



**TWAIN**  
*Linking Images With Applications*

# **ObjectTWAIN™ 3.1**

ActiveX Library

**User's Guide**



**Copyright © 2001, JFL Peripheral Solutions, Inc.**  
All rights reserved.

Phone: (613) 728-2521  
Fax: (613) 728-4459  
<http://www.jflinc.com>  
E-mail: [info@jflinc.com](mailto:info@jflinc.com)

The information contained in this manual is based on information available at the time of publication and is subject to change without notice. Accuracy and completeness are not warranted or guaranteed.

No part of this manual may be reproduced or transmitted in any form or by any means, including electronic medium or machine-readable form, without the express written permission of JFL Peripheral Solutions, Inc. ObjectTWAIN™ is a trademark of JFL Peripheral Solutions, Inc. Other brand or product names are trademarks of their respective holders.

1. INTRODUCTION .....	1
2. FEATURES.....	1
3. SOFTWARE REQUIREMENTS .....	2
4. OBJECTTWIN™ ACTIVEX LIBRARY OCX INSTALLATION.....	3
5. START EXAMPLE .....	4
6. PROGRAMMING GUIDE .....	6
6.1. OBJECTTWIN™ ACQUISITION SESSION STATES .....	6
6.2. HOW TO HANDLE ERROR CONDITIONS.....	8
6.2.1. General Rules .....	8
6.2.2. Handling ObjectTWIN™ Operation Errors .....	8
6.2.2.1. OLE Exceptions .....	8
6.2.2.2. The GetLastError method .....	10
6.3. HOW TO SELECT A DATA SOURCE AND OPEN A TWAIN SESSION .....	11
6.3.1. Select and use the default TWAIN Data Source .....	11
6.3.2. Select and use a custom, fixed TWAIN Data Source .....	11
6.3.3. Use a custom select source dialog .....	12
6.4. HOW TO NEGOTIATE CAPABILITIES .....	13
6.4.1. How to iterate through an enumeration, array or range of capability values .....	14
6.4.1.1. Arrays and Enumerations of values .....	14
6.4.1.2. Ranges of values .....	14
6.4.1.3. Example how to get the capabilities reported as supported by the Data Source.....	15
6.4.1.4. Example how to get the x-axis resolution values supported by the Data Source .....	17
6.4.2. How to set to Data Source an enumeration or array of capability values .....	20
6.4.2.1. Use a TWContainer object returned by a GetAvailable or Get operation .....	20
6.4.2.2. Use a new TWContainer object.....	20
6.5. HOW TO NEGOTIATE THE ACQUISITION FRAME LAYOUT .....	22
6.5.1.1. Visual Basic example .....	22
6.5.1.2. Delphi 4 example .....	23
6.5.1.3. PowerBuilder example .....	24
6.6. HOW TO START THE IMAGE ACQUISITION .....	25
6.7. HOW TO HANDLE TRANSFERRED IMAGE DATA AND OTHER TRANSFER EVENTS .....	26
6.8. HOW TO GET IMAGE INFORMATION .....	29
6.9. HOW TO GET EXTENDED IMAGE INFORMATION .....	30
6.10. HOW TO USE THE AUTOMATIC DOCUMENT FEEDER.....	33
7. LIBRARY REFERENCE .....	34
R.1. OBJECTTWIN™ ACTIVEX/COM CONTROLS .....	34
R.2. OTHER OBJECTTWIN™ COM CONTROLS.....	35
R.3. TWCONTROL ACTIVEX CONTROL PROPERTIES.....	37
R.4. TWCONTROL ACTIVEX CONTROL BASIC METHODS .....	39
R.5. TWCONTROL ACTIVEX CONTROL DEFINED CAPABILITY METHODS .....	46
R.6. TWCONTROL ACTIVEX EVENTS .....	63
R.6. OTHER OBJECTTWIN™ COM OBJECTS INTERFACES .....	64
R.7. OBJECTTWIN™ COM CONSTANT ENUMERATIONS .....	66



## 1. Introduction

TWAIN provides a standard for software developers to access Image Acquisition devices including scanners, digital cameras and multi-function peripherals. The **ObjectTWAIN™ ActiveX Library** provides support for any Application developed in any ActiveX/COM-aware programming environment to access the entire TWAIN Application Programming Interface (API) through an easy-to-use ActiveX/COM interface. Using the ObjectTWAIN™ ActiveX an application can select, open and manage completely with minimum coding requirements and at the highest possible programmatic control TWAIN image acquisition sessions with any TWAIN-compatible Data Source deserving an image acquisition device.

## 2. Features

- High level, object oriented approach to TWAIN image acquisition programming, accessible for Applications developed in any ActiveX/COM-aware programming environment such as Visual Basic®, Visual C++®, Delphi™ and PowerBuilder™.
- Complete coverage of TWAIN protocol to assure access to any part of mandatory, optional or custom functionality that may be supported by a Data Source.
- Fully supports high-speed production scanning.
- Provides compressed and disk file image transfer capabilities even when the Data Source does not support the optional Compressed Memory Transfer or File Transfer mechanism through conversion of Native and Memory transfer modes including compressed Memory Transfers to disk file transfers. Supported software compression and file formats include JPEG Interchange Format (JFIF), Tagged Image File Format (TIFF): CCITT Group 3.1d, Group 3.2d or Group 4 compressed and multi-page.
- Support for advanced TWAIN capability negotiation including support for custom capabilities and custom capability values: supports any type of defined capability operation and container type: enumeration, range, array for any type defined by TWAIN for capability values (such as boolean, frame, string, floating point). TWAIN capability containers are accessed as COM Collections.
- Automatic feeder loaded detection feature: the application may run in the background and activate when the user loads the scanner's feeder with paper.

---

### Note:

ObjectTWAIN™ will not use its internal compression engine when the Data Source will support to and the application will successfully negotiate with the Data Source to deliver compressed image data. Instead ObjectTWAIN™ will use in this case the device's hardware compression engine in order to achieve maximum transfer speed.

---

### 3. Software Requirements

- Microsoft® Windows® 95/98/NT4/2000/XP Operating System.
- ActiveX/COM aware programming environment, such as Visual Basic®, Visual C++®, Delphi™ and PowerBuilder™ (only on the development platform).
- TWAIN Data Source Manager and Twunker files (TWAIN.DLL, TWAIN\_32.DLL, TWUNK16/32.EXE residing in the <Windows> directory.
- TWAIN Release v1.5..v1.8 compliant Data Source deserving any type of image acquisition device (scanner, digital camera, multi-function peripheral): file with extension \*.DS residing in the <Windows>\TWAIN\_32\ or <Windows>\TWAIN\ directory.
- TWAIN Specification document (currently updated for TWAIN Release v1.8). This document is available for free download at [www.twain.org](http://www.twain.org). It is recommended to be used a reference for the TWAIN triplet operations and capabilities used through ObjectTWAIN™.
- TWAIN.H (TWAIN API C/C++ header file): together with the TWAIN Spec this file describes all the constants defined by TWAIN (optional, not needed for Visual Basic or Delphi, may be used when using the ActiveX to develop an application using Visual C++).

---

**Note:**

Although ObjectTWAIN™ ActiveX redefines and may export as COM Enumerations to applications the TWAIN capability constants, because of the lack of support for these COM Enumeration objects in some programming environments (such as PowerBuilder) you may need to directly use some capability values with their numerical values. You may find these values either in the TWAIN Specification document, TWAIN.H or the reference tables in this document.

**Example:**

TWPT\_BW is a value defined as 0 by TWAIN (TWAIN.H and TWAIN Specification). PT\_BW is the related enumerated value exported by ObjectTWAIN™. PT\_BW is only an alias for TWPT\_BW, the numerical value is the same, 0. A Visual Basic application will call SetPixelFormat(PT\_BW), a Visual C++ application, if TWAIN.H is included, will call SetPixelFormat(TWPT\_BW), a PowerBuilder application may call SetPixelFormat(0) (if the TWPT\_BW constant is not defined in the PowerBuilder application code).

---

## 4. ObjectTWAIN™ ActiveX Library OCX Installation

There are two different types of installations for the ObjectTWAIN™ ActiveX Library:

A. **On the development platform** (where the application using the OCX will be compiled):

Execute the setup.exe installation program which comes with ObjectTWAIN™. When prompted, enter the product serial number. The OCX will be installed to the <Windows>\<System> directory (e.g. C:\WINNT\SYSTEM32\ObjTWAIN.ocx) and will be automatically registered for application development purposes. Then you will be able to load and use the OCX in your programming environment.

---

**IMPORTANT:**

An application built with ObjectTWAIN™ ActiveX Library will contain in its code the run-time license for ObjectTWAIN™. You must not install the ObjectTWAIN™ OCX using the provided ObjectTWAIN installation program on the client machines where the ObjectTWAIN application will run since this will install the design-time license and will enable anyone to use the OCX on these machines to develop unauthorized ObjectTWAIN™ applications. Instead, copy ObjTWAIN.ocx and register it as described by the next paragraph.

---

B. **On the client platform** (where the application using ObjectTWAIN™ OCX will run):

When installing your application, **copy** the ObjTWAIN.ocx file to the <Windows>\<System> directory (the directory named \SYSTEM for Win95/98 and \SYSTEM32 for WinNT) then **register** it like any other OCX control.

Usually any tool used to build the application installation package should have an option available to enable the installation script to automatically register OCX controls. If the tool you are using to build the installation package for your application does not provide this feature, you may use the Windows **REGSVR32.EXE** utility: you may either call it directly from the Windows Start Menu REGSVR32 or call from your installation program code the **WinExec** Windows API function to launch REGSVR32.EXE specifying the complete path for ObjTWAIN.ocx as parameter.

## 5. Start Example

This chapter describes the steps to develop a minimal ObjectTWAIN™ application in Visual Basic Version 6.0 to select a TWAIN Data Source and to acquire and display one image from it.

**Step 1:** Start Visual Basic. Choose *New Project | Standard EXE*. A project named Project1 with one form called Form1 will be displayed.

**Step 2:** Select *Project | Components* and check the component called *Object TWAIN ActiveX Library*. If this component is not listed here this means that it was not correctly installed. Refer to the Install section about how to install and register the ObjectTWAIN™ ActiveX for design and development purposes.

The control shall be listed in the Toolbox list of controls (if this is not visible, select *View | Toolbox*) with the name TWControl and the following bitmap:



**Step 3:** Select the control and drag it on the form. A new control called TWControl1 will be created. In the properties window make sure that the NativeMemoryAsFile property is enabled, the NativeMemoryFileType property is set to 2 – DIB and the Quiet property is disabled.

**Step 4:** Add a CommandButton (named Command1) and set its caption to “&Select Source”. Double-click the button and add the following code to display the TWAIN Data Source Manager Select Source dialog:

```
Private Sub Command1_Click()
    On Error GoTo errSelectSource
    TWControl1.UserSelectSource
    Exit Sub
errSelectSource:
    MsgBox "Error on Select Source or cancel pressed"
End Sub
```

**Step 5:** Add another CommandButton (named Command2) and set its caption to “&Acquire”. Double click this newly added button and add the following code to open a TWAIN session with the selected Data Source and start the image acquisition. The code also sets the current value of the CAP\_XFERCOUNT capability to 1 (meaning that the application is willing to transfer one image in one TWAIN session) by calling TWControl1.SetXferCount. The transfer mode used is Native. Because the TWControl.NativeMemoryAsFile property is enabled (set to True) the memory Native DIB image will be saved by ObjectTWAIN™ to a disk file. The acquisition will be done with the Data Source User Interface displayed (the parameter of StartNativeAcquisition):

```
Private Sub Command2_Click()
    On Error GoTo errAcquire
    TWControl1.OpenSession hWnd
    TWControl1.SetXferCount 1
    TWControl1.StartNativeAcquisition True
    Exit Sub
errSelectSource:
    MsgBox Err.Description
    TWControl1.CloseSession
End Sub
```

**Step 6:** Add a PictureBox control, named Picture1. This PictureBox control will be used to display the acquired image.

**Step 7:** Double-click the TWControl1 control and add the following code to receive the transferred image from the Data Source and display it using the PictureBox control. The acquisition will be stopped after one image transfer is completed: if the Data Source will have more images available for transfer the application (in this example) will not continue to transfer the next image:

```
Private Sub TWControl1_DoProcessNativeAsFileData(ByVal FileFormat As
OBJTWINLibCtl.tagImageFileFormat, ByVal FileName As String)
    On Error GoTo errProcessData
    Picture1.Picture = LoadPicture(FileName)
    TWControl1.StopAcquisition
    Exit Sub
errProcessData:
    MsgBox Err.Description
    TWControl1.StopAcquisition
End Sub
```

**Step 8:** Double-click the TWControl1 control and add the following code to handle special transfer conditions that may appear: the user may close the Data Source UI without initiating an image transfer, the Data Source may fail the transfer because of an acquisition hardware problem or the user may simply cancel the transfer while processing. All the transferred data and transfer error notifications are signaled by ObjectTWIN™ as ActiveX events:

```
Private Sub TWControl1_OnCancelTransfer()
    MsgBox "Transfer canceled"
    TWControl1.CloseSession
End Sub

Private Sub TWControl1_OnEndTransfers()
    TWControl1.CloseSession
End Sub

Private Sub TWControl1_OnTransferError(ByVal Error As
OBJTWINLibCtl.tagObjectTwainError, ByVal ErrorDescription As String)
    MsgBox ErrorDescription
    TWControl1.StopAcquisition
End Sub

Private Sub TWControl1_OnUserCloseSourceUI(ByVal CloseOK As Boolean)
    MsgBox "Source UI closed"
    TWControl1.CloseSession
End Sub
```

**Step 9:** Save and run the EXE application. Press *Select Source* to display the DSM dialog and select a Data Source. This will be the default TWAIN Data Source in the system and will be preserved across multiple TWAIN sessions until a new Source is installed or set as default. Press *Acquire* to start the TWAIN acquisition and transfer one image to the application. Make sure you start the scan from the Data Source User Interface: because the Data Source UI is displayed, the 'Transfer Ready' signal will be not triggered automatically by the Data Source.

## 6. Programming Guide

### 6.1. ObjectTWAIN™ Acquisition Session States

TWAIN describes seven states that exist in TWAIN sessions. ObjectTWAIN™ provides a simplified model described by only four states. The following table lists the ObjectTWAIN™ states, the relation with the TWAIN states and how the application may cause a state transition:

ObjectTWAIN™ State	TWAIN State(s)	ObjectTWAIN™ State Transition
<b>0- ObjectTWAIN™ not started</b>	<b>1 - DSM not loaded</b>	TO NEXT STATE:  Transition to next state when a TWControl is created at run time (e.g. when the form containing the control is created).
<b>1- ObjectTWAIN™ started</b>  TO DO:  In this state the application may enumerate the available Data Sources, get Data Source information and/or select a Data Source.  RESTRICTIONS:  In this state the application cannot do state any capability negotiation or request to start acquisition since no Data Source is yet loaded and opened.	<b>2 - DSM loaded</b> <b>3 - DSM opened</b>	TO NEXT STATE:  Transition to next state when a session with a Data Source is opened (when OpenSession is successfully executed).  TO PREVIOUS STATE:  Transition to previous state when the current instance of TWControl is destroyed.
<b>2 – Source Session opened</b>  TO DO:  In this state the application may do capability negotiation (see if a capability is supported, get/set capability values) and negotiate the image layout frame.  <b>Note:</b> A single ObjectTWAIN™ application may open and manage multiple Source sessions simultaneously by using multiple TWControl instances.	<b>4 - Source opened</b>	TO NEXT STATE:  Transition to next state when the ObjectTWAIN™ acquisition is started (e.g. one of the TWControl StartXAcquisition methods is executed).  TO PREVIOUS STATE:  Transition to previous state when the current Source Session is closed (after CloseSession is executed).

*(continue on next page)*

(continue from previous page)

ObjectTWAIN™ State	TWAIN State(s)	ObjectTWAIN™ State Transition
<p><b>3 – Acquisition</b></p> <p>TO DO:</p> <p>In this state the application may wait to receive image data transferred from the Source and/or to be notified about special conditions that may appear during the acquisition sequence. Also the application may request non-image data (e.g. barcode data) after a transfer, ask information about the next image to be transferred and continue or stop the acquisition.</p> <p><b>Note:</b> CloseSession may be called at any time: if the acquisition is started it will first stop the acquisition then close the session.</p> <p>RESTRICTIONS:</p> <p>In this state the Application may not request to change any capability value unless that capability was negotiated in the previous state as extended (through Get/ SetExtendedCaps).</p>	<p><b>5 - Source enabled</b>  <b>6 - Transfer ready</b>  <b>7 – Transferring data</b></p>	<p>TO PREVIOUS STATE:</p> <p>Transition to previous state when the acquisition sequence is interrupted (by executing StopAcquisition) or when all the available images are transferred to application.</p>

All TWAIN operations (including Capability operations) are exposed by ObjectTWAIN™ ActiveX Library as methods of an invisible at run-time ActiveX control called **TWControl**. The transferred image data and the transfer errors and special conditions are signaled through TWControl ActiveX events. The ObjectTWAIN™ ActiveX Library also exposes other COM objects to wrap various TWAIN data structures (e.g. TWContainer which provides access to TWAIN TW\_CAPABILITY data and capability containers such as TW\_ARRAY). All COM objects exported by the Library may be accessed as OLE Automation objects.

In the next sections the basic application operation procedures will be explained (in the order that may appear in an ObjectTWAIN™ session). Some of the techniques described will be accompanied by code examples in Visual Basic or other programming languages.

## 6.2. How To Handle Error Conditions

The big part of the TWAIN API is described by the TWAIN Specification to be only optional to be implemented by Data Sources. Worst, there are few Data Sources which really correctly support correctly the minimal set of mandatory TWAIN operations and capabilities. A well designed TWAIN application must be able to identify and handle any kind of error which has occurred in a TWAIN session, and gracefully manage the session with any Data Source.

### 6.2.1. General Rules

- The application may try to use any kind of optional TWAIN functionality but the application shall not rely on such optional TWAIN functionality to succeed and be able to transfer images from a Data Source. If the Data Source does not support an optional Transfer Mode simply switch to another mode instead of terminating the session prematurely.
- Always check the result of a TWAIN operation: all TWControl methods mapped to TWAIN operations may provide two different error result handling mechanisms (see below).
- It is recommended before using a capability to check if the capability is reported as supported (through GetSupportedCaps). Also, before trying to set a capability value get the values the Data Source supports for that capability. Do not assume that the Data Source will preserve current capability values across sessions.
- The application may interrupt the current acquisition sequence at any time by calling TWControl.**StopAcquisition**. After this the application may try to start other acquisition sequence (by optionally adjusting capability values and calling one of the TWControl.StartXAcquisition methods) or terminate the entire TWAIN session (by calling TWControl.**CloseSession**). It is important to know that CloseSession may be called at any time to terminate the current session: it even calls StopAcquisition automatically if needed. If no session is opened at the time CloseSession is called nothing happens.

### 6.2.2. Handling ObjectTWAIN™ Operation Errors

An application may retrieve the result of a TWControl method in two different modes:

- Handling OLE Exceptions;
- Using the TWControl.GetLastError method.

Each of these two methods is explained in one of the following sub-sections:

#### 6.2.2.1. OLE Exceptions

When an error is encountered ObjectTWAIN™ (if configured appropriately) throws an exception object. The application may handle such an exception by having a try-catch exception-handling block around the failed operation call.

If the application does not handle one exception in its code, it is the default exception handler for the application that will handle this exception: usually this will display a message to the user and terminate the application. While such a behavior (to exit the application at the first TWAIN error encountered) may be useful in the beginning of the application development phase it shall not be used on purpose in the final version of the application: any attempt to get an unsupported capability or to set an unsupported capability value will cause an exception and will terminate the entire application instead of continuing the session and simply consider that capability or value as unsupported by the Data Source.

