

See Also

Initialization

Source Selection

Image Information

Status Information

Capability Negotiation

Indicates whether the device is on line and switched on

Syntax

b = controlname.**IsDeviceOnLine** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Boolean

Parameters

None

Remarks
Returns TRUE if the device is available
Returns FALSE if the device is not available, or if the source cannot support this inquiry

See Also
CAP_DEVICEONLINE capability in the TWAIN documentation

Indicates whether the auto feeder is enabled

Syntax

b = controlname.**IsFeederEnabled** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Boolean

Parameters

None

Remarks

Returns TRUE if the auto feeder is enabled
Returns FALSE if the auto feeder is not enabled, or if the source cannot support this inquiry

See Also
CAP_FEEDERENABLED capability in the TWAIN documentation

Indicates whether the Source UIF can be disabled or not.

Syntax

b = controlname.**IsUIControllable** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Boolean

Parameters

None

Remarks

Returns TRUE if a source supports acquisition with the user interface disabled
Returns FALSE otherwise.

The ShowSourceUIF property will have no effect if this method returns FALSE.

See Also
CAP_UICONTROLLABLE capability in the TWAIN documentation

Gets the current unit of measurement used by the source

Syntax

s = controlname.**GetCurrentUnits** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Short s	Unit Code. Valid return values are	
	0	TWUN_INCHES
	1	TWUN_CENTIMETERS
	2	TWUN_PICAS
	3	TWUN_POINTS
	4	TWUN_TWIPS
	5	TWUN_PIXELS

Parameters

None

Remarks

This tells you unit of measurement being used by the source.

See Also
ICAP_UNITS capability in the TWAIN documentation

Gets the maximum physical height (y-axis) of an image the source is capable of acquiring.

Syntax

f = controlname.**GetPhysicalHeight** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Float

Parameters

None

Remarks
Effectively this is the physical dimension of the device and specifies the maximum size of image that can be acquired.

The value is returned in the current units used by the device.

See Also
ICAP_UNITS capability in the TWAIN documentation

Gets the maximum physical width (x-axis) of an image the source is capable of acquiring.

Syntax

f = controlname.**GetPhysicalWidth** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Float

Parameters

None

Remarks
Effectively this is the physical dimension of the device and specifies the maximum size of image that can be acquired.

The value is returned in the current units used by the device.

See Also
ICAP_UNITS capability in the TWAIN documentation

Gets the date and time when the image was acquired

Syntax

s = controlname.**GetTimeDate** ()

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

String

Parameters

None

Remarks
The string is formatted as follows:-
YYYY/MM/DD HH:MM:ss.sss

where

YYYY	is the year
MM	is the month
DD	is the date
HH	is the hour
MM	is the number of minutes
SS	is the number of seconds and
sss	is the number of milliseconds

See Also
CAP_TIMEDATE capability in the TWAIN documentation

GetCapability	Get the iTWAIN Capability Object. You MUST get this capability object first in order to use the following capability methods
IsSupported	Check whether this capability is supported by the Source
SelectCurrent	Select the current value of this capability
SelectDefault	Select the default value of this capability
SelectAvailable	Select available values of this capability
ResetDefault	Reset default value of this capability
IsSingle	Check if this capability has a single value
IsEnumeration	Check if this capability has enumerated values
IsArray	Check if this capability has an array of values
IsRange	Check if this capability has a range of values
GetValue	Get the value of this capability
SetValue	Set the value of this capability
GetSingleValue	Get a single value
SetSingleValue	Set a single value
GetMinValue	Get the minimum value of a range
SetMinValue	Set the minimum value of a range
GetMaxValue	Get the maximum value of a range
SetMaxValue	Set the maximum value of a range
GetStepSize	Get the step size of a range
GetMultipleCount	Get number of values
NegotiateSingle	Negotiate a single value
NegotiateEnumeration	Negotiate enumerated values
NegotiateArray	Negotiate an array of values
NegotiateRange	Negotiate a range of values
SetFrame	Set frame size
GetFrameLeft	Get frame’s left coordinate
GetFrameRight	Get frame’s right coordinate
GetFrameTop	Get frame’s top coordinate

GetFrameBottom	Get frame’s bottom coordinate

See Also

- Initialization
- Source Selection
- Image Information
- Status Information
- High Level Capabilities

Gets a capability object.

