



Impact ActiveX Reference Guide

Publication # 843-0119 Rev 11.8

Disclaimer

Datalogic Automation, Inc. makes no representations or warranties for merchantability or fitness for any particular purpose, regarding Datalogic's software or hardware. Datalogic Automation, Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this publication or its contents.

Datalogic Automation, Inc. reserves the right to revise this publication from time to time and to make changes in the content hereof without obligation to notify any person of such revision or changes.

Under the copyright laws, neither this publication nor the software may be copied, photocopied, reproduced, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of Datalogic Automation, Inc.

Telephone: 952-996-9500
Facsimile: 952-996-9501
Web site: <http://www.datalogic.com>
E-Mail: mvsupport@datalogic.com

Datalogic and the Datalogic logo are registered trademarks of Datalogic S.p.A. in many countries, including the U.S.A. and the E.U.

All brand and product names mentioned herein are for identification purposes only and may be trademarks or registered trademarks of their respective owners.

Impact, Vision Program Manager, VPM, Control Panel Manager, CPM, and Pinpoint Pattern Find are trademarks of Datalogic Automation, Inc.

Copyright © 2002-2015 Datalogic Automation, Inc. All Rights Reserved

Technical Support

If you have technical questions about the operation of any Datalogic product, contact your distributor or Datalogic. Please have the following information available before you call:

- The model and serial number of the Datalogic camera, located on the unit. The version number of the Impact software you are running, found in the title bar of the software.
- The type and version number of the operating system software you are running on the client computer.

TABLE OF CONTENTS

Introduction

Manual Contents	Intro-1
Overview	Intro-2
Impact ActiveX Install	Intro-3
Interface	Intro-3
Impact Documentation	Intro-4
Vision Device Security	Intro-4
Glossary	Intro-5

Chapter 1: Discovery Interface

Discovery Interface Introduction	1-1
Discovery Interface Methods	1-1
Find	1-1
Check Connection	1-1
Connect	1-1
Close Connection	1-2
DLL Version	1-2
Code Examples	1-2

Chapter 2: Information Interface

Information Interface Introduction	2-1
Information Interface Methods	2-1
Get System Log	2-1
Clear System Log	2-1
Get Details	2-1
Get Camera Calibration	2-2
Set Camera Calibration	2-2
Sync Time to PC	2-2
Code Examples	2-2

Chapter 3: Programs Interface

Programs Interface Introduction	3-1
Programs Interface Methods	3-1
Delete	3-1
Get Files	3-1
Get Loaded Files	3-1
Load on Startup	3-2
Load	3-2
Unload	3-2
Read	3-2
Write	3-3
Code Examples	3-3

Chapter 4: Operate Interface

Operate Interface Introduction	4-1
Operate Interface Methods	4-1
Auto White Balance	4-1
Get File Camera	4-1
Set File Camera	4-1
Get Online	4-2
Set Online	4-2
Online On Startup	4-2
Run Task	4-2
Run Tool	4-2
Snap Image	4-3
Train Tool	4-3
Trigger	4-3

Read Digital Input	4-4
Write Digital Outputs	4-4
Reading and Writing Tool Ports	4-5
Get Image Port	4-6
Get Real Port	4-6
Set Real Port	4-7
Get Integer Port	4-7
Set Integer Port	4-7
Get Boolean Port	4-7
Set Boolean Port	4-7
Get String Port	4-8
Set String Port	4-8
Code Examples	4-8
Chapter 5: Device Events Interface	
Device Events Interface Introduction	5-1
Device Events Interface Methods	5-1
Impact Event	5-1
Set Device Handle	5-2
Set Subscription	5-2
Code Examples	5-2
Chapter 6: Configuration Utility Interface	
Configuration Utility Interface Introduction	6-1
Configuration Utility Interface Properties	6-1
Connection Timeout	6-1
Command Timeout	6-1
Next Command Timeout	6-2
Event Description	6-2
Error Description	6-2
Code Examples	6-2
Chapter 7: File Data Interface	
File Data Interface Methods Introduction	7-1
File Data Interface Methods	7-1
Code Examples	7-2
Chapter 8: Device Details Interface	
Device Details Interface Methods Introduction	8-1
Device Details Interface Methods	8-1
Code Examples	8-2
Chapter 9: PngImage Interface	
PngImage Interface Introduction	9-1
PngImage Interface Methods	9-1
Code Examples	9-1
Chapter 10: Appendix	
Appendix Introduction	10-1
Error Codes	10-1
Code Examples	10-3
Using the Emulator	10-4
Multiple Emulators	10-4
No Network Connection	10-4
Multiple Vision Devices	10-5