The major advantage of this method is its simplicity: the application may provide a single error-handling block for an entire sequence of TWAIN operations. The drawback would be that an exception will be thrown for every error encountered, the application will be unable to choose to continue for non-fatal errors like the one caused when the user closes the DSM Select Source dialog by pressing cancel. Using the second error handling method the application may decide for which error codes to prompt the user or execute special error action.

To use this mechanism the TWControl **Quiet** property must be disabled (set to False), either at design or run-time.

The following examples show how to provide a handler for an error passed by ObjectTWAIN™ as an OLE exception (the TWControl.SetFeederEnabled and SetAutoFeed methods will be used):

Visual Basic example:

```
Private Sub SetFeederBtn_Click()
On Error GoTo errSetFeeder
    If TWControl.GetState <> OPENED Then
        MsgBox "No session opened"
        Exit Sub
    End If
    TWControl.SetFeederEnabled True
    TWControl.SetAutoFeed True
    Exit Sub
errSetFeeder:
    MsgBox Err.Description
End Sub
```

Delphi 3 example:

```
try
    TWControl.SetFeederEnabled(True);
    TWControl.SetAutoFeed(True);
except
    Application.MessageBox('SetADF Error', 'Scan', MB_YESNO)
end;
```

Visual C++ example:

```
try
{
    m_twControl.SetFeederEnabled(TRUE);
    m_twControl.SetAutoFeed(TRUE);
}
catch(COleDispatchException *error)
{
    AfxMessageBox(error->m_strDescription);
}
```

### 6.2.2.2. The **GetLastError** method

ObjectTWIN™ provides an alternate method to retrieve result codes by using a special TWControl method named **GetLastError**. This method will return the status of the last TWAIN operation executed. This method may be especially useful when the application development environment or programming language does not support an exception handling mechanism.

To use this mechanism the TWControl **Quiet** property must be enabled (set to True), either at design or run-time. When the Quiet property is enabled, ObjectTWIN™ does not throw an OLE exception when an error is encountered.

Visual Basic example:

```
TWControl.SetFeederEnabled True
If OBJTWAIN_NO_ERROR <> TWControl.GetLastError Then
    MsgBox "SetFeederEnabled failed"
End If
```

Delphi 3 example:

```
//TWError : tagObjectTwainError;

TWControl1.SetFeederEnabled(True);
TWError := TWControl.GetLastError();
if TWError <> OBJTWAIN_NO_ERROR then
    Application.MessageBox('SetFeederEnabled failed',
        'Error', MB_ICONSTOP)
```

PowerBuilder example:

```
integer result = 0

this.object.SetFeederEnabled(True)
result = this.object.GetLastError
if (512 <> result) then //512 is OBJTWAIN_NO_ERROR
    MessageBox("SetFeederEnabled failed", "Error")
end if
```

---

#### Note:

You may see the values defined by ObjectTwainError in the “ObjectTWIN™ COM Constant Enumerations” reference section at the end of the document.

---

## 6.3. How To Select a Data Source and Open a TWAIN Session

### 6.3.1. Select and use the default TWAIN Data Source

This is the most simple to implement and the most robust mechanism to select and open a session with a Data Source. This technique adheres to the model recommended in the TWAIN Specification. Usually an application will have two User Interface command elements (e.g. menu items or buttons), one to select the default source and another one to open the default source and start the acquisition.

Use the TWControl **UserSelectSource** method to display the default Source Select dialog and let the user to select the default TWAIN Data Source. This default will be preserved across TWAIN sessions until a new Data Source is installed on the system or another default is selected.

Use the TWControl **OpenSession** method to open a session with the default Data Source.

Visual Basic example:

```
TWControl.OpenSession Form1.hWnd
```

Delphi example:

```
TWControl.OpenSession(Application.Handle, '');
```

Visual C++/MFC example:

```
m_twControl.OpenSession((long)m_hWnd, _T(""));
```

### 6.3.2. Select and use a custom, fixed TWAIN Data Source

This may be useful for a special purpose application that may need to be dedicated to only one acquisition device. The second parameter of **OpenSession** may be used to pass as a character string the name of the Data Source to open. This is the same string listed in the Select Source dialog and it corresponds to the TW\_IDENTITY.ProductName reported by the Data Source.

Visual Basic example:

```
TWControl.OpenSession Form1.hWnd, "ScanPartner 15C"
```

### 6.3.3. Use a custom select source dialog

ObjectTWAIN™ ActiveX Library exports two COM objects useful in this situation: **TWSourceInfo** (a wrapper object for the TWAIN TW\_IDENTITY structure) and **TWSourceList** (a collection of TWSourceInfo objects). The TWControl **GetInstalledSources** method may be used to get the list of installed Data Sources information as a TWSourceList object.

First you have to display to the user the names of the Data Sources to select one from. You may display the names for all the Data Source installed or select a subset of the available Data Sources (based on the information contained by the TWSourceInfo object for each Data Source found).

Visual Basic example: how to fill a ListBox object (List1) with the names of all TWAIN Data Sources installed on the system:

```
Dim SourceList As TWSourceList
Dim SourceInfo As TWSourceInfo
Set SourceList = TWControl.GetInstalledSources
For Each SourceInfo In SourceList
    List1.AddItem SourceInfo.ProductName
Next
Set SourceList = Nothing
```

Then, to open the selected Data Source, call the TWControl **OpenSession** method passing the selected Source name as the second parameter:

```
Private Sub SelectBtn_Click()
    TWControl.OpenSession hWnd, List1.List(List1.ListIndex)
End Sub
```

## 6.4. How To Negotiate Capabilities

ObjectTWAIN™ provides access to any possible capability operation (Get, GetCurrent, GetDefault, Reset, Set, QuerySupport) on any capability, either defined by TWAIN or custom defined, through **TWControl** methods and the **TWContainer** COM object. TWContainer is a wrapper object for capability containers (arrays, enumerations and range of values). Where possible, simple types are used (such as integer, boolean or string). In all other cases, a reference to a TWContainer object will be passed to a Set function or returned by a Get operation.

### IMPORTANT:

- Do not assume that any Data Source will support a capability (even though this would be a capability described by the TWAIN Spec to be mandatory to be supported by any Data Source). Always check for error codes signaling an unsupported capability, operation on capability or even a general failure caused by an unexpected problem into the Data Source.
- It is recommended to always check the TWContainer objects (returned by Get/ GetAvailable/ GetCurrent/ GetDefault and Reset operations) to be valid (the VARIANT/ANY variable received needs to be a valid reference to a TWContainer COM object) and the type of container (TWContainer.Type may be ON\_RANGE, ON\_ENUMERATION or ON\_ARRAY) before trying to read the container values (this is specially important when negotiating capabilities that may accept multiple container types).
- It is recommended to check first if a capability is supported (by using GetSupportedCaps), get the available/supported values (by using the GetAvailable or Get method related to that capability) and only then try to change (set) the current or available set of values for that capability. By isolating a problem in the Data Source the application may avoid using a badly implemented Data Source feature that may cause the entire application to hang or crash.
- Do not try to set a capability after the acquisition has been started unless that capability was previously negotiated as extended (through TWControl.Get/SetExtendedCaps). Also make sure a Data Source Session is opened before executing any capability operation.

The next sub-sections will describe some of the most frequent cases of capability negotiation.

#### 6.4.1. How to iterate through an enumeration, array or range of capability values

All `GetAvailable` on enumerated and range type capabilities (e.g. `ICAP_PIXELTYPE`, `ICAP_XRESOLUTION`) or `Get` on array-type capabilities (e.g. `CAP_EXTENDED CAPS`) TWControl methods return when successful a reference to a TWContainer COM object 'filled' with the array, enumeration or range of capability values returned by the Data Source as result to the TWAIN capability operation requested.

##### 6.4.1.1. Arrays and Enumerations of values

There are two possible methods of iterating the values from an enumeration or array of values contained by a TWContainer object:

- Use the automatic OLE Collection iteration feature. This method is currently supported only by the most recent versions of Visual Basic through the **for each** statement.
- Iterate the values by using the special designed methods exposed by TWContainer: **Count()** to get the number of items in container and **Item(Index)** to retrieve an item at the given index. These methods shall be accessible from any programming environment able to understand standard OLE Automation interfaces (all the COM interfaces exposed by ObjectTWAIN™ objects are *dual* interfaces).

---

##### Notes:

- All indexes for objects exposed by ObjectTWAIN ActiveX Library are 1-based in Visual Basic style: the index of the first item in a list is 1 (not 0).
- A TWContainer object is actually a collection of VARIANT objects: a VARIANT element may be of any possible type, from a boolean value to a reference to a complex COM object expressed as a pointer to an IDispatch interface. For capability operations the most 'complex' COM object that may be (under normal conditions) contained by a TWContainer is a TWFrame object (object used for `GetAvailable/GetCurrent/GetDefault/Reset/SetIFrames`).

---

To access the current and/or the default value in a TWContainer enumeration an enumeration TWContainer object exposes the **CurrentIndex** and **DefaultIndex** properties.

##### 6.4.1.2. Ranges of values

The TWContainer objects expose a set of properties to enable an application to access them as ranges of values: **RangeMin**, **RangeMax**, **RangeStep**, **RangeCurrent** and **RangeDefault** (when the container type is `ON_RANGE`).

### 6.4.1.3. Example how to get the capabilities reported as supported by the Data Source

The application may use TWControl **GetSupportedCaps** method to ask the Data Source to return an array of the supported capabilities. GetSupportedCaps will return the array as a **TWContainer** collection of **TWCapabilityName** objects or simple **integer** values (depending how the collection is accessed, as a collection of objects or integers):

Visual Basic example (no error checking code; the 'for each' approach is used):

```
Dim Caps As TWContainer
Dim Capability As TWCapabilityName
Set Caps = TWControl.GetSupportedCaps
For Each Capability In Caps
  ` do something, e.g.:
  If Capability = IC_ORIENTATION Then
    bSupportsOrientation = True
    Exit For
  End If
Next
Set Caps = Nothing
```

Delphi 3 example (no error checking code; the Count/Item(Index) approach is used):

```
//SupportedCapsCon : OleVariant;
//SourceSupportsOrientation : Boolean;
//i : Integer;

SupportedCapsCon := TWControl.GetSupportedCaps;
for i := 1 to SupportedCapsCon.Count do
begin
  // do something, e.g.:
  if SupportedCapsCon.Item[i].Value = IC_ORIENTATION then
    bSupportsOrientation := True;
end;
endSupportedCapsCon := UnAssigned;
```

---

#### Note:

In this case the items in the TWContainer object are references to TWCapabilityName COM objects. In the first example it was not needed to explicitly use the TWCapabilityName.Value property, Visual Basic was smart enough to automatically convert the integer value to the default property of the TWCapabilityName object. The second example shows how to explicitly use this property when the programming environment is not able to automatically make the conversion.

---

## Visual C++ example:

```

BOOL bIcapOrientationSupported = FALSE;

try
{
    VARIANT varSupportedCaps = m_twControl.GetCapSupportedCaps();
    //
    // Proceed only if received a valid pointer to a Dispatch interface:
    //
    if((VT_DISPATCH == varSupportedCaps.vt) && (varSupportedCaps.pdispVal))
    {
        //
        // Connect the received IDispatch to a TWContainer object:
        //
        ITWContainer SupportedCaps(varSupportedCaps.pdispVal);

        //
        // Iterate through the TWContainer and search for ICAP_ORIENTATION:
        //
        int nSupportedCaps = SupportedCaps.GetCount();
        COleVariant varOrientationCapName((long)ICAP_ORIENTATION);

        for(int i = 1; i <= nSupportedCaps; i++)
        {
            COleVariant varCapabilityName = SupportedCaps.GetItem(i);

            //
            // Make sure that we compare here an integer value:
            // (VT_I4 stands for '4 bytes signed integer)
            //
            //
            // (the varCapabilityName may be an IDispatch to an
            // TWCapabilityName object; in this case may be better
            // to access directly this VARIANT as an integer)
            //
            varCapabilityName.ChangeType(VT_I4);

            //if(ICAP_ORIENTATION == varCapabilityName.lVal)
            if(varOrientationCapName == varCapabilityName)
            {
                bIcapOrientationSupported = TRUE;
                break;
            }
        }
    }
}
catch(COleDispatchException *)
{
    // CAP_SUPPORTEDCAPS is not supported
}

```

#### 6.4.1.4. Example how to get the x-axis resolution values supported by the Data Source

The application may use TWControl **GetAvailableXResolution** method to ask the Data Source to return an enumeration or range of the supported x-axis resolutions. The range or enumeration of values will be returned by GetAvailableXResolution as a **TWContainer** collection of floating point values (not integer because ICAP\_XRESOLUTION values are defined by TWAIN as TW\_FIX32). If the Data Source returns a TWON\_ONEVALUE container ObjectTWAIN™ will make automatically an enumeration containing only one item which will also be the current and default one. The application will have to read the TWContainer **Type** property before trying to access the resolution values as an **enumeration** or as a **range** of values (TWAIN defines both container types to be allowed for DAT\_CAPABILITY/ MSG\_GET on ICAP\_XRESOLUTION).

Visual Basic example (no error checking code):

```
Dim Xres As TWContainer
Set Xres = TWControl.GetAvailableIXResolution

If Xres.Type = ON_ENUMERATION Then

    For Each Item In Xres
        ' do something: add the resolutions to a ListBox control
        List1.AddItem Item
    Next

    ' read the current and default values:
    MsgBox Xres(Xres.CurrentIndex), vbOKOnly, "Current x res"
    MsgBox Xres(Xres.DefaultIndex), vbOKOnly, "Default x res"
Else
    k = 0

    Do
        Value = Xres.RangeMin + k * Xres.RangeStep
        If Value > Xres.RangeMax Then
            Exit Do
        End If
        List1.AddItem Value
        k = k + 1
    Loop

    MsgBox Xres.RangeMin, vbOKOnly, "Min x res"
    MsgBox Xres.RangeMax, vbOKOnly, "Max x res"
    MsgBox Xres.RangeCurrent, vbOKOnly, "Current x res"
    MsgBox Xres.RangeDefault, vbOKOnly, "Default x res"
End If
```

## Delphi 3 example:

```

// XResCon : OleVariant;
// SourceSupportsXRes, NeedToSetXRes : Boolean;
// i : Integer;
// ConfigRes, rmin, rmax, rdef, rstep, rcur, f : Double;
// TWError : tagObjectTwainError;

//
// Check if the Data Source supports a 'configured' x-resolution
// and set it to be the current value for this session
// (if not already selected in the DS):
//
SourceSupportsXRes := False;
NeedToSetXRes := True;

try

XResCon := TWControl.GetAvailableIXResolution;

TWError := TWControl.GetLastError;
if TWError = OBJTWAIN_NO_ERROR then
begin
  Assert(varDispatch = VarType(XResCon));
  if XResCon.Type = ON_RANGE then
  begin
    // Check basically only if the configured value is in the
    // requested range (no exact step coverage estimation)
    if (XResCon.RangeMin <= ConfigRes) and
        (XResCon.RangeMax >= ConfigRes) then
    begin
      SourceSupportsXRes := True;
      if XResCon.RangeCurrent = ConfigRes then
      begin
        // No need to set XResolution
        NeedToSetXRes := False;
      end;
    end
  else
  begin
    SourceSupportsXRes := False;
    // Configured XResolution outside available range
  end;
end
else if XResCon.Type = ON_ENUMERATION then
begin
  for i := 1 to XResCon.Count do
  begin
    if XResCon.Item[i] = ConfigRes then
    begin
      SourceSupportsXRes := True;

```

*(continued on next page)*

*(continued from previous page)*

```
        if i = XResCon.CurrentIndex then
        begin
            // XRes already set, no need to set again
            NeedToSetXRes := False;
        end;
        break;
    end;
end;

if SourceSupportsXRes = True and NeedToSetXRes = True then
begin
    TWControl.SetIXResolution(ConfigRes);

    TWError := TWControl.GetLastError;
    if TWError <> OBJTWAIN_NO_ERROR then
    begin
        SourceSupportsXRes := False;
    end;
end;
end;
except
    SourceSupportsXRes := False;
end;
```

### 6.4.2. How to set to Data Source an enumeration or array of capability values

It is recommended to use this technique when the application wants to try to limit the set of available values for an enumerated-type capability (e.g.: limit the pixel types supported by the Data Source in the current session only to B&W and Grayscale – if: (1) the Data Source supports TrueColor by default and the user can change the pixel type from the Data Source UI and (2) the application does not want to receive RGB image data and (3) the Source accepts the application's request to exclude the RGB pixel type from its set of supported pixel types in this session, then the Data Source shall no longer let the user set RGB from its UI – e.g. by disabling the related button in its interface).

There are two different methods to get a TWContainer object to pass the enumeration of values to the Data Source:

#### 6.4.2.1. Use a TWContainer object returned by a GetAvailable or Get operation

Visual Basic example (no error checking code): get the pixel types supported by the Data Source, then remove RGB if present from the enumeration and set it back to the Data Source:

```
Dim PixelTypes As TWContainer
Dim pt As TWPixelFormatValue
Set PixelTypes = TWControl.GetAvailableIPixelType

i = 1
LimitedRGB = False
For Each pt In PixelTypes
    If pt = PT_RGB Then
        PixelTypes.Remove i
        LimitedRGB = True
        Exit For
    End If
    i = i + 1
Next

If LimitedRGB Then
    TWControl.SetIPixelType PixelTypes
End If

Set PixelTypes = Nothing
```

#### 6.4.2.2. Use a new TWContainer object

The only special thing here is that the application has to construct a new COM object, in this case TWContainer:

##### Step 1: Construct a new TWContainer object

Visual Basic example:

```
Dim NewPixelTypes As New TWContainer
```

Delphi example:

```
NewPixelTypes := CoTWContainer.Create;
```

**PowerBuilder example:**

```
OLEObject NewPixelTypes
integer result = 0
NewPixelTypes = CREATE OLEObject
if not IsNull(NewPixelTypes) then
    result = NewPixelTypes.ConnectToNewObject("ObjTWAIN.TWContainer")
    if 0 > result or IsNull(result) then
        /* insert here error handling code */
    else
        /* initialize the new TWContainer */
    end if
end if
```

**Step 2: Initialize the container type****Visual Basic example:**

```
NewPixelTypes.InitEnumeration 2
```

**Notes:**

- In the above example 1 is the number of items in the new enumeration.
- To initialize an Array or Range use TWContainer.InitArray respectively InitRange.

**Step 3: Add values to the container****Visual Basic example (no error checking code):**

```
NewPixelTypes.Add PT_BW
NewPixelTypes.Add PT_GRAY
```

**Step 4: Set to Data Source the container****Visual Basic example (no error checking code):**

```
TWControll1.SetIPixelType NewPixelTypes
```

**Step 5: When finished with the container object, destroy it****Visual Basic example:**

```
Set NewPixelTypes = Nothing
```

**Delphi example:**

```
ImageLayout := UnAssigned;
```

**PowerBuilder example:**

```
DESTROY NewPixelTypes
```

## 6.5. How To Negotiate the Acquisition Frame Layout

You may use the following TWControl methods to negotiate the acquisition frame layout:

- To get the current and the available dimensional units of measurement (ICAP\_UNITS) use **GetAvailableUnits**. Use **GetCurrentUnits** or **GetDefaultUnits** to get only the current or the default value. Use **SetUnits** to change the current unit.
- To get the current image layout frame use **GetImageLayout**. If successful this method will return a **TWImageLayout** object containing the acquisition frame coordinates, in current units.
- To get the fixed page sizes the Data Source supports use **GetAvailableSupportedSizes**. Use **SetSupportedSizes** to set the acquisition frame to a supported page size.
- To set/change the current acquisition layout use **SetImageLayout** (**ResetImageLayout** will revert the current layout to the Source's default – to get this default do **GetDefaultImageLayout**) and pass the new layout as a **TWImageLayout** object. Like for **TWContainer** when setting a capability, you may use either a **TWImageLayout** object returned by a previous **GetImageLayout** or create a new **TWImageLayout** object.
- The current acquisition frame may also be changed by using **SetIFrames** and a **TWFrame** object as parameter (or a collection of **TWFrame** objects, as a **TWContainer**) - if the Data Source supports setting the related capability (ICAP\_FRAMES).
- To change the orientation or rotate the image to transfer use **SetIOrientation** or **SetIRotation** (do not forget to check first if IC\_ORIENTATION or IC\_ROTATION capabilities are supported and get the supported values with **GetAvailableOrientation** or **GetAvailableIRotation**).