Syntax

Declare obj As Object
set obj = controlname.**GetCapability** (Short capability)

controlname is the name of the **iTWAIN** Control object, for example, iTWAIN1.

Return Value

Object obj	The capability Object

Parameters

Short capability	The required TWAIN capability. See TWAIN Capabilities for a list of valid capabilities

Remarks

TWAIN supports a range of capabilities that may be negotiated with a source. **iTWAIN** makes it easy to negotiate any capability by providing a Capability Object.

Use this method to get an **iTWAIN** capability object. A capability is returned as an OLE object. You can use this object to negotiate capability values.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

None

Returns whether the capability is supported by the source.

Syntax

b = obj.**IsSupported**

obj is the name of a local variable of type Object in Visual Basic

Return Value

Boolean b	TRUE if this capability is supported, FALSE otherwise

Parameters

None

Remarks

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

None

Tells the source that we are about the inquire about the current setting for the specified capability

Syntax

obj.**SelectCurrent()**

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

None

Remarks

Sources can return the current value, the preferred default value all the available (or valid) values for a given capability.

Typically the current settings will be returned as a Single value.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

SelectDefault
SelectAvailable
ResetDefault

Tells the source that we are about the inquire about the default setting for the specified capability

Syntax

obj.**SelectDefault**()

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

None

Remarks

Sources can return the current value, the preferred default value all the available (or valid) values for a given capability.

Typically the default settings will be returned as a Single value.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

SelectCurrent
SelectAvailable
ResetDefault

Tells the source that we are about the inquire about the available settings for the specified capability

Syntax

obj.**SelectAvailable**()

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

None

Remarks

Sources can return the current value, the preferred default value and all the available (or valid) values for a given capability.

The available settings will be returned as Single, Enumerations, Arrays or Range depending on how the source provides those values

In Visual Basic you MUST set the capability object to "Nothing" after use. (set obj = Nothing)

See Also

SelectCurrent
ResetDefault
SelectDefault

Specifies whether the capability value returned by the source is a single value

Syntax

b = obj.**IsSingle**

obj is the name of a local variable of type Object in Visual Basic

Return Value

Boolean b	TRUE if capability is single type, FALSE otherwise

Parameters

None

Remarks

Capability values can be returned in one of 4 container types, Single, Enumerations, Arrays or a range of values.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

IsEnumeration
IsArray
IsRange
GetSinglevalue
SetSingleValue
NegotiateSingle

Specifies whether the capability value returned by the source is an enumeration of values

Syntax

b = obj.**IsEnumeration**

obj is the name of a local variable of type Object in Visual Basic

Return Value

Boolean b	TRUE if capability is enumeration type, FALSE otherwise

Parameters

None

Remarks

Capability values can be returned in one of 4 container types, Single, Enumerations, Arrays or a range of values.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

- IsSingle
- IsArray
- IsRange
- GetValue
- SetValue
- NegotiateEnumeration

Specifies whether the capability value returned by the source is a array of values

Syntax

b = obj.**IsArray**

obj is the name of a local variable of type Object in Visual Basic

Return Value

Boolean b	TRUE if capability is array type, FALSE otherwise

Parameters

None

Remarks

Capability values can be returned in one of 4 container types, Single, Enumerations, Arrays or a range of values.