Introduction

This manual documents the Impact ActiveX interface that allows programmers to access data and events on Datalogic's line of cameras and processors.

Using ActiveX, a programmer has the ability to monitor and operate a Datalogic device, including the ability to read and write tool and task inputs and outputs, load and unload programs, trigger the device to snap an image, and respond to device events.

This manual shows you how to implement the Impact ActiveX interface.

NOTE: This manual does not provide detailed operation about Datalogic cameras and processors. It assumes you have some knowledge of how vision programs and Impact tools and tasks are structured, how to access tool and task data, and that you understand Impact terminology.

For more details about using Impact Vision Program Manager (VPM) and Datalogic cameras, refer to the Datalogic Hardware Guides and the Impact Software Reference Guide.

NOTE: ActiveX cannot be used on the Datalogic model A30, T4x-Series, or P-Series Smart Cameras.

Manual Contents

This document consists of the following chapters:

- **Introduction**
This introduction describes, in general terms, what the ActiveX interface does and how to use it.
- **Chapter 1: Discovery**
This chapter explains, in detail, how to use the interface to find Datalogic devices on the network and establish communications with it.
- **Chapter 2: Information**
This chapter describes the static, descriptive information available for the Datalogic device.
- **Chapter 3: Programs**
This chapter explains how to use the ActiveX interface to manipulate and transfer Impact vision programs.
- **Chapter 4: Operate**
This chapter explains how to use the ActiveX interface to control and monitor inspection operations on the Datalogic camera.
- **Chapter 5: Device Events**
This chapter explains how to use the ActiveX interface to control and monitor inspection operations on the Datalogic camera.
- **Chapter 6: Configuration Utility**
This chapter explains how to get access to the timers used to judge connection health and helper functions that convert event and error numbers to strings.
- **Chapter 7: File Data**
This chapter explains how to access data about a set of files and how the data is presented as a set of properties.

- **Chapter 8: Device Details Interface**

This chapter explains the Device Details interface that contains information about one or more Datalogic Cameras.

- **Chapter 9: PngImage Interface**

This chapter explains how the PngImage interface encapsulates an IMAGE in PNG format.

- **Chapter 10: Appendix**

This chapter explains additional information about several aspects of implementing the Impact ActiveX interface, including error code handling.

Overview

Impact ActiveX allows you to directly integrate Datalogic machine vision device operations into your manufacturing control systems. The control system can load Impact vision programs, put the Datalogic camera online, monitor inspection data, and then act on inspection results by controlling pass and fail outputs. The interaction can include archiving images and analyzing inspection results for traceability and quality purposes.

To initially configure the Datalogic camera and create Impact vision programs and control panels, you need to install Impact software, including Vision Program Manager (VPM) and Control Panel Manager (CPM) on a host computer. If you do not intend to create or modify vision programs, or test your application using the Impact Emulator, you do not need to install the Impact software (VPM and CPM) on the host computer. Refer to the Impact Software Reference Guide (843-0093) for the client computer's requirements.

The Impact Active X Application Programming Interface (API) lets you write a program (for example in Visual Basic) to access vision device data by using the necessary function calls. That data can then be sent directly from a Datalogic camera to other devices, such as a Quality Control monitoring system.

Several sample programs are also provided to help you use the APIs. To access these sample programs, you need to install the Impact ActiveX components on the host computer. See page 3.