### 6.5.1.1. Visual Basic example

This example will:

1. Set the current unit to inches.
2. Get the current image layout
3. Modify the received **TWImageLayout** object and pass it back to the Data Source
4. Reset the image layout
5. Create a new **TWImageLayout** object and set it to the Data Source

```
On Error GoTo errImageLayout
TWControl.SetIUnits UN_INCHES

Dim layout As TWImageLayout
Set layout = TWControl.GetImageLayout

layout.FrameRight = (layout.FrameRight - layout.FrameLeft) / 2
layout.FrameBottom = (layout.FrameBottom - layout.FrameTop) / 2
layout.FrameLeft = 0
layout.FrameTop = 0
TWControl.SetImageLayout layout
```

*(continued on next page)*

*(continued from previous page)*

```

Set layout = Nothing
TWControl.ResetImageLayout

Dim newLayout As New TWImageLayout
newLayout.FrameLeft = 0
newLayout.FrameRight = 5
newLayout.FrameTop = 0
newLayout.FrameBottom = 5
TWControl.SetImageLayout newLayout

Set newLayout = Nothing
Exit Sub

errImageLayout:
Exit Sub

```

### 6.5.1.2. Delphi 4 example

The example will show how to set an US-Letter scan page size (8.5 x 11 inches):

1. Set ICAP\_UNITS to TWUN\_INCHES.
2. Check if ICAP\_SUPPORTEDSIZES and TWSS\_USLETTER are supported.
3. If the Data Source supports ICAP\_SUPPORTEDSIZES and TWSS\_USLETTER set ICAP\_SUPPORTEDSIZES to TWSS\_USLETTER and terminate.
4. Otherwise try to set the TW\_IMAGE\_LAYOUT frame data (0, 0, 8.5, 11 inches).

```

SourceSupportsInches := True;
try
TWControl.SetIUnits (UN_INCHES);
except
    SourceSupportsInches := False;
end;

if SourceSupportsInches then
begin
    SourceSupportsUSLetter := False;
    NeedToSetUSLetter := True;

    try

        SupportedSizesCon :=
            TWControl.GetAvailableISupportedSizes (SupportedSizesCon);
        for i := 1 to SupportedSizesCon.Count do
        begin
            x := SupportedSizesCon.Item[i].Value;
            if SupportedSizesCon.Item[i].Value = SS_USLETTER then
            begin
                SourceSupportsUSLetter := True;
                if i = SupportedSizesCon.CurrentIndex then
                begin
                    NeedToSetUSLetter := False;
                end;
                break;
            end;
        end;
    end;
end;
end;

```

*(continued on next page)*

*(continued from previous page)*

```

SupportedSizesCon := UnAssigned;

if SourceSupportsUSLetter = True and NeedToSetUSLetter = True then
begin
    TWControl.SetISupportedSizes(SS_USLETTER);
    SourceSupportsUSLetter := True;
End;

except
    SourceSupportsUSLetter := False;
end;
end; //if SourceSupportsInches

if SourceSupportsInches and False = SourceSupportsUSLetter then
begin
    // Try setting the image layout:
    try

        ImageLayout := CoTWImageLayout.Create;
        ImageLayout.FrameLeft := 0;
        ImageLayout.FrameTop := 0;
        ImageLayout.FrameRight := 8.5;
        ImageLayout.FrameBottom := 11;

        TWControl.SetImageLayout(ImageLayout);
        SourceSupportsUSLetter := True;

    except
        SourceSupportsUSLetter := False;
    end;

    ImageLayout := UnAssigned;
end;

```

**6.5.1.3. PowerBuilder example**

```

OLEObject ImageLayout
integer result = 0

this.object.SetIcapUnits(0) /*TWUN_INCHES*/
ImageLayout = CREATE OLEObject
if not IsNull(ImageLayout) then
    result = ImageLayout.ConnectToNewObject("ObjTWAIN.TWImageLayout")
    if 0 > result or IsNull(result) then
        /* insert error handling code here */
    else
        ImageLayout.FrameLeft = 0
        ImageLayout.FrameTop = 0
        ImageLayout.FrameRight = 8.5
        ImageLayout.FrameBottom = 11
        This.object.SetImageLayout(ImageLayout)
    end if
    DESTROY ImageLayout
end if

```

## 6.6. How To Start the Image Acquisition

TWAIN defines three different modes to execute an image transfer:

- Native
- Buffered Memory: strips or tile buffers, uncompressed or compressed
- Disk File

The TWAIN Specification describes the Disk File and Compressed Memory modes as being optional to be implemented by Data Sources and few Data Sources fully supports them. The Native and Uncompressed Memory (especially using strip buffers) modes are marked as mandatory and must be supported correctly by any TWAIN compatible Data Source. The problem when developing a TWAIN application using a high-level programming language like Visual Basic is that under both these modes the application receives the image data either as a DIB handle in memory or a pointer or handle to raw image data (when using the Buffered Memory mode the application will even have to create the image header and assemble the complete image by itself). The ObjectTWAIN™ ActiveX Library solves this by providing **conversion of Native and Buffered Memory TWAIN image transfers to disk file transfers**. Still the ObjectTWAIN™ ActiveX can execute Native and Buffered Memory transfers in the traditional mode (passing to the application DIB handles or a memory buffers) and let the application to handle and assemble the image by itself.

The following table shows the ObjectTWAIN™ Transfer Modes, what TWAIN Transfer Mechanism uses each of them, which TWControl properties shall be set at design or run-time and which TWControl methods must be called to start the acquisition sequence:

ObjectTWAIN™ Xfer Mode	TWAIN Xfer Mode	How to start
<b>Native</b> (DIB handle)	Native (DIB handle)	Set NativeMemoryAsFile to False Call StartNativeAcquisition
<b>Memory</b> (raw memory buffers)	Memory (raw memory buffers)	Set NativeMemoryAsFile to False Call StartMemoryAcquisition
<b>File</b> (disk file)	File (disk file)	Set TransferFile Call StartFileAcquisition
<b>Native to File</b> (disk file)	Native (DIB handle)	Set NativeMemoryAsFile to True Set NativeMemoryFileType Set NativeMemoryTIFFCompression Set TransferFile Call StartNativeAcquisition
<b>Memory to File</b> (disk file)	Memory (raw memory buffers)	Set NativeMemoryAsFile to True Set NativeMemoryFileType Set NativeMemoryTIFFCompression Set TransferFile Call StartMemoryAcquisition

See the next section for information about how to expect and receive the image data.

## 6.7. How To Handle Transferred Image Data and Other Transfer Events

The ObjectTWAIN™ ActiveX Library notifies an application when an image transfer (or just an image buffer for Memory Mode) is received and when a special transfer condition occurs (e.g. all the images were transferred or a transfer error occurred) using TWControl ActiveX events.

The following table enumerates the events an ObjectTWAIN™ application may expect from a TWControl object:

Event / Meaning	Description / Action to do	When to expect
<b>DoProcessNativeData</b>  IMAGE DATA RECEIVED	The application receives a transferred image as a handle to a DIB in memory.  TO DO: save the DIB.	After starting the transfer sequence in <b>Native</b> mode (NativeMemoryAsFile is False)
<b>DoProcessMemoryData</b>  IMAGE DATA RECEIVED	The application receives a raw buffer containing a strip or tile memory buffer of the image.  TO DO: save the buffer; if this is the last buffer finish to assemble one image.	After starting the transfer sequence in <b>Memory</b> mode (NativeMemoryAsFile is False)
<b>DoProcessFileData</b>  IMAGE DATA RECEIVED	The application receives the file format and the file name where a transferred image was just saved.  TO DO: set the TransferFile property for the next transfer or copy the file to other location.	After starting the transfer sequence in <b>File</b> mode.
<b>DoProcessNativeAsFileData</b>  IMAGE DATA RECEIVED	The application receives the file format and the file name where a transferred image was just saved.  TO DO: set the TransferFile property for the next transfer or copy the file to other location.	After starting the transfer sequence in <b>Native to File</b> mode (NativeMemoryAsFile is True).
<b>DoProcessMemoryAsFileData</b>  IMAGE DATA RECEIVED	The application receives the file format and the file name where a transferred image was just saved.  TO DO: set the TransferFile property for the next transfer or copy the file to other location.	After starting the transfer sequence in <b>Memory to File</b> mode (NativeMemoryAsFile is True).

*(continued on next page)*

*(continued from previous page)*

<b>OnFeederLoaded</b>  FEEDER LOADED	The application is notified that the user put paper in the device's Document Feeder.  TO DO: the application may now activate and start the image acquisition.	When a session is opened and <b>AutoCheckFeeder</b> is True.
<b>OnBeginTransfers</b>  TRANSFER(S) READY	The application is notified that the transfers are ready to begin (the Source sent the Transfer Ready signal).	After starting the transfer sequence (of any type). This event shall come only once per acquisition sequence.
<b>OnEndTransfers</b>  NO MORE TRANSFERS	The application is notified that all available images were transferred and the Data Source does not have more pending transfers.  TO DO: close the session.	After starting the transfer sequence (of any type). This event may come only once per acquisition sequence.
<b>OnCancelTransfer</b>  TRANSFER CANCELED	The application is notified that either the Source or the user canceled a transfer and the acquisition sequence.  TO DO: stop the acquisition.	After starting the transfer sequence (of any type). This event or OnEndTransfers may signal that the acquisition is finished.
<b>OnTransferError</b>  TRANSFER ERROR	The application is notified that a transfer error occurred (the application may receive the error code and a error text description).  TO DO: optionally inform the user about the problem and stop the acquisition.	After starting the transfer sequence (of any type).
<b>OnPreTransfer</b>  PRE-TRANSFER	The application is notified that a transfer is about to begin. The TWAIN session is still in state 6.  TO DO: get information about the image to be transferred.	After starting the transfer sequence (of any type). This event shall come once per image transfer in the acquisition sequence.
<b>OnPostTransfer</b>  POST-TRANSFER	The application is notified that a transfer completed. The TWAIN session is in state 6 or 5 (if there are no more transfers pending).  TO DO: -	After starting the transfer sequence (of any type). This event shall come once per image transfer in the acquisition sequence (excepting a canceled or failed transfer).

*(continued on next page)*

*(continued from previous page)*

<b>OnDeviceEvent</b>  DEVICE EVENT	The application is notified that the device triggered an event.  TO DO: read the event information and act accordingly.	After opening a session and (through SetDeviceEvents). This event may come before or after starting the acquisition.
<b>OnUserCloseSourceUI</b>  USER CLOSES SOURCE UI	The application is notified that the user requested to close the Data Source UI. The event parameter indicates if the user closes the DS UI in 'OK' or 'Cancel' mode.  TO DO: stop the acquisition.	After starting the transfer sequence (of any type) with the option to display the Data Source User Interface.

**Note:** Please see next three sections for explanations about getting image information, getting extended image information and how to scan using the device's Document Feeder.

## 6.8. How To Get Image Information

To get information about the image to be transferred the TWControl **GetImageInfo** method may be called while processing an **OnPreTransfer** event. GetImageInfo returns if successful a **TWImageInfo** object that is a COM wrapper for the TW\_IMAGEINFO data structure defined by TWAIN. You can retrieve in this way the image dimensions, pixel type, compression, bit depth per sample and number of samples (color channels) for the image next to transfer.

Visual basic example:

```
Private Sub TWControl_OnPreTransfer()
    On Error GoTo errGetImageInfo
    Dim ImageInfo As TWImageInfo

    Set ImageInfo = TWControl.GetImageInfo
    ` do something:
    If ImageInfo.PixelType = PT_BW Then
        MsgBox "The next image to be transferred will be B&W"
    End If
    Exit Sub

errGetImageInfo:
    MsgBox "GetImageInfo failed"
End Sub
```

If the application may call **StopAcquisition** or **CloseSession** while processing a **OnPreTransfer** event (if the application does not know how (or does not want to) handle a certain type of image data - e.g. if one combination of pixel type and bit depth are not supported by the application) when processing **OnTransferError** the application should check before reporting a transfer error that the ObjectTWAIN™ session state is ACQUISITION:

Visual basic example:

```
Private Sub TWControl_OnTransferError(ByVal error As
OBJTWAINLibCtl.tagObjectTwainError, ByVal ErrorDescription As String)
    If TWControl.GetState = ACQUISITION Then
        MsgBox ErrorDescription
        TWControl.StopAcquisition
    End If
End Sub
```

Under the following conditions **GetImageInfo** may also be called while processing one of the **OnProcessXData** events (when the TWAIN session is in state 7): (1) if the Data Source is unable to detect the image dimensions before the transfer (e.g. a Data Source deserving a hand held scanner), (2) if the ICAP\_UNEDFINEDIMAGESIZE is True (use GetUndefindImageSize), (3) if the image width and length are returned to be -1 pixels in the TWImageInfo object (the one get at OnPreTransfer) and (4) if the Data Source supports returning image information after the transfer (in the TWAIN session state 7) .

## 6.9. How To Get Extended Image Information

To get extended image information (such as barcode information) follow the next steps:

**Step 1:** Check if the Data Source supports this kind of extended information by calling **GetIExtendedImageInfo** and other capability methods (e.g. **GetIBarCodeDetectionEnabled** if the extended information to get is related to barcodes).

Visual Basic example (from *GenProd* sample, TWControl\_DoProcessMemoryAsFileData):

```
bBarCodeDetectionEnabled = False
iMaxBarCodeSearch = 0

bBarCodeDetectionEnabled = TWControl.GetIBarCodeDetectionEnabled
AppendLog "GetIBarCodeDetectionEnabled ok"

If bBarCodeDetectionEnabled Then
    iMaxBarCodeSearch = TWControl.GetCurrentIBarCodeMaxSearchPriorities
    AppendLog "GetCurrentIBarCodeMaxSearchPriorities ok"
    If iMaxBarCodeSearch <= 0 Then
        bBarCodeDetectionEnabled = False
    End If
End If

If False = bBarCodeDetectionEnabled Then
    AppendLog "Barcode detection disabled"
    Exit Sub
End If

If False = TWControl.GetIExtImageInfo Then
    AppendLog "GetIExtImageInfo ok, returned False"
    AppendLog "No GetExtImageInfo will be requested"
    Exit Sub
Else
    AppendLog "GetIcapExtImageInfo ok"
End If
```

**Step 2:** Construct a **TWContainer** object to contain the array of extended image information objects to ask the Data Source to return data into. Initialize the TWContainer object as an **array** specifying the desired number of extended infos as the number of array items.

Visual Basic example (from *GenProd* sample, TWControl\_DoProcessMemoryAsFileData):

```
If iMaxBarCodeSearch > 6 Then
    iMaxBarCodeSearch = 6
End If
' BARCODETEXT & BARCODETYPE for each barcode, & 1 BARCODECOUNT:
iInfos = ((iMaxBarCodeSearch * 2) + 1)

Set ExtImageInfo = New TWContainer
ExtImageInfo.InitArray iInfos
```

**Step 3:** Construct the extended information objects as **TWInfo** objects. Initialize for each TWInfo object the **InfoID** property. Add the TWInfo objects to the TWContainer array.

Visual Basic example (from *GenProd* sample, TWControl\_DoProcessMemoryAsFileData):

```

For i = 1 To iInfos
    Set Info(i) = New TWInfo
Next

Info(1).InfoID = EI_BARCODECOUNT
For i = 2 To iInfos
    Info(i).InfoID = EI_BARCODETYPE
    i = i + 1
    Info(i).InfoID = EI_BARCODETEXT
Next

For i = 1 To iInfos
    ExtImageInfo.Add Info(i)
Next

```

**Step 4:** Call the TWControl **GetExtImageInfo** method. If successful, the TWInfo objects in the TWContainer array shall contain the extended image information returned by the Data Source.

Visual Basic example (from *GenProd* sample, TWControl\_DoProcessMemoryAsFileData):

```

TWControl.GetExtImageInfo ExtImageInfo
AppendLog "GetExtImageInfo ok"

If ExtImageInfo.Count > 0 Then
    For Each Temp In ExtImageInfo
        If Temp.InfoID = EI_BARCODECOUNT Then
            For Each InfoItem In Temp
                AppendLog "BarCodeCount: " & CStr(InfoItem)
            Next
        End If

        If Temp.InfoID = EI_BARCODETYPE Then
            For Each InfoItem In Temp
                AppendLog "BarCodeType: " & DecodeBarcode(Int(InfoItem))
            Next
        End If

        If Temp.InfoID = EI_BARCODETEXT Then
            For Each InfoItem In Temp
                AppendLog "BarCodeText: " & CStr(InfoItem)
            Next
        End If
    Next 'For Each Info In ExtImageInfo
End If

```

**Step 5:** After finishing with the array of TWInfo objects, destroy them and free the allocated memory.

Visual Basic example (from *GenProd* sample, TWControl\_DoProcessMemoryAsFileData):

```
For i = 1 To iInfos
    Set Info(i) = Nothing
Next
Set ExtImageInfo = Nothing
```

**Note:** For the above sequence of code (from the *GenProd* sample) the local variable definitions and the error handling mechanism are listed next:

```
Dim bBarcodeDetectionEnabled As Boolean
Dim iMaxBarcodeSearch As Integer
Dim iInfos As Integer
Dim ExtImageInfo As TWContainer
Dim Info(1 To 14) As TWInfo
Dim Temp As TWInfo
Dim InfoItem As Variant
Dim i As Integer
```

```
On Error GoTo errGetExtImageInfo
    ' follows the code sequence to get extended image information
    Exit Sub
errGetExtImageInfo:
    AppendLog "Get extended image info failed:"
    AppendLog Err.Description
```

## 6.10. How To Use the Automatic Document Feeder

To 'manually' check the feeder status, enable or disable the feeder, use the following TWControl methods:

- To see if the feeder is enabled call **GetFeederEnabled**.
- To enable or disable the feeder call **SetFeederEnabled**.
- To see if the feeder is loaded call **GetFeederLoaded**.
- To see if the Automatic Document Feeder is enabled call **GetAutoFeed**.
- To enable or disable the Automatic Document Feeder (if present) call **SetAutoFeed**.

To use the automatic feeder loaded detect feature supported by ObjectTWAIN™ set the TWControl **AutoCheckFeeder** property to True (either at design or at run-time). When AutoCheckFeeder is enabled, as soon as a session is opened ObjectTWAIN™ will begin to automatically call GetFeederLoaded. Setting the TWControl property **ACFInterval** configures the time between two automatic GetFeederLoaded calls. When the feeder will be loaded with paper TWControl will send an **OnFeederLoaded** event. The application may open a session and then wait in background until it receives OnFeederLoaded, then activate and start the acquisition.