In Visual Basic you MUST set the capability object to "Nothing" after use. (set obj = Nothing)

See Also

IsSingle
IsEnumeration
IsRange
GetValue
SetValue
NegotiateArray

Specifies whether the capability value returned by the source is a range of value

Syntax

b = obj.**IsRange**

obj is the name of a local variable of type Object in Visual Basic

Return Value

Boolean b	TRUE if capability is range type, FALSE otherwise

Parameters

None

Remarks

Capability values can be returned in one of 4 container types, Single, Enumerations, Arrays or a range of values.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

IsSingle
IsArray
IsEnumeration
GetMinValue
GetMaxValue
GetStepSize
SetMinValue
SetMaxValue
NegotiateRange

This method is used to retrieve the maximum value of a capability which is provided by the source as a range of values

Syntax

v = obj.**GetMaxValue**()

obj is the name of a local variable of type Object in Visual Basic

Return Value

VARIANT v	maximum value of the capability

Parameters

None

Remarks

You must allocate a variable for the return value to correspond to the type of the capability
The variant return type can be used for any capability type except the FRAME type

Twain capability type	**C / C++ data type**	**Visual Basic Data type**
UINT8	unsigned char	Integer
UINT16	unsigned Short	Long
INT16	Short	Integer
BOOL	typedef BOOL Short	Boolean
FIX32	Float	Single
STR32	char[32+1]	String
STR128	char[128+1]	String
STR255	char{255+1]	String

NOTE: most sources will only use the FIX32 data type in specifying a range of values.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

GetMinValue
GetStepSize
SetMinValue
SetMaxValue
NegotiateRange

This method is used to retrieve the minimum value of a capability which is provided by the source as a range of values

Syntax

v = obj.**GetMinValue**()

obj is the name of a local variable of type Object in Visual Basic

Return Value

VARIANT v	minimum value of the capability

Parameters

None

Remarks

You must allocate a variable for the return value to correspond to the type of the capability
The variant return type can be used for any capability type except the FRAME type

Twain capability type	C / C++ data type	Visual Basic Data type
UINT8	unsigned char	Integer
UINT16	unsigned Short	Long
INT16	Short	Integer
BOOL	typedef BOOL Short	Boolean
FIX32	Float	Single
STR32	char[32+1]	String
STR128	char[128+1]	String
STR255	char{255+1]	String

NOTE: most sources will only use the FIX32 data type in specifying a range of values.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

GetMaxValue
GetStepSize
SetMinValue
SetMaxValue
NegotiateRange

This method is used to retrieve the step size of a capability which is provided by the source as a range of values

Syntax

v = obj.**StepSize**()

obj is the name of a local variable of type Object in Visual Basic

Return Value

VARIANT v	step size of the capability range

Parameters

None

Remarks

You must allocate a variable for the return value to correspond to the type of the capability
The variant return type can be used for any capability type except the FRAME type

Twain capability type	**C / C++ data type**	**Visual Basic Data type**
UINT8	unsigned char	Integer
UINT16	unsigned Short	Long
INT16	Short	Integer
BOOL	typedef BOOL Short	Boolean
FIX32	Float	Single
STR32	char[32+1]	String
STR128	char[128+1]	String
STR255	char{255+1]	String

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

NOTE: most sources will only use the FIX32 data type in specifying a range of values.

See Also

GetMinValue
GetMaxValue
SetMinValue
SetMaxValue
NegotiateRange

This method is used to set the maximum value of a capability which can provided to the source as a range of values

Syntax

obj.**SetMaxValue**(VARIANT data)

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

VARIANT data	data value to be set

Remarks

You must allocate a variable for the parameter value to correspond to the type of the capability
The variant type can be used for any capability type except the FRAME type

Twain capability type	**C / C++ data type**	**Visual Basic Data type**
UINT8	unsigned char	Integer
UINT16	unsigned Short	Long
INT16	Short	Integer
BOOL	typedef BOOL Short	Boolean
FIX32	Float	Single
STR32	char[32+1]	String
STR128	char[128+1]	String
STR255	char{255+1]	String

NOTE: most sources will only use the FIX32 data type in specifying a range of values.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

GetMinValue
GetMaxValue
GetStepSize
SetMinValue
NegotiateRange

This method is used to set the minimum value of a capability which can provided to the source as a range of values