To access the Impact ActiveX API, you must install Impact Software Release 8.3.0 or greater on the Datalogic vision device. Refer to the Impact Reference Guide for Impact Software installation instructions. This installs and configures the DPWS server so that the device can communicate with other ActiveX devices on the network.

In VPM, there is a check box on the General System Object Setup - Communication panel (Enable OPC/ActiveX Server). When the server is enabled, the DPWS server on the Datalogic device starts automatically when the device boots.

You must also include the Impact dynamic link library (IMPACT.dll) in your program or project. Refer to the applicable programming language document for more details. (See "C++" on page 3 for C++ implementation.)

IMPACT.dll uses the FreeImage open source image library. (See <http://freeimage.sourceforge.net> for details.) FreeImage is used under the FreeImage Public License version 1.0. See the license in "FreeImage Public License - Version 1.0.txt" in the ActiveX\Documentation folder.

Impact Control Panel Manager (CPM) is still the best tool to use when you want to easily create an operator interface to a vision program. With CPM, you can quickly build a control panel, then drag and drop data connections from the Datalogic device. The operator can then monitor and control the inspection using the control panel.

Impact ActiveX Install

The Impact ActiveX install provides the Impact library (IMPACT.dll) that you need to include in your program or project. This library contains the functions you need to access the Datalogic vision device.

To install Impact ActiveX:

1. You may need to turn off automatic virus checking during the install if it causes installation problems.
2. You must have administrative privileges to install Impact software on a Windows operating system.
3. You can download the ActiveX install file from Datalogic's web site (www.datalogic.com). When the download is complete, locate the file named IMPACTActiveXSetup_n_n_n.exe (where "n" is a version number). Double-click the file.
4. The installation program should start.
5. Follow the on-screen instructions.
6. Refer to the applicable programming language document for more details about file locations and using the Impact ActiveX API with that language.

Interface

NOTE: Explaining how to use a programming language to create a program is beyond the scope of this manual. This manual describes the commands, properties, methods, and events available in the Impact ActiveX interface. Please refer to other Getting Started manuals for details on how to implement them in various programming languages such as Visual Basic.

The ActiveX interface is divided into four essential parts: Discovery, Information, Programs, and Operate. These parts are explained in detail in the other chapters of this manual.

All methods except some in the Discovery interface include a parameter (IDevHnd) that identifies the particular Datalogic device to which the method is being directed. This parameter is created by a log in operation whether security is enabled on the device or not. See "Connect" on page 1 for details.

IMPORTANT: Impact ActiveX is not designed for multi-threaded access. It should only be accessed in a single-threaded manner.

Notations

The data types used in each programming language are somewhat different. Each parameter in an Impact ActiveX method call is prefixed by a character with the following meanings:

- i is an Integer
- s is a String
- l is a Long
- f is a Floating point number
- p is an object or pointer

Error Codes

If a method fails, an error code is returned describing the reason for the failure. Error codes are common across all methods and interfaces. A return code of zero indicates that the method was successful.

Error codes are defined by the `ImpactLib.ImpactErrorCode` enumeration. There are more than eighty possible error responses with all calls giving multiple options. The common errors are listed here. See “Error Codes” on page 1 for a more complete list of errors.

- `ImpactOK`: Success
- `ImpactCommunicationError`: Failed to communicate with the camera. The camera was found but communication failed.
- `ImpactSmartCameraNotFound`: Failed to find the camera. The camera may be unplugged or the power turned off.
- `ImpactBadHandle`: The `IDevHnd` parameter is invalid. The ActiveX interface is not currently connected to a device with that handle or the device no longer exists on the network.
- `ImpactBadPort`: The program, task, tool, port, or subport parameter is invalid. The object being addressed does not have the parameter indicated in the method call.
- `ImpactBadPortType`: There is a mismatch between the call type and the port type. For example, trying to read an Integer port value with a String method call.

Using Impact ActiveX in non-English language environments

Impact text can be translated from English to another language (the target language) so that text, messages, program names, task names, and tool names displayed to the operator in VPM and CPM are displayed in the target language. This is done by translating the Impact language resource file into the target language. (Refer to Impact translation manuals for details).