Delphi 3 example:

```

procedure TScanForm.TWControlFeederLoaded(Sender: TObject);
var
  TWError : tagObjectTwainError;
begin
  if (TWControl.GetState <> ACQUISITION) and
    (TWControl.AutoCheckFeeder) then
  begin
    TWControl.StartNativeAcquisition(False);
    TWError := TWControl.GetLastError;
    if TWError <> OBJTWAIN_NO_ERROR then
    begin
      Application.MessageBox('Unable to start acquisition.',
        'Error', MB_ICONSTOP);
      CloseSession;
      TWControl.AutoCheckFeeder := False;
    end;
  end;
end;

```

## 7. Library Reference

### R.1. ObjectTWAIN™ ActiveX/COM Controls

Control name	Interfaces	Description
<b>TWControl</b>	ITWControl Idispatch	TWControl is the main ActiveX control, insertable and invisible at run-time, exported by the ObjectTWAIN™ ActiveX Library. By using this control an application can select a TWAIN Data Source, open and control programatically a complete TWAIN image acquisition session.
<b>TWContainer</b>	ITWContainer Idispatch	Implements an array or enumeration of values or COM objects or a range of values as a COM Collection. Wrapper for TWAIN capability containers and for the array of extended image attributes.
<b>TWSourceList</b>	ITWSourceList Idispatch	Implements as a COM Collection a list of TWSourceInfo objects to hold information about the TWAIN Data Sources installed on system (get with GetInstalledSources).
<b>TWSourceInfo</b>	ITWSourceInfo Idispatch	Wrapper COM object for TWAIN Data Source identity data structure. Describes the attributes of a Data Source.
<b>TWImageLayout</b>	ITWImageLayout Idispatch	Wrapper COM object for TWAIN image layout data structure. Used to negotiate the acquisition frame layout.
<b>TWImageInfo</b>	ITWImageInfo Idispatch	Wrapper COM object for the TWAIN image info data structure. Describes the attributes of the next image to transfer or the last transferred image.
<b>TWInfo</b>	ITWInfo Idispatch	Wrapper COM object for the TWAIN extended information/attribute data structure. May be stored to TWContainer to create an array of infos to request extended image information from the Data Source.
<b>TWFrame</b>	ITWFrame Idispatch	Wrapper COM object for the TWAIN frame data structure. May be stored to TWContainer to create an enumeration or array of frames.
<b>TWCapQuery</b>	ITWCapQuery Idispatch	Wrapper COM object for the TWAIN query capability support data structure. Used with QueryCapability.

## R.2. Other ObjectTWAIN™ COM Controls

The ObjectTWAIN™ ActiveX Library exports wrapper COM objects for all enumerated capability values defined by TWAIN Release v1.8. Each of these objects has as a single default property (named **Value**), of a type of the related ObjectTWAIN™ COM enumeration which defines the related capability values (e.g. there is a TWPixelFormatValue wrapper COM object for the defined ICAP\_PIXELTYPE capability values, TWPixelFormatValue.Value is of type PixelType, PixelType is defined as a COM Enumeration containing aliases for the TWPT\_x values defined by TWAIN).

These objects can be very useful in Visual Basic and other development environments which may be able to display in the editor a selection with the available TWAIN defined values when typing a TWControl capability method.

---

**IMPORTANT:** These objects may be replaced in almost all cases by simple numerical values (as defined by TWAIN) when calling a TWControl method to execute a capability operation. TWControl is able to handle an enumerated capability value passed as a parameter to a capability method as either a wrapper COM object or simple numerical value. If the development environment supports this, an enumerated capability value returned by ObjectTWAIN™ as a COM object will be converted automatically by the development environment to a simple value.

In Visual Basic, to request set the current value of ICAP\_PIXELTYPE to TWPT\_BW, you may write:

```
TWControl.SetIPixelType PT_BW
` or:
TWControl.SetIPixelType 0
`or:
Dim BW as new TWPixelFormatValue
` Here VB may be able to display automatically while typing
` a selection box with the available PT values so you will
` not have to look in documentation for the PT_BW value:
BW = PT_BW
TWControl.SetIPixelType BW
Set BW = nothing
```

When typing code in Visual Basic to compare a received ICAP\_PIXELTYPE value with a constant value, Visual Basic may display a selection list with the available values to select from and will let you select a value or type your own:

```
Dim PixelTypes As TWContainer
Dim pt As TWPixelFormatValue
Set PixelTypes = TWControl.GetIAvailablePixelFormatType
For Each pt In PixelTypes
    ` Here VB may be able to display automatically while typing
    ` a selection box with the available PT values so you will
    ` not have to look in documentation to find the PT_BW value:
    If pt = PW_BW Then
        `...
    End If
Next
```

---

In the following table are listed all the ObjectTWAIN™ COM wrapper objects for enumerated-type TWAIN capability values. The third column displays the ObjectTWAIN™ prefix (e.g. PT for PT\_BW) and the TWAIN prefix (e.g. TWPT for TWPT\_BW; PT\_BW is an alias for TWPT\_BW, their values are identical, 0). The values related to a single capability all have the same prefix.

COM Object Name	TWAIN Capability	Value Prefix
<b>TWCompressionValue</b>	ICAP_COMPRESSION	CMP/TWCP
<b>TWPixelFormatValue</b>	ICAP_PIXELTYPE	PT/TWPT
<b>TWUnitsValue</b>	ICAP_UNITS	UN/TWUN
<b>TWXferMechValue</b>	ICAP_XFERMECH ACAP_XFERMECH	SX/TWSX
<b>TWCapabilityName</b>	CAP_SUPPORTEDCAPS CAP_EXTENDED CAPS	C/CAP IC/ICAP AC/ACAP
<b>TWDuplexValue</b>	CAP_DUPLEX	DX/TWDX
<b>TWJobControlValue</b>	CAP_JOBCONTROL	JC/TWJC
<b>TWFilterValue</b>	ICAP_FILTER	FT/TWFT
<b>TWImageFileFormatValue</b>	ICAP_IMAGEFILTER	IF/TWIF
<b>TWLightSourceValue</b>	ICAP_LIGHTSOURCE	LS/TWLS
<b>TWOrientationValue</b>	ICAP_ORIENTATION	OR/TWOR
<b>TWBitOrderValue</b>	ICAP_BITORDER	BO/TWBO
<b>TWLightPathValue</b>	ICAP_LIGHTPATH	LP/TWLP
<b>TWPixelFlavorValue</b>	ICAP_PIXELFLAVOR ICAP_PIXELFLAVORCODES	PF/TWPF
<b>TWPlanarChunkyValue</b>	ICAP_PLANARCHUNKY	PC/TWPC
<b>TWSupportedSizesValue</b>	ICAP_SUPPORTEDSIZES	SS/TWSS
<b>TWBitDepthReductionValue</b>	ICAP_BITDEPTHREDUCTION	BR/TWBR
<b>TWAlarmsValue</b>	CAP_ALARMS	AL/TWAL
<b>TWDeviceEventsValue</b>	CAP_DEVICEEVENT	DE/TWDE
<b>TWClearBuffersValue</b>	CAP_CLEARBUFFERS	CB/TWCB
<b>TWAudioFileFormatValue</b>	ACAP_AUDIOFILEFORMAT	AF/TWAF
<b>TWFeederAlignmentValue</b>	CAP_FEEDERALIGNMENT	FA/TWFA
<b>TWFeederOrderValue</b>	CAP_FEEDERORDER	FO/TWFO
<b>TWLanguageValue</b>	CAP_LANGUAGE	LG/TWLG
<b>TWPowerSupplyValue</b>	CAP_POWERSUPPLY	PS/TWPS
<b>TWPrinterValue</b>	CAP_PRINTER	PR/TWPR
<b>TWPrinterModeValue</b>	CAP_PRINTERMODE	PM/TWPM
<b>TWBarcodeSearchModeValue</b>	ICAP_BARCODESEARCHMODE	BC/TWBD
<b>TWPatchCodeSearchModeValue</b>	ICAP_PATCHCODESEARCHMODE	PC/TWBD
<b>TWFlashUsed2Value</b>	ICAP_FLASHUSED2	FL/TWFL
<b>TWFlipRotationValue</b>	ICAP_FLIPROTATION	FR/TWFR
<b>TWImageFilterValue</b>	ICAP_IMAGEFILTER	IF/TWIF
<b>TWNoiseFilterValue</b>	ICAP_NOISEFILTER	NF/TWNF
<b>TWOverScanValue</b>	ICAP_OVERSCAN	OV/TWOV
<b>TWBarcodeTypeValue</b>	ICAP_SUPPORTEDBARCODETYPES ICAP_BARCODESEARCHPRIORITIES	BT/TWBT
<b>TWDeskewStatusValue</b>	ICAP_DESKEWSTATUS	DSK/TWDSK
<b>TWPatchCodeValue</b>	ICAP_SUPPORTEDPATCHCODETYPES ICAP_PATCHCODESEARCHPRIORITIES	PCH/TWPCH

### R.3. TWControl ActiveX Control Properties

**Note:** All TWControl properties are accessible both at design and at a run-time.

Property Name and Type	Allowed Values	Description
<b>Quiet</b> (Boolean)	True False	When enabled (set to True) ObjectTWAIN™ will throw an OLE Exception at every error encountered.
<b>TransferFile</b> (Character String)	Any valid file name, up to 255 characters in length.	The file to save the transferred image to (for disk file transfer modes).
<b>NativeMemoryTIFFCompression</b> (enum. TIFFCompression)	TIFF_NONE = 0 TIFF_GROUP31D = 2 TIFF_GROUP32D = 4 TIFF_GROUP4 = 5 TIFF_JPEG = 6	The compression mode to save a Native or Memory transferred image to a disk file (when NativeMemoryAsFile is enabled).  <b>IMPORTANT:</b> TIFF_JPEG is allowed only for 8 bpp Grayscale and 24 bpp RGB data; TIFF_GROUPx are allowed only for 1 bit per pixel B&W data.
<b>NativeMemoryAsFile</b> (Boolean)	True False	When enabled (set to True) ObjectTWAIN™ will save to disk the images transferred through Native and Memory modes.
<b>NativeMemoryFileType</b> (enum. NativememoryFileType)	DIB = 2 JPEG = 4 TIFF = 0 MULTIPAGE_TIFF = 6	The format of the file to save a Native or Memory transferred image to disk (when NativeMemoryAsFile is enabled).  <b>IMPORTANT:</b> JPEG is allowed only for 8 bpp Grayscale and 24 bpp RGB data.
<b>AppProductName</b> (Character String)	Any string up to 32 characters in length.	The product name to be listed in the application's TWAIN identity data structure.
<b>AppProductFamily</b> (Character String)	Any string up to 32 characters in length.	The product family to be listed in the application's TWAIN identity data structure.
<b>AppManufacturer</b> (Character String)	Any string up to 32 characters in length.	The manufacturer name to be listed in the application's TWAIN identity data structure.
<b>AppVerMajor</b> (short integer, 16 bits)	Any number between 0 and 32767.	The application's major version number to be listed in the application's TWAIN identity data structure.

*(continued on next page)*

*(continued from previous page)*

<b>AppVerMinor</b> (short integer, 16 bits)	Any number between 0 and 32767.	The application's minor version number to be listed in the application's TWAIN identity data structure.
<b>AppLanguage</b> (enum. Language)	Any LG_x value defined by ObjectTWAIN™ or any TWLG_x value defined by TWAIN.	The application's language identifier to be listed in the application's TWAIN identity data structure.
<b>AppCountry</b> (enum. Country)	Any CY_x value defined by ObjectTWAIN™ or any TWCY_x value defined by TWAIN.	The application's country identifier to be listed in the application's TWAIN identity data structure.
<b>AutoCheckFeeder</b> (boolean)	True False	When enabled (set to True) and a TWAIN session is opened ObjectTWAIN™ will call automatically GetFeederLoaded and will trigger an OnFeederLoaded event when the feeder will be loaded.
<b>ACFInterval</b> (long integer, 32 bits)	Any number between 1 and 999 (sec, 16.65 min)	The time in seconds between two GetFeederLoaded automatic calls (when AutoCheckFeeder is enabled).

## R.4. TWControl ActiveX Control Basic Methods

This section describes the basic methods (i.e. the main methods the application may use to control a TWAIN session) exposed by TWControl.

### IMPORTANT:

All **object** parameters or return values for TWControl methods are passed/returned to/by ObjectTWAIN™ as **VARIANT** variables (ANY for PowerBuilder) containing pointers to IDispatch interfaces of the referred COM objects.

When ObjectTWAIN™ returns an object the application must destroy the object after finishing with it to free the allocated memory and other resources. When an ObjectTWAIN™ method expects a reference to an object as a parameter, the application must construct and initialize the object before to call the method and destroy the object after it finishes working with it.

All values or objects described to be returned by TWControl methods are returned only when the related methods are completed successfully.

The methods are listed below along with short descriptions for parameter(s), return values (if any) and the TWAIN operation triplets called:

#### About

Displays the About dialog box for ObjTWAIN.ocx.

#### UserSelectSource

Calls DG\_CONTROL/ DAT\_IDENTITY/ MSG\_USERSELECT to display the DSM Source Select dialog and select the default TWAIN Data Source.

#### GetInstalledSources

Returns a **TWSourceList** object containing an array of TWSourceInfo objects for the Data Sources installed on the system. ObjectTWAIN™ retrieves this list at TWControl initialization time by calling DG\_CONTROL/ DAT\_IDENTITY/ MSG\_GETFIRST, MSG\_GETNEXT.

#### GetSourceInfo

Parameter **DataSource**: string parameter to specify the Source ProductName to return information for (optional). If no Source name is specified as parameter, the information will be returned for the default Data Source.

Returns a **TWSourceInfo** object.

#### GetState

Returns an **ObjectTwainState** enumeration value describing the current ObjectTWAIN™ state: NOT\_STARTED (0), STARTED (1), OPENED (2) or ACQUISITION (3).

**OpenSession**

Opens the TWAIN session (loads the DSM, calls DG\_CONTROL/ DAT\_PARENT/ MSG\_OPENDSM, DG\_CONTROL/ DAT\_IDENTITY/ MSG\_OPENDS). This method may be called in any state (it first closes the current session if any). However, it is recommended to call it only in state STARTED. If successful it moves the ObjectTWAIN™ session to state OPENED.

Parameter **hWndParent**: handle to a valid application's window (HWND specified as long integer) to be used when opening the DSM.

Parameter **DataSource**: string parameter to specify the Source ProductName to open (optional). If no Source name is specified as parameter the default Data Source will be opened.

**GetCapability**

Calls DG\_CONTROL/ DAT\_CAPABILITY/ MSG\_GET on the specified capability (TWAIN defined or custom capability). This method is allowed only in the state OPENED and state ACQUISITION.

Parameter **Cap**: the capability identifier as a CapabilityName enumerated value or a numerical value.

Returns a **TWContainer** object containing the available, current and default capability values reported by the Data Source (OneValue containers are converted by ObjectTWAIN™ to arrays with one single item).

**GetCurrentCapability**

Calls DG\_CONTROL/ DAT\_CAPABILITY/ MSG\_GETCURRENT on the specified capability (TWAIN defined or custom capability). This method is allowed only in the state OPENED and state ACQUISITION.

Parameter **Cap**: the capability identifier as a CapabilityName enumerated value or a numerical value.

Returns a **TWContainer** object containing the current capability value(s) reported by the Data Source (OneValue containers are converted by ObjectTWAIN™ to arrays with one single item).

**GetDefaultCapability**

Calls DG\_CONTROL/ DAT\_CAPABILITY/ MSG\_GETDEFAULT on the specified capability (TWAIN defined or custom capability). This method is allowed only in the state OPENED and state ACQUISITION.

Parameter **Cap**: the capability identifier as a CapabilityName enumerated value or a numerical value.

Returns a **TWContainer** object containing the default capability value(s) reported by the Data Source (OneValue containers are converted by ObjectTWAIN™ to arrays with one single item).

### ResetCapability

Calls DG\_CONTROL/ DAT\_CAPABILITY/ MSG\_RESET on the specified capability (TWAIN defined or custom capability). This method is allowed only in the state OPENED and state ACQUISITION.

Parameter **Cap**: the capability identifier as a CapabilityName enumerated value or a numerical value.

Returns a **TWContainer** object containing the default capability value(s) reported by the Data Source (OneValue containers are converted by ObjectTWAIN™ to arrays with one single item).

### SetCapability

Calls DG\_CONTROL/ DAT\_CAPABILITY/ MSG\_SET on the specified capability (TWAIN defined or custom capability). It is recommended to call this method only in the state OPENED. The method may succeed in state ACQUISITION (in TWAIN states 5 and 6, while processing OnPreTransfer and OnPostTransfer events) on capabilities previously negotiated as extended (through GetExtendedCaps and/or SetExtendedCaps).

Parameter **Cap**: the capability identifier as a CapabilityName enumerated value or a numerical value.

Parameter **NewVal**: VARIANT parameter that may contain a simple value, an enumerated type value, a TWFrame object (these will be converted by ObjectTWAIN™ to TWON\_ONEVALUE TWAIN containers), a TWContainer object containing an array, enumeration or range of simple type values (e.g. boolean, string), enumerated type objects (e.g. TWPixelFormatValue) and TWFrame objects (a TWContainer object will be converted here by ObjectTWAIN™ to either a TWON\_ARRAY, TWON\_ENUMERATION or TWON\_RANGE TWAIN container). Note that every capability may accept only certain types of containers for MSG\_SET.

Parameter **ValType**: parameter of type ItemType (enumeration containing aliases for all TWAIN defined TWTY values; e.g. TY\_UINT16 is the same as TWTY\_UINT16). The type of value passed must be appropriate for the capability to set (e.g. do not set a TWFrame object to a capability defined to accept integer values). ObjectTWAIN™ will convert the NewVal value(s) to the TWAIN type specified by the ItemType parameter before executing the DG\_CONTROL/ DAT\_CAPABILITY/ MSG\_SET operation. For Set on frame (TWFrame) and enumerated type capabilities (when used with enumeration objects, not when used with simple integer values) the item type must be correctly specified (e.g. TY\_UINT16 or TY\_FRAME). For other types any combination is allowed (e.g. ObjectTWAIN™ may convert a character string to the appropriate floating-point value if the requested item type is TY\_FIX32).

### QueryCapability

Calls DG\_CONTROL/ DAT\_CAPABILITY/ MSG\_QUERY SUPPORT on the specified capability (TWAIN defined or custom capability). This method is allowed only in the state OPENED and state ACQUISITION.

Parameter **Cap**: the capability identifier as a CapabilityName enumerated value or a numerical value.

Returns a **TWCapQuery** object describing the operations reported as supported by the data Source on the specified capability.

### GetImageLayout

Calls DG\_IMAGE/ DAT\_IMAGELAYOUT/ MSG\_GET. This method is allowed only in the state OPENED.

Returns a **TWImageLayout** object containing the current Data Source layout frame coordinates, the document, page and frame numbers.

### GetDefaultImageLayout

Calls DG\_IMAGE/ DAT\_IMAGELAYOUT/ MSG\_GETDEFAULT. This method is allowed only in the state OPENED.

Returns a **TWImageLayout** object containing the default Data Source layout frame coordinates the document, page and frame numbers.

### ResetImageLayout

Calls DG\_IMAGE/ DAT\_IMAGELAYOUT/ MSG\_RESET. This method is allowed only in the state OPENED.

Resets the current Data Source image layout to default and returns this default as a **TWImageLayout** object.

### SetImageLayout

Calls DG\_IMAGE/ DAT\_IMAGELAYOUT/ MSG\_SET. This method is allowed only in the state OPENED.

Parameter **newVal**: a **TWImageLayout** object containing the new layout frame coordinates the document, page and frame numbers to set.

### StartNativeAcquisition

Calls several TWAIN operation triplets to enable the Data Source (DG\_CONTROL/ DAT\_USERINTERFACE/ MSG\_ENABLEDS), wait for MSG\_XFERREADY and transfer the available images through the TWAIN Native Transfer mechanism (DG\_IMAGE/ DAT\_IMAGENATIVEXFER/ MSG\_GET, DG\_CONTROL/ DAT\_PENDINGXFERS/ MSG\_ENDXFER). This method is allowed only in state OPENED. If successful moves the session in state ACQUISITION.