Syntax

obj.**SetMinValue**(VARIANT data)

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

VARIANT data	data value to be set

Remarks

You must allocate a variable for the parameter value to correspond to the type of the capability
The variant type can be used for any capability type except the FRAME type

Twain capability type	**C / C++ data type**	**Visual Basic Data type**
UINT8	unsigned char	Integer
UINT16	unsigned Short	Long
INT16	Short	Integer
BOOL	typedef BOOL Short	Boolean
FIX32	Float	Single
STR32	char[32+1]	String
STR128	char[128+1]	String
STR255	char{255+1]	String

NOTE: most sources will only use the FIX32 data type in specifying a range of values.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

GetMinValue
GetMaxValue
GetStepSize
SetMaxValue
NegotiateRange

This method is used to attempt to negotiate the minimum and maximum values provided to the source using the SetMinValue and SetmaxValue methods

Syntax

obj.**NegotiateRange**()

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

None

Remarks

Sources are not required to set the values you have specified. You must verify that the source has indeed set the values you wanted to set by retrieving the current value.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

NegotiateSingle
NegotiateEnumeration
NegotiateArray

This method is used to retrieve the count of the number of values of a capability which is provided by the source as an Enumeration or Array

Syntax

s = obj.**GetMultipleCount()**

obj is the name of a local variable of type Object in Visual Basic

Return Value

Short s	number of capability values

Parameters

None

Remarks

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

None

This method is used to retrieve the value of a capability which is provided by the source as an Enumeration or an Array

Syntax

v = obj.**GetValue**(Short index)

obj is the name of a local variable of type Object in Visual Basic

Return Value

VARIANT v	required data value

Parameters

Short index	index into enumeration or array

Remarks

You must allocate a variable for the return value to correspond to the type of the capability
The variant return type can be used for any capability type except the FRAME type

Twain capability type	**C / C++ data type**	**Visual Basic Data type**
UINT8	unsigned char	Integer
UINT16	unsigned Short	Long
INT16	Short	Integer
BOOL	typedef BOOL Short	Boolean
FIX32	Float	Single
STR32	char[32+1]	String
STR128	char[128+1]	String
STR255	char{255+1]	String

The index is 1 based and not zero based. The first value is obtained by calling GetValue(1) and so on.
The index must not exceed the count obtained from GetMultipleCount()

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

SetValue
GetSingleValue
SetSingleValue

This method is used to set the value of a capability which can be provided to the source as an enumeration or an Array

Syntax

obj.**SetValue**(Short index, VARIANT data)

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

Short index	index into enumeration or array
VARIANT data	data value to be set

Remarks

You must allocate a variable for the parameter value to correspond to the type of the capability
The variant type can be used for any capability type except the FRAME type

Twain capability type	**C / C++ data type**	**Visual Basic Data type**
UINT8	unsigned char	Integer
UINT16	unsigned Short	Long
INT16	Short	Integer
BOOL	typedef BOOL Short	Boolean
FIX32	Float	Single
STR32	char[32+1]	String
STR128	char[128+1]	String
STR255	char{255+1]	String

The index is 1 based and not zero based. The first value is set by calling SetValue(1) and so on.
The last index value used will determine how many values will get negotiated when you eventually call NegotiateEnumeration() or NegotiateArray()

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

GetValue
GetSingleValue
SetSingleValue
NegotiateEnumeration
NegotiateArray

This method is used to attempt to negotiate the enumeration values provided to the source using the SetValue method

Syntax

obj.**NegotiateEnumeration()**

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

None

Remarks

Sources are not required to set the values you have specified. You must verify that the source has indeed set the values you wanted to set by retrieving the current value.

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

NegotiateSingle
NegotiateArray
NegotiateRange

This method is used to attempt to negotiate the array of values provided to the source using the SetValue method

Syntax

obj.**NegotiateArray**()

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

None

Remarks

Sources are not required to set the values you have specified. You must verify that the source has indeed set the values you wanted to set by retrieving the current value.