It is, of course, possible to develop your ActiveX program on a computer configured to use a non-English language. However, no matter what language you are using on the development system, when you create an Impact ActiveX method call, the port and subport name parameters used in the call must be in English.

For example, if you use the call,

```
RunTool( IDevHnd, sProgram, sTask, sTool )
```

the parameters `sProgram`, `sTask`, and `sTool` can be in the target language.

If you use the call,

```
GetRealPort( IDevHnd, sProgram, sTask, sTool, sPort, sSubport, fData )
```

the parameters `sProgram`, `sTask`, and `sTool` can be in the target language, but the parameters `sPort`, and `sSubport` must be in English.

To view VPM in English, start the VPM program using the language command line switch (`-language en`). For details, refer to the Impact Reference Guide (Publication # 843-0093).

Impact Documentation

This manual does not provide detailed operation about Datalogic machine vision devices. For more details on operations and other settings, refer to the Datalogic Hardware Guides and the Impact Software Reference Guide.

Vision Device Security

If security is enabled on the device, you must provide a valid User name and Password when you connect to the device. (See “Connect” on page 1.) The User name and Password you provide will determine the level of

access you have to the device and the method calls you can make. The User Level required to complete a method call varies for each call. In general, the following rules apply:

User Level	Access Level
Programmer	All calls Including file transfers (e.g. "Write" on page 3)
Controller	Device set up calls Set Data calls (e.g. "Auto White Balance" on page 1) Read Data calls Cannot transfer files
Operator	Set Data calls (e.g. "Set Boolean Port" on page 7) Read Data calls
Monitor	Read Data calls only (e.g. "Get Integer Port" on page 7)

For more details about device security and user management, refer to the Impact Reference Guide.

Glossary

Many of the terms used in this document are explained in more detail in the Impact Reference Guide. However, some of the terms that are specific to the Impact ActiveX software are briefly defined here.

.NETAn software operating environment for later Windows operating systems.

ActiveXAn extension of COM that provides standard capabilities for an object to be used in "automation" environments such as Visual Basic.

COMComponent Object Model - A Microsoft technology for providing language-neutral software resources to application programs.

DAData Access - The core OPC standard for communicating data to and from field devices.

DPWSDevices Profile for Web Services - A standard that combines elements of the Web Services standards to provide support for small stand-alone devices

DLLDynamic-link Library - A binary file that contains a related set of interfaces and methods. It is loaded on an as-needed basis and may be shared between multiple, independent applications.

GUIGraphical User Interface

Marshalling..A software mechanism where data types are converted so as to be useful to another layer or program that requires a different format. For example, big endian versus small endian representation of integers.

Middleware..Software that allows application programs to access other computational resources inside a computer and over networks.

OLEObject Linking and Embedding. An older term that described Microsoft-specific tech-

nologies for making inter-process and inter-computer procedure calls. The original "OLE" was replaced with COM and DCOM

OPC Originally "OLE for Process Control" but now a non-specific TLA denoting a factory automation technology

RPC Remote Procedure Call - Where one computer causes another to execute a task. Data may be supplied or returned.

UDP User Datagram Protocol - Computer applications can send messages to other hosts on a network without requiring prior communications to set up special transmission channels or data paths.

VPM Vision Program Manager

XML Extensible Mark-up Language - A standard for describing information in text files.

Discovery Interface

Discovery Interface Introduction

The Discovery interface is used to find and connect to Datalogic cameras.

Discovery Interface Methods

Find

Find (pDevices)

Parameters:

pdevices: An ImpactLib.DeviceDetails interface pointer

This method looks for Datalogic cameras on the network. It returns an interface with a count of devices found and the IP address (including the port number for Emulators), name, software revision, and description for each found Datalogic camera.

NOTE: Only devices with Impact software version 8.3.0 or later installed will appear in the found device list.

ActiveX cannot be used on the Datalogic model A30, T4x-Series, or P-Series Smart Cameras.