Parameter **ShowSourceUI**: boolean parameter indicating how the Data Source shall be enabled, with its UI displayed (ShowSourceUI set to True) or without to display its UI (False).

### StartFileAcquisition

Calls several TWAIN operation triplets to enable the Data Source (DG\_CONTROL/ DAT\_USERINTERFACE/ MSG\_ENABLEDS), wait for MSG\_XFERREADY and transfer the available images through the TWAIN File Transfer mechanism (DG\_IMAGE/ DAT\_IMAGEFILEXFER/ MSG\_GET, DG\_CONTROL/ DAT\_PENDINGXFERS/ MSG\_ENDXFER). This method is allowed only in state OPENED. If successful moves the session in state ACQUISITION.

Parameter **ShowSourceUI**: boolean parameter indicating how the Data Source shall be enabled, with its UI displayed (ShowSourceUI set to True) or without to display its UI (False).

Parameter **FileFormat**: ImageFileFormat enumeration value (alias for TWAIN TWFF values, default is FF\_BMP) to specify the Transfer File Format. The transfer file name is specified through the **TransferFile** TWControl property that can be changed also in state ACQUISITION, between transfers.

### StartMemoryAcquisition

Calls several TWAIN operation triplets to enable the Data Source (DG\_CONTROL/ DAT\_USERINTERFACE/ MSG\_ENABLEDS), wait for MSG\_XFERREADY and transfer the available images through the TWAIN Memory Transfer mechanism (DG\_IMAGE/ DAT\_IMAGEMEMXFER/ MSG\_GET, DG\_CONTROL/ DAT\_PENDINGXFERS/ MSG\_ENDXFER). This method is allowed only in state OPENED. If successful moves the session in state ACQUISITION.

Parameter **ShowSourceUI**: boolean parameter indicating how shall be the Data Source enabled, with its UI displayed (ShowSourceUI set to True) or without to display its UI (False).

### StopAcquisition

May call several TWAIN operation triplets to stop the acquisition and disable the Data Source (DG\_CONTROL/ DAT\_PENDINGXFERS/ MSG\_ENDXFER, DG\_CONTROL/ DAT\_PENDINGXFERS/ MSG\_RESET, DG\_CONTROL/ DAT\_USERINTERFACE/ MSG\_DISABLED). This method is allowed only in state ACQUISITION. If successful moves the session in state OPENED.

### CloseSession

Closes the TWAIN session (executes StopAcquisition if needed, calls DG\_CONTROL/ DAT\_IDENTITY/ MSG\_CLOSED, DG\_CONTROL/ DAT\_PARENT/ MSG\_CLOSED, unloads the DSM). This method may be called in any state (it is functional in state OPENED and state ACQUISITION).

### GetLastError

Returns the result of the last TWControl operation as an **ObjectTwainError** enumeration value. OBJTWAIN\_NO\_ERROR (defined as 0x0200) means successful operation (no error). Any other code signals an error.

**GetImageInfo**

Calls DG\_IMAGE/ DAT\_IMAGEINFO/ MSG\_GET. This method is allowed only in the state ACQUISITION.

Returns a **TWImageInfo** object containing information about the next image to transfer (if called in the TWAIN state 6, while processing the OnPreTransfer event) or about the image just transferred (when called in state 7, while processing a DoProcessXData event).

**GetSourceCustomData**

Calls DG\_CONTROL/ DAT\_CUSTOMDSDATA/ MSG\_GET. This method is allowed only in the state OPENED.

Parameter **FileName**: specifies the file where to save the received data from the Source.

**SetSourceCustomData**

Calls DG\_CONTROL/ DAT\_CUSTOMDSDATA/ MSG\_SET. This method is allowed only in the state OPENED.

Parameter **FileName**: specifies the file containing the data to set to the Source.

**EnableSourceUIOnly**

Calls DG\_CONTROL/ DAT\_USERINTERFACE/ MSG\_ENABLEDSUIONLY. This method is allowed only in the state OPENED. If successful, the application must wait for **OnUserCloseSourceUI** before executing any other call allowed only in state OPENED.

**GetExtImageInfo**

Calls DG\_IMAGE/ DAT\_EXTIMAGEINFO/ MSG\_GET. This method is allowed only in the state ACQUISITION. If successful writes in the specified array of TWInfo objects the extended information received from the Data Source.

Parameter **InfoListCon**: **TWContainer** object containing the array of **TWInfo** objects related to the extended image information attributes to ask the Data Source to return. Each TWInfo object in array must specify through the **InfoID** property a valid extended information attribute.

**ChangeCamera**

Executes DG\_CONTROL/ DAT\_FILESYSTEM/ MSG\_CHANGEDIRECTORY to change the current camera device context to the one described by the path name passed as parameter. This operation could be useful when using a TWAIN Data Source supporting more than one acquisition camera and also separate negotiation of certain TWAIN capabilities for each camera. For example a duplex scanner might allow the TWAIN application to change the camera context using this DAT\_FILESYSTEM operation to either the front or back side scan cameras and then negotiate capabilities for that scan side (camera context).

Parameter **FileName**: Absolute camera path name as described by your TWAIN Data Source OEM documentation (if multiple camera device contexts and DAT\_FILESYSTEM operations).

**GetSoftwareJPEGQuality**

Returns the current JPEG compression quality factor (0..100, smaller factor meaning less compression and higher image quality, higher factor meaning more aggressive compression and lower image quality) that ObjectTWAIN will use for upcoming Native or memory To File transfers (NativeMemoryAsFile property set to True) when the NativeMemoryFileType property is set to JPEG.

**Note:** This quality factor is used only when the JPEG compression is applied through Software by ObjectTWAIN itself, is not used in cases where the image is transferred JPEG compressed from the TWAIN Data Source (to configure the JPEG Quality Factor in that case please negotiate the ICAP\_JPEGQUALITY capability, if supported – see note below).

**Note about ICAP\_JPEGQUALITY:** This version of ObjectTWAIN does not have dedicated TWControl methods to negotiate ICAP\_JPEGQUALITY. Please use the general capability negotiation methods for this purpose. See TWAIN.H for the ICAP\_JPEGQUALITY capability ID and the TWAIN Release v1.9 Specification (both available from <http://www.twain.org>) for the description of this capability.

**SetSoftwareJPEGQuality**

Changes the JPEG compression quality factor (0..100, smaller factor meaning less compression and higher image quality, higher factor meaning more aggressive compression and lower image quality) that ObjectTWAIN will use in the current session for upcoming Native or memory To File transfers (NativeMemoryAsFile property set to True) when the NativeMemoryFileType property is set to JPEG. See above notes about this method's scope and ICAP\_JPEGQUALITY.

Parameter **newVal**: short integer (0..100) describing new JPEG quality factor.

## R.5. TWControl ActiveX Control Defined Capability Methods

TWControl provides a special set of methods to execute TWAIN operations on the defined capabilities (for custom defined capabilities use the TWControl basic capability methods: GetCapability, GetCurrentCapability, GetDefaultCapability, ResetCapability and SetCapability). The main advantage of these methods over the basic capability methods is simpler argument handling (e.g. the item type shall be not specified for a Set operation, the parameter types are already defined accordingly with the related capability allowed value types).

The following table displays the naming convention used for these capability methods:

<b>TWAIN Capability</b>	<b>DAT_CAPABILITY Operation</b>	<b>TWControl Method Name</b>
<i>CAP_CAPNAME</i>	MSG_GET returning only one value MSG_GET returning multiple values MSG_GETCURRENT MSG_GETDEFAULT MSG_RESET MSG_SET	<b>GetCapName()</b> <b>GetAvailableCapName ()</b> <b>GetCurrentCapName ()</b> <b>GetDefaultCapName ()</b> <b>ResetCapName ()</b> <b>SetCapName ()</b>
<i>ACAP_CAPNAME</i>	MSG_GET returning only one value MSG_GET returning multiple values MSG_GETCURRENT MSG_GETDEFAULT MSG_RESET MSG_SET	<b>GetACapName ()</b> <b>GetAvailableACapName ()</b> <b>GetCurrentACapName ()</b> <b>GetDefaultACapName ()</b> <b>ResetACapName ()</b> <b>SetACapName ()</b>
<i>ICAP_CAPNAME</i>	MSG_GET returning only one value MSG_GET returning multiple values MSG_GETCURRENT MSG_GETDEFAULT MSG_RESET MSG_SET	<b>GetICapName ()</b> <b>GetAvailableICapName ()</b> <b>GetCurrentICapName ()</b> <b>GetDefaultICapName ()</b> <b>ResetICapName ()</b> <b>SetICapName ()</b>

All methods for TWAIN defined capabilities are allowed in ObjectTWAIN™ states OPENED and ACQUISITION. All Reset and Set methods shall be called only in state OPENED unless the related capabilities were previously negotiated as extended (use GetExtendedCaps and SetExtendedCaps to negotiate the extended capabilities).

The following table enumerates the TWControl methods for operations on TWAIN defined capabilities. A VARIANT type parameter or return value either indicates a TWContainer object or a simple value, enumeration, TWFrame or COM-wrapper for an ObjectTWAIN™ enumeration.

<b>Method</b>	<b>Parameter type(s)</b>	<b>Return value type</b>
GetAvailableAAudioFileFormat		VARIANT: TWContainer TWAudioFileFormatValue enumeration.
GetCurrentAAudioFileFormat		AudioFileFormat
GetDefaultAAudioFileFormat		AudioFileFormat
ResetAAudioFileFormat		AudioFileFormat
SetAAudioFileFormat	VARIANT: single value (AudioFileFormat, integer or TWAudioFileFormatValue) or TWContainer enumeration.	

*(continued on next page)*

(continued from previous page)

GetAvailableAXferMech		VARIANT: TWContainer TWXferMechValue enumeration.
GetCurrentAXferMech		XferMech
GetDefaultAXferMech		XferMech
ResetAXferMech		XferMech
SetAXferMech	VARIANT: single value (XferMech, integer or TWXferMechValue) or TWContainer enumeration.	
GetAlarms		VARIANT: TWContainer TWAAlarmsValue array.
GetDefaultAlarms		VARIANT: TWContainer TWAAlarmsValue array.
ResetAlarms		VARIANT: TWContainer TWAAlarmsValue array.
SetAlarms	VARIANT: TWContainer TWAAlarmsValue array.	
GetAvailableAlarmVolume		VARIANT: TWContainer long range or enumeration w. only one item.
GetCurrentAlarmVolume		Long
GetDefaultAlarmVolume		Long
ResetAlarmVolume		Long
SetAlarmVolume	VARIANT: long value or TWContainer range.	
GetAuthor		String
GetDefaultAuthor		String
ResetAuthor		String
SetAuthor	String	
GetAutoFeed		Boolean
GetDefaultAutoFeed		Boolean
ResetAutoFeed		Boolean
SetAutoFeed	Boolean	
GetAvailableAutomaticCapture		VARIANT: TWContainer long range or enumeration w. only one item.
GetCurrentAutomaticCapture		Long
GetDefaultAutomaticCapture		Long
ResetAutomaticCapture		Long
SetAutomaticCapture	VARIANT: long value or TWContainer range.	
GetAutoScan		Boolean
GetDefaultAutoScan		Boolean
ResetAutoScan		Boolean
SetAutoScan	Boolean	

(continued on next page)

*(continued from previous page)*

GetAvailableBatteryMinutes		VARIANT: TWContainer.
GetCurrentBatteryMinutes		Long
GetDefaultBatteryMinutes		Long
GetBatteryPercentage		Short
GetCameraPreviewUI		Boolean
GetCaption		String
GetDefaultCaption		String
ResetCaption		String
SetCaption	String	
GetClearBuffers		ClearBuffers
GetDefaultClearBuffers		ClearBuffers
ResetClearBuffers		ClearBuffers
SetClearBuffers	ClearBuffers	
SetClearPage	Boolean	
GetCustomDSData		Boolean
GetDeviceEvents		VARIANT: TWContainer TWDeviceEventsValue array.
GetDefaultDeviceEvents		VARIANT: TWContainer TWDeviceEventsValue array.
ResetDeviceEvents		VARIANT: TWContainer TWDeviceEventsValue array.
SetDeviceEvents	VARIANT: TWContainer TWDeviceEventsValue array.	
GetDeviceOnline		Boolean
GetDeviceTimeDate		String
GetDefaultDeviceTimeDate		String
ResetDeviceTimeDate		String
SetDeviceTimeDate	String	
GetDuplex		Duplex
GetDuplexEnabled		Boolean
GetDefaultDuplexEnabled		Boolean
ResetDuplexEnabled		Boolean
SetDuplexEnabled	Boolean	
GetEnableDSUIOnly		Boolean
GetEndorser		long
GetDefaultEndorser		long
ResetEndorser		long
SetEndorser	Long	

*(continued on next page)*

*(continued from previous page)*

GetExtendedCaps		VARIANT: TWContainer TWCapabilityName array.
GetDefaultExtendedCaps		VARIANT: TWContainer TWCapabilityName array.
ResetExtendedCaps		VARIANT: TWContainer TWCapabilityName array.
SetExtendedCaps	VARIANT: TWContainer TWCapabilityName array.	
GetFeederAlignment		FeederAlignment
GetDefaultFeederAlignment		FeederAlignment
ResetFeederAlignment		FeederAlignment
SetFeederAlignment	FeederAlignment	
GetFeederEnabled		Boolean
GetDefaultFeederEnabled		Boolean
ResetFeederEnabled		Boolean
SetFeederEnabled	Boolean	
GetFeederLoaded		Boolean
GetFeederOrder		FeederOrder
GetDefaultFeederOrder		FeederOrder
ResetFeederOrder		FeederOrder
SetFeederOrder	FeederOrder	
SetFeedPage	Boolean	
GetIndicators		Boolean
GetDefaultIndicators		Boolean
ResetIndicators		Boolean
SetIndicators	Boolean	
GetAvailableJobControl		VARIANT: TWContainer TWJobControlValue enumeration.
GetCurrentJobControl		JobControl
GetDefaultJobControl		JobControl
ResetJobControl		JobControl
SetJobControl	JobControl	
GetAvailableLanguage		VARIANT: TWContainer TWLanguageValue enumeration.
GetCurrentLanguage		Language
GetDefaultLanguage		Language
ResetLanguage		Language
SetLanguage	VARIANT: single value (Language, integer or TWLanguageValue) or TWContainer enumeration.	

*(continued on next page)*

*(continued from previous page)*

GetAvailableMaxBatchBuffers		VARIANT: TWContainer.
GetCurrentMaxBatchBuffers		Long
GetDefaultMaxBatchBuffers		Long
ResetMaxBatchBuffers		Long
SetMaxBatchBuffers	Long	
GetReacquireAllowed		Boolean
GetPaperDetectable		Boolean
GetPowerSaveTime		Long
GetDefaultPowerSaveTime		Long
ResetPowerSaveTime		Long
SetPowerSaveTime	long	
GetAvailablePowerSupply		VARIANT: TWContainer TWPowerSupplyValue enumeration.
GetCurrentPowerSupply		PowerSupply
GetAvailablePrinter		VARIANT: TWContainer TWPrinterValue enumeration.
GetCurrentPrinter		Printer
GetDefaultPrinter		Printer
ResetPrinter		Printer
SetPrinter	VARIANT: single value (Printer, integer or TWPrinterValue) or TWContainer enumeration.	
GetPrinterEnabled		Boolean
GetDefaultPrinterEnabled		Boolean
ResetPrinterEnabled		Boolean
SetPrinterEnabled	Boolean	
GetPrinterIndex		Long
GetDefaultPrinterIndex		Long
ResetPrinterIndex		Long
SetPrinterIndex	Long	
GetAvailablePrinterMode		VARIANT: TWContainer TWPrinterMode enumeration.
GetCurrentPrinterMode		PrinterMode
GetDefaultPrinterMode		PrinterMode
ResetPrinterMode		PrinterMode
SetPrinterMode	PrinterMode	
GetAvailablePrinterString		VARIANT: TWContainer.
GetCurrentPrinterString		String
GetDefaultPrinterString		String
ResetPrinterString		String
SetPrinterString	VARIANT: string value or TWContainer enumeration.	

*(continued on next page)*

*(continued from previous page)*

GetPrinterSuffix		String
GetDefaultPrinterSuffix		String
ResetPrinterSuffix		String
SetPrinterSuffix	String	
SetRewindPage	Boolean	
GetSerialNumber		String
GetSupportedCaps		VARIANT: TWContainer TWCcapabilityName array.
GetAvailableTimeBeforeFirstCapture		VARIANT: TWContainer.
GetCurrentTimeBeforeFirstCapture		Long
GetDefaultTimeBeforeFirstCapture		Long
ResetTimeBeforeFirstCapture		Long
SetTimeBeforeFirstCapture	VARIANT: long value or TWContainer range.	
GetAvailableTimeBetweenCaptures		VARIANT: TWContainer.
GetCurrentTimeBetweenCaptures		Long
GetDefaultTimeBetweenCaptures		Long
ResetTimeBetweenCaptures		Long
SetTimeBetweenCaptures	VARIANT: long value or TWContainer range.	
GetTimeDate		String
GetThumbnailsEnabled		Boolean
GetDefaultThumbnailsEnabled		Boolean
ResetThumbnailsEnabled		Boolean
SetThumbnailsEnabled	Boolean	
GetUIControllable		Boolean
GetXferCount		Short
GetDefaultXferCount		Short
ResetXferCount		Short
SetXferCount	Short	
GetIAutomaticBorderDetection		Boolean
GetDefaultIAutomaticBorderDetection		Boolean
ResetIAutomaticBorderDetection		Boolean
SetIAutomaticBorderDetection	Boolean	
GetIAutoBright		Boolean
GetDefaultIAutoBright		Boolean
ResetIAutoBright		Boolean
SetIAutoBright	Boolean	
GetIAutomaticDeskew		Boolean
GetDefaultIAutomaticDeskew		Boolean
ResetIAutomaticDeskew		Boolean
SetIAutomaticDeskew	Boolean	

*(continued on next page)*

(continued from previous page)

GetIAutomaticRotate		Boolean
GetDefaultIAutomaticRotate		Boolean
ResetIAutomaticRotate		Boolean
SetIAutomaticRotate	Boolean	
GetIBarCodeDetectionEnabled		Boolean
GetDefaultIBarCodeDetectionEnabled		Boolean
ResetIBarCodeDetectionEnabled		Boolean
SetIBarCodeDetectionEnabled	Boolean	
GetAvailableIBarCodeMaxRetries		VARIANT: TWContainer
GetCurrentIBarCodeMaxRetries		Long
GetDefaultIBarCodeMaxRetries		Long
ResetIBarCodeMaxRetries		Long
SetIBarCodeMaxRetries	Long	
GetAvailableIBarCodeMaxSearchPriorities		VARIANT: TWContainer
GetCurrentIBarCodeMaxSearchPriorities		Long
GetDefaultIBarCodeMaxSearchPriorities		Long
ResetIBarCodeMaxSearchPriorities		Long
SetIBarCodeMaxSearchPriorities	Long	
GetAvailableIBarCodeSearchMode		VARIANT: TWContainer TWBarCodeSearchMode Value enumeration.
GetCurrentIBarCodeSearchMode		BarCodeSearchMode
GetDefaultIBarCodeSearchMode		BarCodeSearchMode
ResetIBarCodeSearchMode		BarCodeSearchMode
SetIBarCodeSearchMode	BarCodeSearchMode	
GetIBarCodeSearchPriorities		VARIANT: TWContainer TWBarCodeTypeValue array.
GetDefaultIBarCodeSearchPriorities		VARIANT: TWContainer TWBarCodeTypeValue array.
ResetIBarCodeSearchPriorities		VARIANT: TWContainer TWBarCodeTypeValue array.
SetIBarCodeSearchPriorities	VARIANT: TWContainer TWBarCodeTypeValue array.	
GetAvailableIBarCodeTimeout		VARIANT: TWContainer.
GetCurrentIBarCodeTimeout		Long
GetDefaultIBarCodeTimeout		Long
ResetIBarCodeTimeout		Long
SetIBarCodeTimeout	Long	
GetAvailableIBitDepth		VARIANT: TWContainer enumeration.
GetCurrentIBitDepth		Short
GetDefaultIBitDepth		Short
ResetIBitDepth		Short
SetIBitDepth	VARIANT: short value or TWContainer enumeration.	