In Visual Basic you MUST set the capability object to "Nothing" after use. (set obj = Nothing)

See Also

NegotiateSingle
NegotiateEnumeration
NegotiateRange

This method is used to retrieve the value of a capability which is provided by the source as a single value

Syntax

v = obj.**GetSingleValue**

obj is the name of a local variable of type Object in Visual Basic

Return Value

VARIANT v	single capability value

Parameters

None

Remarks

You must allocate a variable for the return value to correspond to the type of the capability
The variant return type can be used for any capability type except the FRAME type

Twain capability type	C / C++ data type	Visual Basic Data type
UINT8	unsigned char	Integer
UINT16	unsigned Short	Long
INT16	Short	Integer
BOOL	typedef BOOL Short	Boolean
FIX32	Float	Single
STR32	char[32+1]	String
STR128	char[128+1]	String
STR255	char{255+1]	String

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

SetSinglevalue
GetValue
SetValue

This method is used to set the value of a capability which can by provided to the source as a single value

Syntax

obj.**SetSingleValue**(variant)

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

None

Remarks

You must allocate a variable for the parameter value to correspond to the type of the capability
The variant type can be used for any capability type except the FRAME type

Twain capability type	C / C++ data type	Visual Basic Data type
UINT8	unsigned char	Integer
UINT16	unsigned Short	Long
INT16	Short	Integer
BOOL	typedef BOOL Short	Boolean
FIX32	Float	Single
STR32	char[32+1]	String
STR128	char[128+1]	String
STR255	char{255+1]	String

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

GetSingleValue
SetValue
GetValue
NegotiateSingle

This method is used to attempt to negotiate the Single value provided to the source using the SetSingleValue method

Syntax

obj.**NegotiateSingle**()

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

None

Remarks

Sources are not required to set the values you have specified. You must verify that the source has indeed set the values you wanted to set by retrieving the current value.

In Visual Basic you MUST set the capability object to "Nothing" after use. (set obj = Nothing)

See Also

NegotiateEnumeration
NegotiateArray
NegotiateRange

Sets the value of a capability back to its default value.
The default value is defined by the source and cannot be changed.

Syntax

obj.**ResetDefault()**

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

None

Remarks

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

None

This method can be used to specify one or more rectangular frames on a given page.

Syntax

obj. **SetFrame** (Short index, Float left, Float right, Float top, Float bottom)

obj is the name of a local variable of type Object in Visual Basic

Return Value

None

Parameters

Short index	index of the frame being set (1 based)
Float left	left coordinate of the frame
Float right	right coordinate of the frame
Float top	top coordinate of the frame
Float bottom	bottom coordinate of the frame

Remarks
The index value is 1 based and not zero based. 1 specifies the first frame and so on.

The last index value specified will determine the number of frames which will be set when you eventually call NegotiateEnumeration() or NegotiateArray()

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also
GetFrameLeft
GetFrameRight
GetFrameTop
GetFrameBottom
NegotiateEnumeration
NegotiateArray

This method retrieves the left co-ordinate of the frame specified

Syntax

f = obj.**GetFrameLeft**(Short index)

obj is the name of a local variable of type Object in Visual Basic

Return Value

Float f	left co-ordinate of the frame

Parameters

Short index	index of the frame

Remarks

The index is 1 based and not zero based. The left coordinate of the first frame is obtained by calling GetFrameLeft(1) and so on. The index must not exceed the count obtained from GetMultipleCount()

In Visual Basic you MUST set the capability object to "Nothing" after use. (set obj = Nothing)

See Also

SetFrame
GetFrameRight
GetFrameTop
GetFrameBottom
NegotiateEnumeration
NegotiateArray