Check Connection

CheckConnection(sIpAddr)

Parameters:

sIpAddr A String containing the IP address of the target device

This method determines if the interface connection will work to the selected device by IP address. It is possible that a device can respond to the UDP identity queries and not be available for the needed TCP/IP connections.

Connect

Connect(sIpAddr, sUserName, sPassword, lDevHnd)Parameters:

Parameters:

sIpAddr A String containing the IP address of the target device

sUserName..... The camera's user name (leave blank if not needed)

sPassword..... The password for the username (leave blank if not needed)

IDevHnd..... The returned device handle or zero if the call failed

This method establishes a connection to a Datalogic device using the IP address, user name, and password. The user name and password can be passed as blank if the device does not have security enabled. If security is not enabled, all connections to the device are made at the Administrator access level.

The returned parameter "IDevHnd" is used in other methods and interfaces. It identifies the Datalogic device for the method and the privilege level of the connection.

Close Connection

CloseConnection(IDevHnd)

Parameters:

IDevHnd..... The device handle for the connection to close

This method releases any data used on the vision device and host and logs the user out from the Datalogic device, if needed. The handle is invalid for use after this call.

It is important to call this method when you are done using the Datalogic camera in order to terminate the event subscription for the connection. If you do not, the Datalogic device will continue to attempt to establish a connection in order to send events to the client.

DLL Version

DLLVersion(strVersion)

Parameters:

strVersion A string containing the IMPACT.DLL version number

This method returns the version number of IMPACT.DLL that is registered with Windows. It can help diagnose problems with programs that may occur if there are multiple versions of IMPACT.DLL installed on the computer.

The returned string is in major/minor/subminor/build format. For example "8.3.0.10" indicates that major version 8, minor version 3, subminor version 0, and build 10 is installed. This reflects the file version as displayed in the file properties dialog available in Windows.

Code Examples

Here is a simple example of how to call methods in the Discovery Interface.

Visual Basic

```
Sub ShowDiscovery()
    Dim disc As ImpactLib.Discovery
    Dim lRes As Long
    Dim uCnt As Long
    Dim vDevices As ImpactLib.DeviceDetails
    Dim lDevHnd As Long

    disc = New ImpactLib.Discovery
```

```
lRes = disc.Find( vDevices )  
lRes = disc.CheckConnection( "ipaddr" )  
lRes = disc.Connect( "192.168.0.1", "User", "Pass", lDevHnd )  
lRes = disc.CloseConnection( lDevHnd )  
End Sub
```


Information Interface

Information Interface Introduction

The Information interface gets static and near static descriptive information from Datalogic cameras.

Information Interface Methods

Get System Log

GetSysLog(IDevHnd, ILines, sStr)

Parameters:

IDevHnd..... The device handle for the connection

ILines The number of system log lines to return

sStr A String containing the requested lines

GetSysLog returns the system log as a string. It will return ILines as a maximum or all of the lines if ILines is zero. The individual lines of the log are joined together with Carriage Return - Line Feed characters between them. If the log is empty, an error is returned.

Clear System Log

ClearSysLog(IDevHnd)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

This method clears the system log.

Get Details

GetDetails(IDevHnd, pDetails)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

pDetails An ImpactLib.DeviceDetails object

This method retrieves static descriptive information from the device. This information includes the amount of free memory, whether the device is to go online at startup, the software revision number, and the device's description. See "Device Details Interface Methods" on page 8-1 for more details.

Get Camera Calibration

GetCameraCal(IDevHnd, sStr)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

sStr A String containing the contents of the calibration file

This method reads the device's CameraCalibration.xml file, which contains the camera calibration information, and returns it as a string. This lets you read the calibration file as part of product and inspection program changes. Use the SetCameraCal method to write any changes to the camera.

Set Camera Calibration

SetCameraCal(IDevHnd, sStr)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

sStr A String that contains the new contents for the calibration file.