(continued on next page)

(continued from previous page)

GetAvailableBitDepthReduction		VARIANT: TWContainer TWBitDepthReductionValue enumeration.
GetCurrentBitDepthReduction		BitDepthReduction
GetDefaultBitDepthReduction		BitDepthReduction
ResetBitDepthReduction		BitDepthReduction
SetBitDepthReduction	VARIANT: single value (BitDepthReduction, integer or TWBitDepthReduction) or TWContainer enumeration.	
GetAvailableBitOrder		VARIANT: TWContainer TWBitOrderValue enumeration.
GetCurrentBitOrder		BitOrder
GetDefaultBitOrder		BitOrder
ResetBitOrder		BitOrder
SetBitOrder	BitOrder	
GetAvailableBitOrderCodes		VARIANT: TWContainer TWBitOrderValue enumeration.
GetCurrentBitOrderCodes		BitOrder
GetDefaultBitOrderCodes		BitOrder
ResetBitOrderCodes		BitOrder
SetBitOrderCodes	BitOrder	
GetAvailableBrightness		VARIANT: TWContainer enumeration or range.
GetCurrentBrightness		Float
GetDefaultBrightness		Float
ResetBrightness		Float
SetBrightness	VARIANT: float value, TWContainer enumeration or range.	
GetAvailableCCITTKFactor		Short
GetDefaultCCITTKFactor		Short
ResetCCITTKFactor		Short
SetCCITTKFactor	Short	
GetAvailableCompression		VARIANT: TWContainer TWCompressionValue enumeration
GetCurrentCompression		Compression
GetDefaultCompression		Compression
ResetCompression		Compression
SetCompression	VARIANT: single value (Compression, integer or TWCompressionValue) or TWContainer enumeration.	

(continued on next page)

*(continued from previous page)*

GetAvailableContrast		VARIANT: TWContainer enumeration or range.
GetCurrentContrast		Float
GetDefaultContrast		Float
ResetContrast		Float
SetContrast	VARIANT: float value, TWContainer enumeration or range.	
GetCustHalfTone		VARIANT: TWContainer array (short integers).
GetDefaultCustHalfTone		VARIANT: TWContainer array (short integers).
ResetCustHalfTone		VARIANT: TWContainer array (short integers).
SetCustHalfTone	VARIANT: TWContainer array (short integers).	
GetAvailableExposureTime		VARIANT: TWContainer enumeration or range.
GetCurrentExposureTime		Float
GetDefaultExposureTime		Float
ResetExposureTime		Float
SetExposureTime	VARIANT: float value, TWContainer enumeration or range.	
GetExtImageInfo		Boolean
GetDefaultExtImageInfo		Boolean
ResetExtImageInfo		Boolean
SetExtImageInfo	Boolean	
GetFilter		VARIANT: TWContainer TWFilterValue array.
GetDefaultFilter		VARIANT: TWContainer TWFilterValue array.
ResetFilter		VARIANT: TWContainer TWFilterValue array.
SetFilter	VARIANT: TWContainer TWFilterValue array.	
GetFlashUsed		Boolean
GetDefaultFlashUsed		Boolean
ResetFlashUsed		Boolean
SetFlashUsed	Boolean	

*(continued on next page)*

(continued from previous page)

GetAvailableFlashUsed2		VARIANT: TWContainer TWFlashUsed2Value enumeration.
GetCurrentFlashUsed2		FlashUsed2
GetDefaultFlashUsed2		FlashUsed2
ResetFlashUsed2		FlashUsed2
SetFlashUsed2	VARIANT: single value (FlashUsed2, integer or TWFlashUsed2Value) or TWContainer enumeration.	
GetFlipRotation		FlipRotation
GetDefaultFlipRotation		FlipRotation
ResetFlipRotation		FlipRotation
SetFlipRotation	FlipRotation	
GetAvailableFrames		VARIANT: TWContainer TWFrame enumeration.
GetCurrentFrames		VARIANT: TWFrame.
GetDefaultFrames		VARIANT: TWFrame.
ResetFrames		VARIANT: TWFrame.
SetFrames	VARIANT: TWFrame or TWContainer TWFrame enumeration.	
GetGamma		Float
GetDefaultGamma		Float
ResetGamma		Float
SetGamma	Float	
GetAvailableHalfTones		VARIANT: TWContainer string enumeration or array.
GetCurrentHalfTones		String
GetDefaultHalfTones		String
ResetHalfTones		String
SetHalfTones	VARIANT: string value, TWContainer enumeration or array.	
GetAvailableHighlight		VARIANT: TWContainer enumeration or range.
GetCurrentHighlight		Float
GetDefaultHighlight		Float
ResetHighlight		Float
SetHighlight	VARIANT: float value, TWContainer enumeration or range.	
GetImageDataSet		VARIANT: TWContainer.
SetImageDataSet	VARIANT: numeric value, TWContainer enumeration or range.	

(continued on next page)

*(continued from previous page)*

GetAvailableImageFileFormat		VARIANT: TWContainer TWImageFileFormatValue enumeration.
GetCurrentImageFileFormat		ImageFileFormat
GetDefaultImageFileFormat		ImageFileFormat
ResetImageFileFormat		ImageFileFormat
SetImageFileFormat	VARIANT: single value (ImageFileFormat, integer, TWImageFileFormatValue) or TWContainer enumeration.	
GetAvailableImageFilter		VARIANT: TWContainer TWImageFilterValue enumeration.
GetCurrentImageFilter		ImageFilter
GetDefaultImageFilter		ImageFilter
ResetImageFilter		ImageFilter
SetImageFilter	VARIANT: single value (ImageFilter, integer or TWImageFilterValue) or TWContainer enumeration.	
GetAvailableJPEGPixelFormat		VARIANT: TWContainer TWPixelFormatValue enumeration.
GetCurrentJPEGPixelFormat		PixelFormat
GetDefaultJPEGPixelFormat		PixelFormat
ResetJPEGPixelFormat		PixelFormat
SetJPEGPixelFormat	VARIANT: single value (PixelFormat, integer or TWPixelFormatValue) or TWContainer enumeration.	
GetLampState		Boolean
GetDefaultLampState		Boolean
ResetLampState		Boolean
SetLampState	Boolean	
GetAvailableLightPath		VARIANT: TWContainer TWLightPathValue enumeration.
GetCurrentLightPath		LightPath
GetDefaultLightPath		LightPath
ResetLightPath		LightPath
SetLightPath	LightPath	

*(continued on next page)*

(continued from previous page)

GetAvailableLightSource		VARIANT: TWContainer TWLightSourceValue enumeration.
GetCurrentLightSource		LightSource
GetDefaultLightSource		LightSource
ResetLightSource		LightSource
SetLightSource	VARIANT: single value (LightSource, integer or TWLightSourceValues) or TWContainer enumeration.	
GetMaxFrames		Short
GetDefaultMaxFrames		Short
ResetMaxFrames		Short
SetMaxFrames	Short	
GetMinimumHeight		Float
GetMinimumWidth		Float
GetAvailableNoiseFilter		VARIANT: TWContainer TWNNoiseFilterValue enumeration.
GetCurrentNoiseFilter		NoiseFilter
GetDefaultNoiseFilter		NoiseFilter
ResetNoiseFilter		NoiseFilter
SetNoiseFilter	VARIANT: single value (NoiseFilter, integer or TWNNoiseFilterValue) or TWContainer enumeration.	
GetAvailableOrientation		VARIANT: TWContainer TWOrientationValue enumeration.
GetCurrentOrientation		Orientation
GetDefaultOrientation		Orientation
ResetOrientation		Orientation
SetOrientation	VARIANT: single value (Orientation, integer or TWOrientationValue) or TWContainer enumeration.	
GetAvailableOverScan		VARIANT: TWContainer TWOverScanValue enumeration.
GetCurrentOverScan		OverScan
GetDefaultOverScan		OverScan
ResetOverScan		OverScan
SetOverScan	VARIANT: single value (OverScan, integer or TWOverScanValue) or TWContainer enumeration.	

(continued on next page)

*(continued from previous page)*

GetIPatchCodeDetectionEnabled		Boolean
GetDefaultIPatchCodeDetectionEnabled		Boolean
ResetIPatchCodeDetectionEnabled		Boolean
SetIPatchCodeDetectionEnabled	Boolean	
GetAvailableIPatchCodeMaxRetries		VARIANT: TWContainer.
GetCurrentIPatchCodeMaxRetries		Long
GetDefaultIPatchCodeMaxRetries		Long
ResetIPatchCodeMaxRetries		Long
SetIPatchCodeMaxRetries	Long	
GetAvailableIPatchCodeMaxSearchPriorities		VARIANT: TWContainer.
GetCurrentIPatchCodeMaxSearchPriorities		Long
GetDefaultIPatchCodeMaxSearchPriorities		Long
ResetIPatchCodeMaxSearchPriorities		Long
SetIPatchCodeMaxSearchPriorities	Long	
GetAvailableIPatchCodeSearchMode		VARIANT: TWContainer TWPatchCodeSearchMode Value enumeration.
GetCurrentIPatchCodeSearchMode		PatchCodeSearchMode
GetDefaultIPatchCodeSearchMode		PatchCodeSearchMode
ResetIPatchCodeSearchMode		PatchCodeSearchMode
SetIPatchCodeSearchMode	PatchCodeSearchMode	
GetIPatchCodeSearchPriorities		VARIANT: TWContainer TWPatchCodeValue array.
GetDefaultIPatchCodeSearchPriorities		VARIANT: TWContainer TWPatchCodeValue array.
ResetIPatchCodeSearchPriorities		VARIANT: TWContainer TWPatchCodeValue array.
SetIPatchCodeSearchPriorities	VARIANT: TWContainer TWPatchCodeValue array.	
GetAvailableIPatchCodeTimeout		VARIANT: TWContainer.
GetCurrentIPatchCodeTimeout		Long
GetDefaultIPatchCodeTimeout		Long
ResetIPatchCodeTimeout		Long
SetIPatchCodeTimeout	Long	
GetIPhysicalHeight		Float
GetIPhysicalWidth		Float
GetAvailableIPixelFlavor		VARIANT: TWContainer TWPixelFlavorValue enumeration.
GetCurrentIPixelFlavor		PixelFlavor
GetDefaultIPixelFlavor		PixelFlavor
ResetIPixelFlavor		PixelFlavor
SetIPixelFlavor	PixelFlavor	

*(continued on next page)*

*(continued from previous page)*

GetAvailableIPixelFlavorCodes		VARIANT: TWContainer TWPixelFlavorValue enumeration.
GetCurrentIPixelFlavorCodes		PixelFlavor
GetDefaultIPixelFlavorCodes		PixelFlavor
ResetIPixelFlavorCodes		PixelFlavor
SetIPixelFlavorCodes	PixelFlavor	
GetAvailableIPixelType		VARIANT: TWContainer TWPixelTypeValue enumeration.
GetCurrentIPixelType		PixelType
GetDefaultIPixelType		PixelType
ResetIPixelType		PixelType
SetIPixelType	VARIANT: single value (PixelType, integer or TWPixelTypeValue) or TWContainer enumeration.	
GetAvailableIPlanarChunky		VARIANT: TWContainer TWPlanarChunkyValue enumeration.
GetCurrentIPlanarChunky		PlanarChunky
GetDefaultIPlanarChunky		PlanarChunky
ResetIPlanarChunky		PlanarChunky
SetIPlanarChunky	VARIANT: single value (PlanarChunky, integer or TWPlanarChunkyValue) or TWContainer enumeration.	
GetAvailableIRotation		VARIANT: TWContainer enumeration or range.
GetCurrentIRotation		Float
GetDefaultIRotation		Float
ResetIRotation		Float
SetIRotation	Float	
GetAvailableIShadow		VARIANT: TWContainer enumeration or range.
GetCurrentIShadow		Float
GetDefaultIShadow		Float
ResetIShadow		Float
SetIShadow	VARIANT: float value, TWContainer enumeration or range.	
GetISupportedBarCodeTypes		VARIANT: TWContainer TWBarCodeTypeValue array.
GetISupportedPatchCodeTypes		VARIANT: TWContainer TWPatchCodeValue array.

*(continued on next page)*

*(continued from previous page)*

GetAvailableISupportedSizes		VARIANT: TWContainer TWSupportedSizesValue enumeration.
GetCurrentISupportedSizes		SupportedSizes
GetDefaultISupportedSizes		SupportedSizes
ResetISupportedSizes		SupportedSizes
SetISupportedSizes	VARIANT: single value (SupportedSizes, integer or TWSupportedSizes) or TWContainer enumeration.	
GetAvailableIThreshold		VARIANT: TWContainer enumeration or range.
GetCurrentIThreshold		Float
GetDefaultIThreshold		Float
ResetIThreshold		Float
SetIThreshold	VARIANT: float value, TWContainer enumeration or range.	
GetITiles		Boolean
GetDefaultITiles		Boolean
ResetITiles		Boolean
SetITiles	Boolean	
GetAvailableITimeFill		VARIANT: TWContainer.
GetCurrentITimeFill		Long
GetDefaultITimeFill		Long
ResetITimeFill		Long
SetITimeFill	Long	
GetIUndefinedImageSize		Boolean
GetDefaultIUndefinedImageSize		Boolean
ResetIUndefinedImageSize		Boolean
SetIUndefinedImageSize	Boolean	
GetAvailableIUnits		VARIANT: TWContainer TWUnitsValue enumeration.
GetCurrentIUnits		Units
GetDefaultIUnits		Units
ResetIUnits		Units
SetIUnits	VARIANT: single value (Units, integer or TWUnitsValue) or TWContainer enumeration.	

*(continued on next page)*

(continued from previous page)

GetAvailableIXferMech		VARIANT: TWContainer TWXferMechValue enumeration.
GetCurrentIXferMech		XferMech
GetDefaultIXferMech		XferMech
ResetIXferMech		XferMech
SetIXferMech	VARIANT: single value (XferMech, integer or TWXferMechValue) or TWContainer enumeration.	
GetAvailableIXNativeResolution		VARIANT: TWContainer.
GetCurrentIXNativeResolution		Float
GetDefaultIXNativeResolution		Float
GetAvailableIXResolution		VARIANT: TWContainer enumeration or range.
GetCurrentIXResolution		Float
GetDefaultIXResolution		Float
ResetIXResolution		Float
SetIXResolution	VARIANT: float value, TWContainer enumeration or range.	
GetAvailableIXScaling		VARIANT: TWContainer enumeration or range.
GetCurrentIXScaling		Float
GetDefaultIXScaling		Float
ResetIXScaling		Float
SetIXScaling	VARIANT: float value, TWContainer enumeration or range.	
GetAvailableIYNativeResolution		VARIANT: TWContainer.
GetCurrentIYNativeResolution		Float
GetDefaultIYNativeResolution		Float
GetAvailableIYResolution		VARIANT: TWContainer enumeration or range.
GetCurrentIYResolution		Float
GetDefaultIYResolution		Float
ResetIYResolution		Float
SetIYResolution	VARIANT: float value, TWContainer enumeration or range.	
GetAvailableIYScaling		VARIANT: TWContainer enumeration or range.
GetCurrentIYScaling		Float
GetDefaultIYScaling		Float
ResetIYScaling		Float
SetIYScaling	VARIANT: float value, TWContainer enumeration or range.	

(continued on next page)

*(continued from previous page)*

GetAvailableZoomFactor		VARIANT: TWContainer.
GetCurrentZoomFactor		Short
GetDefaultZoomFactor		Short
ResetZoomFactor		Short
SetZoomFactor	Short	

## R.6. TWControl ActiveX Events

The ObjectTWAIN™ ActiveX Library notifies an application when an image transfer (or just an image buffer for Memory Mode) is received and when a special transfer condition occurs (e.g. all the images were transferred or a transfer error occurred) using TWControl ActiveX events:

Event	Description
<b>DoProcessNativeData</b>	The application receives a transferred image as a handle to a DIB in memory.
<b>DoProcessMemoryData</b>	The application receives a raw buffer containing a strip or tile of the image to transfer.
<b>DoProcessFileData</b>	The application receives the file format and the file name where a transferred image was just saved.
<b>DoProcessNativeAsFileData</b>	The application receives the file format and the file name where a transferred image was just saved.
<b>DoProcessMemoryAsFileData</b>	The application receives the file format and the file name where a transferred image was just saved.
<b>OnFeederLoaded</b>	The application is notified that the user put paper in the device's Document Feeder.
<b>OnBeginTransfers</b>	The application is notified that the transfers are ready to begin (the Source sent the Transfer Ready signal).
<b>OnEndTransfers</b>	The application is notified that all available images were transferred and the Data Source does not have more pending transfers.
<b>OnCancelTransfer</b>	The application is notified that either the Source or the user canceled a transfer and the acquisition sequence.
<b>OnTransferError</b>	The application is notified that a transfer error occurred (the application may receive the error code and a error text description).
<b>OnPreTransfer</b>	The application is notified that a transfer is about to begin. The TWAIN session is still in state 6.
<b>OnPostTransfer</b>	The application is notified that a transfer completed. The TWAIN session is in state 6 or 5 (if there are no more transfers pending).
<b>OnDeviceEvent</b>	The application is notified that the device triggered an event.
<b>OnUserCloseSourceUI</b>	The application is notified that the user requested to close the Data Source UI. The event parameter indicates if the user closes the DS UI in 'OK' or 'Cancel' mode.

## R.6. Other ObjectTWIN™ COM Objects Interfaces

In the following table, the names marked with () are methods, the other are properties (get/set):

COM Object	Properties & Methods	Description
<b>TWContainer</b>	<b>Type()</b> <b>InitEnumeration</b> (MaxItems) <b>InitArray</b> (MaxItems) <b>InitRange</b> (Min, Max, Step) <u>Enumeration/Array only:</u> <b>Count()</b> <b>Item</b> (Index) <b>Insert</b> (NewVal, BeforeIndex) <b>Add</b> (NewVal) <b>Remove</b> (Index) <u>Enumeration only:</u> <b>CurrentIndex</b> <b>DefaultIndex</b> <u>Range only:</u> <b>RangeMin</b> <b>RangeMax</b> <b>RangeStep</b> <b>RangeDefault</b> <b>RangeCurrent</b>	Returns the container type (ConType). Initializes a new empty container as enum. Initializes a new empty container as array. Initializes a new empty container as range. Returns the number of items in container. Returns the item at the specified index. Inserts a new item at the specified index. Adds a new item to container. Removes the item at specified index. Enumeration container attribute. Enumeration container attribute. Range container attribute. Range container attribute. Range container attribute. Range container attribute. Range container attribute.
<b>TWSourceList</b>	<b>Count()</b> <b>Item</b> (Index) <b>DefaultIndex</b> ()	Returns the number of available Sources. Returns TWSourceInfo at given index. Returns the index of the default Source.
<b>TWSourceInfo</b>	<b>ProductName</b> () <b>ProductFamily</b> () <b>Manufacturer</b> () <b>VersionInfo</b> () <b>VersionMajor</b> () <b>VersionMinor</b> () <b>ProtocolMajor</b> () <b>ProtocolMinor</b> () <b>Language</b> () <b>Country</b> () <b>AudioData</b> () <b>ImageData</b> ()	Returns the product name string. Returns the product family string. Returns the manufacturer name string. Returns the version info string. Returns the version major number. Returns the version minor number. Returns the TWAIN version major number. Returns the TWAIN version minor number. Returns the language id (Language). Returns the country id (Country). Returns True if Source supports Audio. Returns True if Source supports Image.
<b>TWImageLayout</b>	<b>FrameLeft</b> <b>FrameTop</b> <b>FrameRight</b> <b>FrameBottom</b> <b>DocumentNumber</b> <b>PageNumber</b> <b>FrameNumber</b>	Acquisition frame coordinate. Acquisition frame coordinate. Acquisition frame coordinate. Acquisition frame coordinate. Document number. Page number Frame number.