This method retrieves the right co-ordinate of the frame specified

Syntax

f = obj.**GetFrameRight**(Short index)

obj is the name of a local variable of type Object in Visual Basic

Return Value

Float f	right co-ordinate of the frame

Parameters

Short index	index of the frame

Remarks

The index is 1 based and not zero based. The right coordinate of the first frame is obtained by calling GetFrameRight(1) and so on. The index must not exceed the count obtained from GetMultipleCount()

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

SetFrame
GetFrameLeft
GetFrameTop
GetFrameBottom
NegotiateEnumeration
NegotiateArray

This method retrieves the top co-ordinate of the frame specified

Syntax

f = obj.**GetFrameTop**(Short index)

obj is the name of a local variable of type Object in Visual Basic

Return Value

Float f	Top coordinate of the frame

Parameters

Short index	index of the frame

Remarks

The index is 1 based and not zero based. The top coordinate of the first frame is obtained by calling GetFrameTop(1) and so on. The index must not exceed the count obtained from GetMultipleCount()

In Visual Basic you MUST set the capability object to “Nothing” after use. (set obj = Nothing)

See Also

SetFrame
GetFrameLeft
GetFrameRight
GetFrameBottom
NegotiateEnumeration
NegotiateArray

This method retrieves the bottom co-ordinate of the frame specified

Syntax

f = obj.**GetFrameBottom**(Short index)

obj is the name of a local variable of type Object in Visual Basic

Return Value

Float f	Bottom coordinate of the frame

Parameters

Short index	index of the frame

Remarks

The index is 1 based and not zero based. The bottom coordinate of the first frame is obtained by calling GetFrameLBottom(1) and so on. The index must not exceed the count obtained from GetMultipleCount()

In Visual Basic you MUST set the capability object to "Nothing" after use. (set obj = Nothing)

See Also

SetFrame
GetFrameLeft
GetFrameTop
GetFrameRight
NegotiateEnumeration
NegotiateArray

iTWAIN generates the following events:

TransferReady
TransferComplete
TransferCancelled
SourceClosed

See Also

- Properties
- Methods
- Examples

The TransferReady event occurs when the source is ready to transfer some image data.

Syntax

Sub controlname_**TransferReady**()

The TransferReady event procedure uses the following argument.

Argument	**Description**
controlname	The name of the **iTWAIN** Control object, for example, iTWAIN1.

Remarks

A source can indicate that it has image data ready for your application.   The TransferReady event useful when you need to get information about the image that your application is about to receive.   You can also specify a filename and file type for the image when using the file transfer mode.

This event is fired before each and every image or frame is transferred.

See Also

EnableTransferReadyEvent

The TransferComplete event occurs when the source has finished transferring the image data

Syntax

Sub controlname_**TransferComplete**()

The TransferComplete event procedure uses the following argument.

Argument	**Description**
controlname	The name of the **iTWAIN** Control object, for example, iTWAIN1.

Remarks

This event occurs once the source has completed transferring the image data to your application.

This event is fired after each and every image or frame is transferred

See Also

EnableTransferCompleteEvent

The TransferCancelled event occurs when the user cancels or aborts image acquisition

Syntax

Sub controlname_**TransferCancelled**()

The TransferCancelled event procedure uses the following argument.

Argument	**Description**
controlname	The name of the **iTWAIN** Control object, for example, iTWAIN1.

Remarks

This event is fired if the user cancels or aborts the image acquisition. This is useful when the user selects Acquire but subsequently cancels the acquisition, for example from the Source UIF.

See Also

EnableTransferCancelledEvent

The SourceClosed event occurs after the source has been closed down.

Syntax

Sub controlname_**SourceClosed**()

The SourceClosed event procedure uses the following argument.

Argument	**Description**
controlname	The name of the **iTWAIN** Control object, for example, iTWAIN1.

Remarks

This event occurs when a source is closed either automatically (no more sheets left in the auto feeder for example) or manually via the CloseSource method.

This is useful when acquiring multiple images. Although the TransferComplete event tell you when each image is transferred it does not tell you that all pending images have been acquired. The SourceClosed event allows you to check this.

See Also

EnableSourceClosedEvent

In order to help you get started with using **iTWAIN** we have included various sample applications with this product.

<u>Visual Basic and Visual C++ Samples</u>

AutoMode is an example of using the OCX in Automatic mode and shows how it can be used for rapidly creating a powerful TWAIN compliant image acquisition application in a matter of minutes.

ManualMode is an example of a slightly more complex application which provides control over interrogating all TWAIN capabilities and negotiating them with the TWAIN source.

These applications are installed in the directory where you chose to install iTWAIN as follows :-

```
<InstallDirectory>\Samples\VB\Automode\Automode.vbp
<InstallDirectory>\Samples\VB\Manualmode\Manualmode.vbp

<InstallDirectory>\Samples\Mfc\Automode\Automode.mdp
<InstallDirectory>\Samples\Mfc\Manualmode\Manualmode.mdp
```

<u>Other Samples</u>
Sample applications in Microsoft Word97, Excel97, Access97 and Visual Foxpro 5 are also included which demonstrate how to embed and use the iTWAIN ocx control in these applications.

These applications are installed in the directory where you chose to install iTWAIN as follows :-

```
<InstallDirectory>\Samples\Word97
<InstallDirectory>\Samples\Excel97
<InstallDirectory>\Samples\Access97
<InstallDirectory>\Samples\Foxpro5
```

The following table gives details about all TWAIN capabilities which sources may support, including the data types to be used from C/ C++ and Basic. The table also details which type of containers may be used for retrieving and setting these capabilities. Note that some capabilities are READ ONLY and cannot be set. Please refer to the TWAIN documentation for more details about each capability.

key for container types:

S	Single Value
E	Enumerated Values
A	Array of Values
R	Range of Values

Capability	**TWAIN Data Type**	**C / C++ Data Type**	**Basic Data Type**	**Supporte d Container Types for Reading**	**Supported Container Types for Writing**
CAP_AUTHOR	STR128	char	String	S	S
CAP_AUTOFEED	BOOL	BOOL	Boolean	S	S
CAP_CAPTION	STR255	char	String	S	S
CAP_CLEARPAGE	BOOL	BOOL	Boolean	S	S
CAP_DEVICEONLINE	BOOL	BOOL	Boolean	S	READ ONLY
CAP_EXTENDEDCAPS	UINT16	unsigned short	Long	A	A
CAP_FEEDERENABLED	BOOL	BOOL	Boolean	S	S
CAP_FEEDERLOADED	BOOL	BOOL	Boolean	S	READ ONLY
CAP_FEEDERPAGE	BOOL	BOOL	Boolean	S	S
CAP_INDICATORS	BOOL	BOOL	Boolean	S	S
CAP_REWINDPAGE	BOOL	BOOL	Boolean	S	S
CAP_SUPPORTEDCAPS	UINT16	unsigned short	Long	A	READ ONLY
CAP_TIMEDATE	STR32	char	String	S	READ ONLY
CAP_UICONTROLLABLE	BOOL	BOOL	Boolean	S	READ ONLY
CAP_XFERCOUNT	INT16	short	Integer	S	S
ICAP_AUTOBRIGHT	BOOL	BOOL	Boolean	S	S
ICAP_BITDEPTH	UINT16	unsigned short	Long	S, E	S, E
ICAP_BITDEPTHREDUCTION	UINT16	unsigned short	Long	S, E	S, E
ICAP_BITORDER	UINT16	unsigned short	Long	S, E	S
ICAP_BITORDERCODES	UINT16	unsigned short	Long	S, E	S
ICAP_BRIGHTNESS	FIX32	float	Single	S, E, R	S, E, R
ICAP_CCITTKFACTOR	UINT16	unsigned short	Long	S	S
ICAP_COMPRESSION	UINT16	unsigned short	Long	S, E	S, E
ICAP_CONTRAST	FIX32	float	Single	S, E, R	S, E, R
ICAP_CUSTHALFTONE	UINT8	char	Integer	A	A
ICAP_EXPOSURETIME	FIX32	float	Single	S, E, R	S, E, R
ICAP_FILTER	UINT16			S, A	S, A
ICAP_FLASHUSED	BOOL	BOOL	Boolean	S	S
ICAP_FRAMES	FRAME			S, E	S, E
ICAP_GAMMA	FIX32	float	Single	S	S
ICAP_HALFTONES	STR32	char	String	S, E, A	S, E, A
ICAP_HIGHLIGHT	FIX32	float	Single	S, E, R	S, E, R
ICAP_IMAGEFILEFORMAT	UINT16	unsigned short	Long	S, E	S, E
ICAP_JPEGPIXELTYPE	UINT16	unsigned short	Long	S, E	S, E
ICAP_LAMPSTATE	BOOL	BOOL	Boolean	S, E	S
ICAP_LIGHTPATH	UINT16	unsigned short	Long	S, E	S