(continued on next page)

(continued from previous page)

<b>TWImageInfo</b>	<b>ImageLength()</b> <b>ImageWidth()</b> <b>PixelFormat()</b> <b>Compression()</b> <b>BitsPerPixel()</b> <b>SamplesPerPixel()</b> <b>BitsPerSampleOne()</b> <b>BitsPerSampleTwo()</b> <b>BitsPerSampleThree()</b> <b>BitsPerSampleFour()</b> <b>BitsPerSampleFive()</b> <b>BitsPerSampleSix()</b> <b>BitsPerSampleSeven()</b> <b>BitsPerSampleEight()</b> <b>PlanarChunky()</b> <b>Xresolution()</b> <b>Yresolution()</b>	Returns the image length in pixels. Returns the image width in pixels. Returns the pixel type (PixelFormat). Returns the compression (Compression). Returns the number of bits per pixel. Returns the number of samples pp. Returns the number of bits in sample #1. Returns the number of bits in sample #2. Returns the number of bits in sample #3. Returns the number of bits in sample #4. Returns the number of bits in sample #5. Returns the number of bits in sample #6. Returns the number of bits in sample #7. Returns the number of bits in sample #8. Returns the image color transfer mode. Returns the resolution on x-axis (in p/units). Returns the resolution on y-axis (in p/units).
<b>TWInfo</b>	<b>Count()</b> <b>Init(MaxItems)</b> <b>Item(Index)</b> <b>Add()</b> <b>InfoID</b> <b>ReturnCode</b>	Returns the number of items in info. Initializes the item array. Returns the item at specified index. Adds a new item. The info ID (ExtendedImageInfo). The Return Code (ReturnCode).
<b>TWFrame</b>	<b>Left</b> <b>Top</b> <b>Right</b> <b>Bottom</b>	Left frame coordinate. Top frame coordinate. Right frame coordinate. Bottom frame coordinate.
<b>TWCapQuery</b>	<b>SupportsGet()</b> <b>SupportsGetCurrent()</b> <b>SupportsGetDefault()</b> <b>SupportsReset()</b> <b>SupportsSet()</b>	Returns if MSG_GET supported on cap. Returns if MSG_GETCURRENT supported. Returns if MSG_GETDEFAULT supported. Returns if MSG_RESET supported on cap. Returns if MSG_SET supported on cap.

## R.7. ObjectTWAIN™ COM Constant Enumerations

Enumeration	Values
<b>Compression</b>	CMP_NONE = 0 CMP_PACKBITS = 1 CMP_GROUP31D = 2 CMP_GROUP31DEOL = 3 CMP_GROUP32D = 4 CMP_GROUP4 = 5 CMP_JPEG = 6 CMP_LZW = 7 CMP_JBIG = 8
<b>PixelType</b>	PT_BW = 0 PT_GRAY = 1 PT_RGB = 2 PT_PALETTE = 3 PT_CMY = 4 PT_CMYK = 5 PT_YUV = 6 PT_YUVK = 7 PT_CIEXYZ = 8
<b>Units</b>	UN_INCHES = 0 UN_CENTIMETERS = 1 UN_PICAS = 2 UN_POINTS = 3 UN_TWIPS = 4 UN_PIXELS = 5
<b>XferMech</b>	SX_NATIVE = 0 SX_FILE = 1 SX_MEMORY = 2
<b>CapabilityName</b>	C_CUSTOMBASE = 0x8000 C_XFERCOUNT = 0x0001 IC_COMPRESSION = 0x0100 IC_PIXELTYPE = 0x0101 IC_UNITS = 0x0102 IC_XFERMECH = 0x0103 C_AUTHOR = 0x1000 C_CAPTION = 0x1001 C_FEEDERENABLED = 0x1002 C_FEEDERLOADED = 0x1003 C_TIMEDATE = 0x1004 C_SUPPORTEDCAPS = 0x1005 C_EXTENDEDCAPS = 0x1006 C_AUTOFEED = 0x1007 C_CLEARPAGE = 0x1008 C_FEEDPAGE = 0x1009 C_REWINDPAGE = 0x100a C_INDICATORS = 0x100b C_SUPPORTEDCAPSEXT = 0x100c C_PAPERDETECTABLE = 0x100d C_UICONTROLLABLE = 0x100e C_DEVICEONLINE = 0x100f C_AUTOSCAN = 0x1010 C_THUMBNAILENABLED = 0x1011 C_DUPLEX = 0x1012 C_DUPLEXENABLED = 0x1013 C_ENABLEDSUIONLY = 0x1014 C_CUSTOMSDATA = 0x1015 <i>(continued on next page)</i>

<b>CapabilityName (2)</b>	C_ENDORSER = 0x1016 C_JOBCONTROL = 0x1017 C_ALARMS = 0x1018 C_ALARMVOLUME = 0x1019 C_AUTOMATICCAPTURE = 0x101a C_TIMEBEFOREFIRSTCAPTURE = 0x101b C_TIMEBETWEENCAPTURES = 0x101c C_CLEARBUFFERS = 0x101d C_MAXBATCHBUFFERS = 0x101e C_DEVICETIMEDATE = 0x101f C_POWERSUPPLY = 0x1020 C_CAMERAPREVIEWUI = 0x1021 C_DEVICEEVENT = 0x1022 C_REACQUIREALLOWED = 0x1030 C_SERIALNUMBER = 0x1024 C_PRINTER = 0x1026 C_PRINTERENABLED = 0x1027 C_PRINTERINDEX = 0x1028 C_PRINTERMODE = 0x1029 C_PRINTERSTRING = 0x102a C_PRINTERSUFFIX = 0x102b C_LANGUAGE = 0x102c C_FEEDERALIGNMENT = 0x102d C_FEEDERORDER = 0x102e C_BATTERYMINUTES = 0x1032 C_BATTERYPERCENTAGE = 0x1033 IC_AUTOBRIGHT = 0x1100 IC_BRIGHTNESS = 0x1101 IC_CONTRAST = 0x1103 IC_CUSTHALFTONE = 0x1104 IC_EXPOSURETIME = 0x1105 IC_FILTER = 0x1106 IC_FLASHUSED = 0x1107 IC_GAMMA = 0x1108 IC_HALFTONES = 0x1109 IC_HIGHLIGHT = 0x110a IC_IMAGEFILEFORMAT = 0x110c IC_LAMPSTATE = 0x110d IC_LIGHTSOURCE = 0x110e IC_ORIENTATION = 0x1110 IC_PHYSICALWIDTH = 0x1111 IC_PHYSICALHEIGHT = 0x1112 IC_SHADOW = 0x1113 IC_FRAMES = 0x1114 IC_XNATIVERESOLUTION = 0x1116 IC_YNATIVERESOLUTION = 0x1117 IC_XRESOLUTION = 0x1118 IC_YRESOLUTION = 0x1119 IC_MAXFRAMES = 0x111a IC_TILES = 0x111b IC_BITORDER = 0x111c IC_CCITTKFACTOR = 0x111d IC_LIGHTPATH = 0x111e IC_PIXELFLAVOR = 0x111f IC_PLANARCHUNKY = 0x1120 IC_ROTATION = 0x1121 IC_SUPPORTEDSIZES = 0x1122 IC_THRESHOLD = 0x1123 IC_XSCALING = 0x1124 IC_YSCALING = 0x1125 <i>(continued on next page)</i>
---------------------------	---

<b>CapabilityName (3)</b>	IC_BITORDERCODES = 0x1126 IC_PIXELFLAVORCODES = 0x1127 IC_JPEGPIXELTYPE = 0x1128 IC_TIMEFILL = 0x112a IC_BITDEPTH = 0x112b IC_BITDEPTHREDUCTION = 0x112c IC_UNDEFINEDIMAGESIZE = 0x112d IC_IMAGEDATASET = 0x112e IC_EXTIMAGEINFO = 0x112f IC_MINIMUMHEIGHT = 0x1130 IC_MINIMUMWIDTH = 0x1131 IC_AUTOBORDERDETECTION = 0x1132 IC_AUTODESKEW = 0x1133 IC_AUTOROTATE = 0x1135 IC_FLIPROTATION = 0x1136 IC_BARCODEDETECTIONENABLED = 0x1137 IC_SUPPORTEDBARCODETYPES = 0x1138 IC_BARCODEMAXSEARCHPRIORITIES = 0x1139 IC_BARCODESEARCHPRIORITIES = 0x113a IC_BARCODESEARCHMODE = 0x113b IC_BARCODEMAXRETRIES = 0x113c IC_BARCODETIMEOUT = 0x113d IC_ZOOMFACTOR = 0x113e IC_PATCHCODEDETECTIONENABLED = 0x113f IC_SUPPORTEDPATCHCODETYPES = 0x1140 IC_PATCHCODEMAXSEARCHPRIORITIES = 0x1141 IC_PATCHCODESEARCHPRIORITIES = 0x1142 IC_PATCHCODESEARCHMODE = 0x1143 IC_PATCHCODEMAXRETRIES = 0x1144 IC_PATCHCODETIMEOUT = 0x1145 IC_FLASHUSED2 = 0x1146 IC_IMAGEFILTER = 0x1147 IC_NOISEFILTER = 0x1148 IC_OVERSCAN = 0x1149 IC_AUTOMATICBORDERDETECTION = 0x1150 IC_AUTOMATICDESKEW = 0x1151 IC_AUTOMATICROTATE = 0x1152 AC_AUDIOFILEFORMAT = 0x1201 AC_XFERMECH = 0x1202
<b>Duplex</b>	DX_NONE = 0 DX_1PASSDUPLEX = 1 DX_2PASSDUPLEX = 2
<b>JobControl</b>	JC_NONE = 0 JC_JSIC = 1 JC_J SIS = 2 JC_JSXC = 3 JC_JSXS = 4
<b>Filter</b>	FT_RED = 0 FT_GREEN = 1 FT_BLUE = 2 FT_NONE = 3 FT_WHITE = 4 FT_CYAN = 5 FT_MAGENTA = 6 FT_YELLOW = 7 FT_BLACK = 8

<b>ImageFileFormat</b>	FF_TIFF = 0 FF_PICT = 1 FF_BMP = 2 FF_XBM = 3 FF_JFIF = 4 FF_FPX = 5 FF_TIFFMULTI = 6 FF_PNG = 7 FF_SPIFF = 8 FF_EXIF = 9
<b>LightSource</b>	LS_RED = 0 LS_GREEN = 1 LS_BLUE = 2 LS_NONE = 3 LS_WHITE = 4 LS_UV = 5 LS_IR = 6
<b>Orientation</b>	OR_ROT0 = 0 OR_ROT90 = 1 OR_ROT180 = 2 OR_ROT270 = 3 OR_PORTRAIT = OR_ROT0 OR_LANDSCAPE = OR_ROT270
<b>BitOrder</b>	BO_LSBFIRST = 0 BO_MSBFIRST = 1
<b>LightPath</b>	LP_REFLECTIVE = 0 LP_TRANSMISSIVE = 1
<b>PixelFlavor</b>	PF_CHOCOLATE = 0 PF_VANILLA = 1
<b>PlanarChunky</b>	PC_CHUNKY = 0 PC_PLANAR = 1
<b>SupportedSizes</b>	SS_NONE = 0 SS_A4LETTER = 1 SS_B5LETTER = 2 SS_USLETTER = 3 SS_USLEGAL = 4 SS_A5 = 5 SS_B4 = 6 SS_B6 = 7 SS_USLEDGER = 9 SS_USEXECUTIVE = 10 SS_A3 = 11 SS_B3 = 12 SS_A6 = 13 SS_C4 = 14 SS_C5 = 15 SS_C6 = 16 SS_4A0 = 17 SS_2A0 = 18 SS_A0 = 19 SS_A1 = 20 SS_A2 = 21 SS_A4 = SS_A4LETTER SS_A7 = 22 SS_A8 = 23 SS_A9 = 24 SS_A10 = 25 SS_ISO B0 = 26 SS_ISO B1 = 27 <i>(continued on next page)</i>

<b>SupportedSizes (2)</b>	SS_ISO2 = 28 SS_ISO3 = SS_B3 SS_ISO4 = SS_B4 SS_ISO5 = 29 SS_ISO6 = SS_B6 SS_ISO7 = 30 SS_ISO8 = 31 SS_ISO9 = 32 SS_ISO10 = 33 SS_JISB0 = 34 SS_JISB1 = 35 SS_JISB2 = 36 SS_JISB3 = 37 SS_JISB4 = 38 SS_JISB5 = SS_B5LETTER SS_JISB6 = 39 SS_JISB7 = 40 SS_JISB8 = 41 SS_JISB9 = 42 SS_JISB10 = 43 SS_C0 = 44 SS_C1 = 45 SS_C2 = 46 SS_C3 = 47 SS_C7 = 48 SS_C8 = 49 SS_C9 = 50 SS_C10 = 51 SS_USSTATEMENT = 52 SS_BUSINESSCARD = 53
<b>BitDepthReduction</b>	BR_THRESHOLD = 0 BR_HALFTONE = 1 BR_CUSTHALFTONE = 2 BR_DIFFUSION = 3
<b>Alarms</b>	AL_ALARMS = 0 AL_FEEDERERROR = 1 AL_FEEDERWARNING = 2 AL_BARCODE = 3 AL_DOUBLEFEED = 4 AL_JAM = 5 AL_PATCHCODE = 6 AL_POWER = 7 AL_SKEW = 8
<b>DeviceEvents</b>	DE_CHECKAUTOMATICCAPTURE = 0 DE_CHECKBATTERY = 1 DE_CHECKDEVICEONLINE = 2 DE_CHECKFLASH = 3 DE_CHECKPOWERSUPPLY = 4 DE_CHECKRESOLUTION = 5 DE_DEVICEADDED = 6 DE_DEVICEOFFLINE = 7 DE_DEVICEREADY = 8 DE_DEVICEREMOVED = 9 DE_IMAGECAPTURED = 10 DE_IMAGEDELETED = 11 DE_PAPERDOUBLEFEED = 12 DE_PAPERJAM = 13 DE_LAMPFAILURE = 14 DE_POWERSAVE = 15 DE_POWERSAVENOTIFY = 16

<b>ClearBuffers</b>	CB_AUTO = 0 CB_CLEAR = 1 CB_NOCLEAR = 2
<b>AudioFileFormat</b>	AF_WAV = 0 AF_AIFF = 1 AF_AU = 3 AF_SND = 4
<b>FeederAlignment</b>	FA_NONE = 0 FA_LEFT = 1 FA_CENTER = 2 FA_RIGHT = 3
<b>FeederOrder</b>	FO_FIRSTPAGEFIRST = 0 FO_LASTPAGEFIRST = 1
<b>Language</b>	LG_DAN = 0 LG_DUT = 1 LG_ENG = 2 LG_FCF = 3 LG_FIN = 4 LG_FRN = 5 LG_GER = 6 LG_ICE = 7 LG_ITN = 8 LG_NOR = 9 LG_POR = 10 LG_SPA = 11 LG_SWE = 12 LG_USA = 13 LG_USERLOCALE = -1 LG_AFRIKAANS = 14 LG_ALBANIA = 15 LG_ARABIC = 16 LG_ARABIC_ALGERIA = 17 LG_ARABIC_BAHRAIN = 18 LG_ARABIC_EGYPT = 19 LG_ARABIC_IRAQ = 20 LG_ARABIC_JORDAN = 21 LG_ARABIC_KUWAIT = 22 LG_ARABIC_LEBANON = 23 LG_ARABIC_LIBYA = 24 LG_ARABIC_MOROCCO = 25 LG_ARABIC_OMAN = 26 LG_ARABIC_QATAR = 27 LG_ARABIC_SAUDIARABIA = 28 LG_ARABIC_SYRIA = 29 LG_ARABIC_TUNISIA = 30 LG_ARABIC_UAE = 31 LG_ARABIC_YEMEN = 32 LG_BASQUE = 33 LG_BYELORUSSIAN = 34 LG_BULGARIAN = 35 LG_CATALAN = 36 LG_CHINESE = 37 LG_CHINESE_HONGKONG = 38 LG_CHINESE_PRC = 39 LG_CHINESE_SINGAPORE = 40 LG_CHINESE_SIMPLIFIED = 41 LG_CHINESE_TAIWAN = 42 LG_CHINESE_TRADITIONAL = 43 LG_CROATIA = 44 LG_CZECH = 45 <i>(continued on next page)</i>

<b>Language (2)</b>	LG_DANISH = LG_DAN LG_DUTCH = LG_DUT LG_DUTCH_BELGIAN = 46 LG_ENGLISH = LG_ENG LG_ENGLISH_AUSTRALIAN = 47 LG_ENGLISH_CANADIAN = 48 LG_ENGLISH_IRELAND = 49 LG_ENGLISH_NEWZEALAND = 50 LG_ENGLISH_SOUTHAFRICA = 51 LG_ENGLISH_UK = 52 LG_ENGLISH_USA = LG_USA LG_ESTONIAN = 53 LG_FAEROESE = 54 LG_FARSI = 55 LG_FINNISH = LG_FIN LG_FRENCH = LG_FRN LG_FRENCH_BELGIAN = 56 LG_FRENCH_CANADIAN = LG_FCF LG_FRENCH_LUXEMBOURG = 57 LG_FRENCH_SWISS = 58 LG_GERMAN = LG_GER LG_GERMAN_AUSTRIAN = 59 LG_GERMAN_LUXEMBOURG = 60 LG_GERMAN_LIECHTENSTEIN = 61 LG_GERMAN_SWISS = 62 LG_GREEK = 63 LG_HEBREW = 64 LG_HUNGARIAN = 65 LG_ICELANDIC = LG_ICE LG_INDONESIAN = 66 LG_ITALIAN = LG_ITN LG_ITALIAN_SWISS = 67 LG_JAPANESE = 68 LG_KOREAN = 69 LG_KOREAN_JOHAB = 70 LG_LATVIAN = 71 LG_LITHUANIAN = 72 LG_NORWEGIAN = LG_NOR LG_NORWEGIAN_BOKMAL = 73 LG_NORWEGIAN_NYNORSK = 74 LG_POLISH = 75 LG_PORTUGUESE = LG_POR LG_PORTUGUESE_BRAZIL = 76 LG_ROMANIAN = 77 LG_RUSSIAN = 78 LG_SERBIAN_LATIN = 79 LG_SLOVAK = 80 LG_SLOVENIAN = 81 LG_SPANISH = LG_SPA LG_SPANISH_MEXICAN = 82 LG_SPANISH_MODERN = 83 LG_SWEDISH = LG_SWE LG_THAI = 84 LG_TURKISH = 85 LG_UKRANIAN = 86 LG_ASSAMESE = 87 LG_BENGALI = 88 LG_BIHARI = 89 LG_BODO = 90 LG_DOGRI = 91 <i>(continued on next page)</i>
---------------------	--