ICAP_LIGHTSOURCE	UINT16	unsigned short	Long	S, E	S, E
ICAP_MAXFRAMES	UINT16	unsigned short	Long	S	S
ICAP_ORIENTATION	UINT16	unsigned short	Long	S, E	S, E
ICAP_PHYSICALHEIGHT	FIX32	float	Single	S	READ ONLY
ICAP_PHYSICALWIDTH	FIX32	float	Single	S	READ ONLY
ICAP_PIXELFLAVOR	UINT16	unsigned short	Long	S, E	S
ICAP_PIXELFLAVORCODES	UINT16	unsigned short	Long	S, E	S
ICAP_PIXELTYPE	UINT16	unsigned short	Long	S, E	S, E
ICAP_PLANARCHUNKY	UINT16	unsigned short	Long	S, E	S, E
ICAP_ROTATION	FIX32	float	Single	S, E, R	S
ICAP_SHADOW	FIX32	float	Single	S, E, R	S, E, R
ICAP_SUPPORTEDSIZES	UINT16	unsigned short	Long	S, E	S, E
ICAP_THRESHOLD	FIX32	float	Single	S, E, R	S, E, R
ICAP_TILES	BOOL	BOOL	Boolean	S	S
ICAP_TIMEFILL	UINT16	unsigned short	Long	S, R	S
ICAP_UNDEFINEDIMAGESIZE	BOOL	BOOL	Boolean	S	S
ICAP_UNITS	UINT16	unsigned short	Long	S, E	S, E
ICAP_XFERMECH	UINT16	unsigned short	Long	S, E	S, E
ICAP_XNATIVERESOLUTION	FIX32	float	Single	S, E	READ ONLY
ICAP_XRESOLUTION	FIX32	float	Single	S, E, R	S, E, R
ICAP_XSCALING	FIX32	float	Single	S, E, R	S, E, R
ICAP_YNATIVERESOLUTION	FIX32	float	Single	S, E	READ ONLY
ICAP_XRESOLUTION	FIX32	float	Single	S, E, R	S, E, R
ICAP_YSCALING	FIX32	float	Single	S, E, R	S, E, R

Select this option for Manual Mode.   Click here for more details.

Select this option for Automatic Mode. Click here for more details.

If checked, enables the TransferReady Event. Click here for more details.

If checked, enables the TransferComplete Event. Click here for more details.

If checked, enables the TransferComplete Event.   Click here for more details.

If checked, uses an automatic sheet feeder if available. Click here for more details

Specifies the number of images to acquire.
Enter -1 for unlimited,   1 for a single image, 2 or more for that many images. Click here for more details

Select this option to use Native Transfer Mode. Click here for more details

Select this to use File Transfer Mode. Click here for more details

Select this to use Memory Transfer Mode. Click here for more details

Check this to display the source device’s own progress indicator. Click here for more details.

If checked, enables the SourceClosed Event.   Click here for more details.

Check this to display the source device’s own user interface. Click here for more details.