<b>Language (3)</b>	LG_GUJARATI = 92 LG_HARYANVI = 93 LG_HINDI = 94 LG_KANNADA = 95 LG_KASHMIRI = 96 LG_MALAYALAM = 97 LG_MARATHI = 98 LG_MARWARI = 99 LG_MEGHALAYAN = 100 LG_MIZO = 101 LG_NAGA = 102 LG_ORISSI = 103 LG_PUNJABI = 104 LG_PUSHTU = 105 LG_SERBIAN_CYRILLIC = 106 LG_SIKKIMI = 107 LG_SWEDISH_FINLAND = 108 LG_TAMIL = 109 LG_TELUGU = 110 LG_TRIPURI = 111 LG_URDU = 112 LG_VIETNAMESE = 113
<b>PowerSupply</b>	PS_EXTERNAL = 0 PS_BATTERY = 1
<b>Printer</b>	PR_IMPRINTERTOPBEFORE = 0 PR_IMPRINTERTOPAFTER = 1 PR_IMPRINTERBOTTOMBEFORE = 2 PR_IMPRINTERBOTTOMAFTER = 3 PR_ENDORSERTOPBEFORE = 4 PR_ENDORSERTOPAFTER = 5 PR_ENDORSERBOTTOMBEFORE = 6 PR_ENDORSERBOTTOMAFTER = 7
<b>PrinterMode</b>	PM_SINGLESTRING = 0 PM_MULTISTRING = 1 PM_COMPOUNDSTRING = 2
<b>BarCodeSearchMode</b>	BC_HORZ = 0 BC_VERT = 1 BC_HORZVERT = 2 BC_VERTHORZ = 3
<b>PatchCodeSearchMode</b>	PC_HORZ = 0 PC_VERT = 1 PC_HORZVERT = 2 PC_VERTHORZ = 3
<b>FlashUsed2</b>	FL_NONE = 0 FL_OFF = 1 FL_ON = 2 FL_AUTO = 3 FL_REDEYE = 4
<b>FlipRotation</b>	FR_BOOK = 0 FR_FANFOLD = 1
<b>ImageFilter</b>	IF_NONE = 0 IF_AUTO = 1 IF_LOWPASS = 2 IF_BANDPASS = 3 IF_HIGHPASS = 4 IF_TEXT = IF_BANDPASS IF_FINELINE = IF_HIGHPASS

<b>NoiseFilter</b>	NF_NONE = 0 NF_AUTO = 1 NF_LONEPIXEL = 2 NF_MAJORITRYRULE = 3
<b>OverScan</b>	OV_NONE = 0 OV_AUTO = 1 OV_TOPBOTTOM = 2 OV_LEFTRIGHT = 3 OV_ALL = 4
<b>BarCodeType</b>	BT_3OF9 = 0 BT_2OF5INTERLEAVED = 1 BT_2OF5NONINTERLEAVED = 2 BT_CODE93 = 3 BT_CODE128 = 4 BT_UCC128 = 5 BT_CODABAR = 6 BT_UPCA = 7 BT_UPCE = 8 BT_EAN8 = 9 BT_EAN13 = 10 BT_POSTNET = 11 BT_PDF417 = 12 BT_2OF5INDUSTRIAL = 13 BT_2OF5MATRIX = 14 BT_2OF5DATALOGIC = 15 BT_2OF5IATA = 16 BT_3OF9FULLASCII = 17 BT_CODABARWITHSTARTSTOP = 18
<b>DeskewStatus</b>	DSK_SUCCESS = 0 DSK_REPORTONLY = 1 DSK_FAIL = 2 DSK_DISABLED = 3
<b>PatchCode</b>	PCH_PATCH1 = 0 PCH_PATCH2 = 1 PCH_PATCH3 = 2 PCH_PATCH4 = 3 PCH_PATCH6 = 4 PCH_PATCHT = 5
<b>ExtendedImageInfo</b>	EI_BARCODEX = 0x1200 EI_BARCODEY = 0x1201 EI_BARCODETEXT = 0x1202 EI_BARCODETYPE = 0x1203 EI_DESHADETOP = 0x1204 EI_DESHADELEFT = 0x1205 EI_DESHADEHEIGHT = 0x1206 EI_DESHADEWIDTH = 0x1207 EI_DESHADESIZE = 0x1208 EI_SPECKLESREMOVED = 0x1209 EI_HORZLINEXCOORD = 0x120A EI_HORZLINEYCOORD = 0x120B EI_HORZLINELENGTH = 0x120C EI_HORZLINETHICKNESS = 0x120D EI_VERTLINEXCOORD = 0x120E EI_VERTLINEYCOORD = 0x120F EI_VERTLINELENGTH = 0x1210 EI_VERTLINETHICKNESS = 0x1211 EI_PATCHCODE = 0x1212 EI_ENDORSEDTEXT = 0x1213 EI_FORMCONFIDENCE = 0x1214 (continued on next page)

<b>ExtendedImageInfo (2)</b>	EI_FORMTEMPLATEMATCH = 0x1215 EI_FORMTEMPLATEPAGEMATCH = 0x1216 EI_FORMHORIZDOCOFFSET = 0x1217 EI_FORMVERTDOCOFFSET = 0x1218 EI_BARCODECOUNT = 0x1219 EI_BARCODECONFIDENCE = 0x121A EI_BARCODEROTATION = 0x121B EI_BARCODETEXTLENGTH = 0x121C EI_DESHADECOUNT = 0x121D EI_DESHADEBLACKCOUNTOLD = 0x121E EI_DESHADEBLACKCOUNTNEW = 0x121F EI_DESHADEBLACKRLMIN = 0x1220 EI_DESHADEBLACKRLMAX = 0x1221 EI_DESHADEWHITECOUNTOLD = 0x1222 EI_DESHADEWHITECOUNTNEW = 0x1223 EI_DESHADEWHITERLMIN = 0x1224 EI_DESHADEWHITERLAVE = 0x1225 EI_DESHADEWHITERLMAX = 0x1226 EI_BLACKSPECKLESREMOVED = 0x1227 EI_WHITESPECKLESREMOVED = 0x1228 EI_HORZLINECOUNT = 0x1229 EI_VERTLINECOUNT = 0x122A EI_DESKEWSTATUS = 0x122B EI_SKEWORIGINALANGLE = 0x122C EI_SKEWFINALANGLE = 0x122D EI_SKEWCONFIDENCE = 0x122E EI_SKEWWINDOWX1 = 0x122F EI_SKEWWINDOWY1 = 0x1230 EI_SKEWWINDOWX2 = 0x1231 EI_SKEWWINDOWY2 = 0x1232 EI_SKEWWINDOWX3 = 0x1233 EI_SKEWWINDOWY3 = 0x1234 EI_SKEWWINDOWX4 = 0x1235 EI_SKEWWINDOWY4 = 0x1236 EJ_NONE = 0x0000 EJ_MIDSEPARATOR = 0x0001 EJ_PATCH1 = 0x0002 EJ_PATCH2 = 0x0003 EJ_PATCH3 = 0x0004 EJ_PATCH4 = 0x0005 EJ_PATCH6 = 0x0006 EJ_PATCH7 = 0x0007
<b>NativeMemorFileType</b>	DIB = FF_BMP JPEG = FF_JFIF TIFF = FF_TIFF MULTIPAGE_TIFF = FF_TIFFMULTI
<b>TIFFCompression</b>	NO_COMPRESSION = CMP_NONE TIFF_GROUP31D = CMP_GROUP31DEOL TIFF_GROUP32D = CMP_GROUP32D TIFF_GROUP4 = CMP_GROUP4 TIFF_JPEG = CMP_JPEG
<b>ObjectTwainState</b>	NOT_STARTED = 0 STARTED = 1 OPENED = 2 ACQUISITION = 3
<b>ConType</b>	ON_ARRAY = 3 ON_ENUMERATION = 4 ON_ONEVALUE = 5 ON_RANGE = 6

<b>ItemType</b>	TY_INT8 = 0x0000 TY_INT16 = 0x0001 TY_INT32 = 0x0002 TY_UINT8 = 0x0003 TY_UINT16 = 0x0004 TY_UINT32 = 0x0005 TY_BOOL = 0x0006 TY_FIX32 = 0x0007 TY_FRAME = 0x0008 TY_STR32 = 0x0009 TY_STR64 = 0x000a TY_STR128 = 0x000b TY_STR255 = 0x000c
<b>Country</b>	CY_AFGHANISTAN = 1001 CY_ALGERIA = 213 CY_AMERICANSAMOA = 684 CY_ANDORRA = 033 CY_ANGOLA = 1002 CY_ANGUILLA = 8090 CY_ANTIGUA = 8091 CY_ARGENTINA = 54 CY_ARUBA = 297 CY_ASCENSIONI = 247 CY_AUSTRALIA = 61 CY_AUSTRIA = 43 CY_BAHAMAS = 8092 CY_BAHRAIN = 973 CY_BANGLADESH = 880 CY BARBADOS = 8093 CY_BELGIUM = 32 CY_BELIZE = 501 CY_BENIN = 229 CY_BERMUDA = 8094 CY_BHUTAN = 1003 CY_BOLIVIA = 591 CY_BOTSWANA = 267 CY_BRITAIN = 6 CY_BRITVIRGINIS = 8095 CY_BRAZIL = 55 CY_BRUNEI = 673 CY_BULGARIA = 359 CY_BURKINAFASO = 1004 CY_BURMA = 1005 CY_BURUNDI = 1006 CY_CAMAROON = 237 CY_CANADA = 2 CY_CAPEVERDEIS = 238 CY_CAYMANIS = 8096 CY_CENTRALAFREP = 1007 CY_CHAD = 1008 CY_CHILE = 56 CY_CHINA = 86 CY_CHRISTMASIS = 1009 CY_COCOSIS = 1009 CY_COLOMBIA = 57 CY_COMOROS = 1010 CY_CONGO = 1011 CY_COOKIS = 1012 CY_COSTARICA = 506 CY_CUBA = 005 (continued on next page)

<b>Country (2)</b>	CY_CYPRUS = 357 CY_CZECHOSLOVAKIA = 42 CY_DENMARK = 45 CY_DJIBOUTI = 1013 CY_DOMINICA = 8097 CY_DOMINICANREP = 8098 CY_EASTERIS = 1014 CY_ECUADOR = 593 CY_EGYPT = 20 CY_ELSALVADOR = 503 CY_EQGUINEA = 1015 CY_ETHIOPIA = 251 CY_FALKLANDIS = 1016 CY_FAEROEIS = 298 CY_FIJIISLANDS = 679 CY_FINLAND = 358 CY_FRANCE = 33 CY_FRANTILLES = 596 CY_FRGUIANA = 594 CY_FRPOLYNEISA = 689 CY_FUTANAIS = 1043 CY_GABON = 241 CY_GAMBIA = 220 CY_GERMANY = 49 CY_GHANA = 233 CY_GIBRALTER = 350 CY_GREECE = 30 CY_GREENLAND = 299 CY_GRENADA = 8099 CY_GRENEDINES = 8015 CY_GUADELOUPE = 590 CY_GUAM = 671 CY_GUANTANAMOBAY = 5399 CY_GUATEMALA = 502 CY_GUINEA = 224 CY_GUINEABISSAU = 1017 CY_GUYANA = 592 CY_HAITI = 509 CY_HONDURAS = 504 CY_HONGKONG = 852 CY_HUNGARY = 36 CY_ICELAND = 354 CY_INDIA = 91 CY_INDONESIA = 62 CY_IRAN = 98 CY_IRAQ = 964 CY_IRELAND = 353 CY_ISRAEL = 972 CY_ITALY = 39 CY_IVORYCOAST = 225 CY_JAMAICA = 8010 CY_JAPAN = 81 CY_JORDAN = 962 CY_KENYA = 254 CY_KIRIBATI = 1018 CY_KOREA = 82 CY_KUWAIT = 965 CY_LAOS = 1019 CY_LEBANON = 1020 CY_LIBERIA = 231 <i>(continued on next page)</i>
--------------------	---

<b>Country (3)</b>	CY_LIBYA = 218 CY_LIECHTENSTEIN = 41 CY_LUXENBOURG = 352 CY_MACAO = 853 CY_MADAGASCAR = 1021 CY_MALAWI = 265 CY_MALAYSIA = 60 CY_MALDIVES = 960 CY_MALI = 1022 CY_MALTA = 356 CY_MARSHALLIS = 692 CY_MAURITANIA = 1023 CY_MAURITIUS = 230 CY_MEXICO = 3 CY_MICRONESIA = 691 CY_MIQUELON = 508 CY_MONACO = 33 CY_MONGOLIA = 1024 CY_MONTSERRAT = 8011 CY_MOROCCO = 212 CY_MOZAMBIQUE = 1025 CY_NAMIBIA = 264 CY_NAURU = 1026 CY_NEPAL = 977 CY_NETHERLANDS = 31 CY_NETHANTILLES = 599 CY_NEVIS = 8012 CY_NEWCALEDONIA = 687 CY_NEWZEALAND = 64 CY_NICARAGUA = 505 CY_NIGER = 227 CY_NIGERIA = 234 CY_NIUE = 1027 CY_NORFOLKI = 1028 CY_NORWAY = 47 CY_OMAN = 968 CY_PAKISTAN = 92 CY_PALAU = 1029 CY_PANAMA = 507 CY_PARAGUAY = 595 CY_PERU = 51 CY_PHILLIPPINES = 63 CY_PITCAIRNIS = 1030 CY_PNEWGUINEA = 675 CY_POLAND = 48 CY_PORTUGAL = 351 CY_QATAR = 974 CY_REUNIONI = 1031 CY_ROMANIA = 40 CY_RWANDA = 250 CY_SAIPAN = 670 CY_SANMARINO = 39 CY_SAOTOME = 1033 CY_SAUDIARABIA = 966 CY_SENEGAL = 221 CY_SEYCHELLESIS = 1034 CY_SIERRALEONE = 1035 CY_SINGAPORE = 65 CY_SOLOMONIS = 1036 CY_SOMALI = 1037 <i>(continued on next page)</i>
--------------------	---

<b>Country (4)</b>	CY_SOUTHAFRICA = 27 CY_SPAIN = 34 CY_SRILANKA = 94 CY_STHELENA = 1032 CY_STKITTS = 8013 CY_STLUCIA = 8014 CY_STPIERRE = 508 CY_STVINCENT = 8015 CY_SUDAN = 1038 CY_SURINAME = 597 CY_SWAZILAND = 268 CY_SWEDEN = 46 CY_SWITZERLAND = 41 CY_SYRIA = 1039 CY_TAIWAN = 886 CY_TANZANIA = 255 CY_THAILAND = 66 CY_TOBAGO = 8016 CY_TOGO = 228 CY_TONGAIS = 676 CY_TRINIDAD = 8016 CY_TUNISIA = 216 CY_TURKEY = 90 CY_TURKSCAICOS = 8017 CY_TUVALU = 1040 CY_UGANDA = 256 CY_USSR = 7 CY_UAEMIRATES = 971 CY_UNITEDKINGDOM = 44 CY_USA = 1 CY_URUGUAY = 598 CY_VANUATU = 1041 CY_VATICANCITY = 39 CY_VENEZUELA = 58 CY_WAKE = 1042 CY_WALLISIS = 1043 CY_WESTERNSAHARA = 1044 CY_WESTERNSAMOA = 1045 CY_YEMEN = 1046 CY_YUGOSLAVIA = 38 CY_ZAIRE = 243 CY_ZAMBIA = 260 CY_ZIMBABWE = 263 CY_ALBANIA = 355 CY_ARMENIA = 374 CY_AZERBAIJAN = 994 CY_BELARUS = 375 CY_BOSNIAHERZGO = 387 CY_CAMBODIA = 855 CY_CROATIA = 385 CY_CZECHREPUBLIC = 420 CY_DIEGOGARCIA = 246 CY_ERITREA = 291 CY_ESTONIA = 372 CY_GEORGIA = 995 CY_LATVIA = 371 CY_LESOTHO = 266 CY_LITHUANIA = 370 CY_MACEDONIA = 389 CY_MAYOTTEIS = 269 <i>(continued on next page)</i>
--------------------	---

<b>Country (5)</b>	CY_MOLDOVA = 373 CY_MYANMAR = 95 CY_NORTHKOREA = 850 CY_PUERTORICO = 787 CY_RUSSIA = 7 CY_SERBIA = 381 CY_SLOVAKIA = 421 CY_SLOVENIA = 386 CY_SOUTHKOREA = 82 CY_UKRAINE = 380 CY_USVIRGINIS = 340 CY_VIETNAM = 84
<b>ObjectTwainError</b>	OBJTWAIN_NO_ERROR = 0x0200  TWAIN_ERROR = 0x0300  TWAIN_CANCEL = 0x0301 TWAIN_END_OF_LIST = 0x0302 TWAIN_FAILURE = 0x0303 TWAIN_PARTIAL_FAILURE = 0x0304 TWAIN_UI_NOT_SUPPRESSED = 0x0305 TWAIN_LOW_MEMORY = 0x0401 TWAIN_BUMMER = 0x0402 TWAIN_NO_DS = 0x0403 TWAIN_MAX_CONNECTIONS = 0x0404 TWAIN_OPERATION_ERROR = 0x0405 TWAIN_BAD_CAP = 0x0406 TWAIN_BAD_PROTOCOL = 0x0407 TWAIN_BAD_VALUE = 0x0408 TWAIN_SEQ_ERROR = 0x0409 TWAIN_BAD_DEST = 0x040A TWAIN_CAP_UNSUPPORTED = 0x040B TWAIN_CAP_BAD_OPERATION = 0x040C TWAIN_CAP_SEQ_ERROR = 0x040D  OBJTWAIN_INIT_ERROR = 0x0501 OBJTWAIN_OPERATION_INVALID_AT_THIS_TIME = 0x0502 OBJTWAIN_DIB_FILE_OPEN_ERROR = 0x0503 OBJTWAIN_BAD_DIB_DATA = 0x0504 OBJTWAIN_DISK_WRITE_ERROR = 0x0505 OBJTWAIN_NOT_24BPP_IMAGE = 0x0506 OBJTWAIN_JPEG_COMPRESSION_ERROR = 0x0507 OBJTWAIN_UNSUPPORTED_FILE_FORMAT = 0x0508 OBJTWAIN_UNKNOWN_TRANSFER_ERROR = 0x0509  WIN32_EXCEPTION_BREAKPOINT = 0x0601 WIN32_EXCEPTION_SINGLE_STEP = 0x0602 WIN32_EXCEPTION_GUARD_PAGE = 0x0603 WIN32_EXCEPTION_ACCESS_VIOLATION = 0x0604 WIN32_EXCEPTION_DATATYPE_MISALIGNMENT = 0x0605 WIN32_EXCEPTION_ARRAY_BOUNDS_EXCEEDED = 0x0606 WIN32_EXCEPTION_FLT_DENORMAL_OPERAND = 0x0607 WIN32_EXCEPTION_FLT_DIVIDE_BY_ZERO = 0x0608 WIN32_EXCEPTION_FLT_INEXACT_RESULT = 0x0609 WIN32_EXCEPTION_FLT_INVALID_OPERATION = 0x060A WIN32_EXCEPTION_FLT_OVERFLOW = 0x060B WIN32_EXCEPTION_FLT_STACK_CHECK = 0x060C WIN32_EXCEPTION_FLT_UNDERFLOW = 0x060D WIN32_EXCEPTION_INT_DIVIDE_BY_ZERO = 0x060F WIN32_EXCEPTION_INT_OVERFLOW = 0x0610 <i>(continued on next page)</i>

<b>ObjectTwainError (2)</b>	WIN32_EXCEPTION_PRIV_INSTRUCTION = 0x0611 WIN32_EXCEPTION_IN_PAGE_ERROR = 0x0612 WIN32_EXCEPTION_ILLEGAL_INSTRUCTION = 0x0613 WIN32_EXCEPTION_NONCONTINUABLE = 0x0614 WIN32_EXCEPTION_STACK_OVERFLOW = 0x0615 WIN32_EXCEPTION_INVALID_DISPOSITION = 0x0616 WIN32_STATUS_NO_MEMORY = 0x0617 WIN32_UNKNOWN_EXCEPTION = 0x0618 WIN32_EXCEPTION_NONCONTINUABLE_EXCEPTION = 0x0619  UNKNOWN_ERROR = 0x8000
-----------------------------	---