This method writes the device's CameraCalibration.xml file as a string. This file contains the camera calibration information. This lets you modify and write the calibration file as part of product and inspection program changes. Use the GetCameraCal method to read the information from the camera.

Sync Time to PC

SyncTimeToPC(IDevHnd)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

This method reads the current time and date from the host PC then sets that time and date on the Datalogic device.

Code Examples

Here are simple examples of how to call methods in the Information Interface.

Visual Basic

```
Sub ShowInfo()
    Dim info As ImpactLib.Info
    Dim lRes As Long
    Dim lLines As Long
    Dim lDevHnd As Long
    Dim sStr As String
    Dim pDetails As ImpactLib.DeviceDetails

    info = New ImpactLib.Info
    pDetails = New ImpactLib.DeviceDetails
    sStr = ""
```

```
lRes = info.GetSysLog( lDevHnd, 10, sStr )  
lRes = info.ClearSysLog( lDevHnd )  
lRes = info.SyncTimeToPC( lDevHnd )  
lRes = info.GetDetails( lDevHnd, pDetails )  
lRes = info.GetCameraCal( lDevHnd, sStr )  
lRes = info.SetCameraCal( lDevHnd, sStr )  
End Sub
```


Programs Interface

Programs Interface Introduction

The Programs interface works with vision program files on the device. This includes copying, deleting, loading, and unloading files.

All program file name parameters must include the extension “.vp” which is the extension that indicates it is a vision program.

Programs Interface Methods

Delete

Delete(IDevHnd, sFileName)

Parameters:

IDevHnd..... The Datalogic camera’s connection handle
sFileName The file name to delete, as a String

This will delete the indicated file from storage on the Datalogic device. If the filename does not include the “.vp” extension, the method will return an error.

Get Files

GetFiles(IDevHnd, pFiles)

Parameters:

IDevHnd..... The Datalogic camera’s connection handle
pFiles..... An ImpactLib.FileData object

This method returns an interface that describes the file name, program name, file-loaded status, and load-on-start status for each vision program (.vp) file on the Datalogic device. See “File Data Interface Methods” on page 7-1 for more details about this interface.

Get Loaded Files

GetLoaded(IDevHnd, pFiles)

Parameters:

IDevHnd..... The Datalogic camera’s connection handle

pFiles..... An ImpactLib.FileData object

This method returns an interface that describes the file name and program name for each vision program (.vp) file that is currently loaded on the Datalogic device. See “File Data Interface Methods” on page 7-1 for more details about this interface.

Load on Startup

LoadOnStartup(IDevHnd, sFileName, lLoad)

Parameters:

IDevHnd..... The Datalogic camera’s connection handle

sFileName The name of the file to set

lLoad Whether the file should be loaded on startup (1 is True) or not (0 is False)

This method sets the Load On Startup status value for the indicated file to True or False.

Load

Load(IDevHnd, sFileName)

Parameters:

IDevHnd..... The Datalogic camera’s connection handle

sFileName The name of the file to load

This method loads the file named filename. If the file is already loaded, or a file with the same program name is already loaded, an error code is returned.

Unload

Unload(IDevHnd, sProgramName)

Parameters:

IDevHnd..... The Datalogic camera’s connection handle

sProgramName..... The name of the program to unload

This method unloads the program named ProgramName. If that program is not loaded, an error code is returned.

Read

Read(IDevHnd, sImpactFileName, sLocalFileName)

Parameters:

IDevHnd..... The Datalogic camera’s connection handle

sImpactFileName ... The name of the vision program (.vp) file on the Datalogic device

sLocalFileName The file name to save

This method reads the file named ImpactFileName from the Datalogic device and saves it to the file named LocalFileName on the host computer. The file format is not selected or affected by this transfer.

Write

Write(lDevHnd, sImpactFileName, sLocalFileName)

Parameters:

lDevHnd..... The Datalogic camera's connection handle

sLocalFileName..... The program's saved file name

sImpactFileName... The name of the vision program (.vp) file on the Datalogic device

This method is the opposite of the Read method. It reads the file named LocalFileName from the host computer and saves it to the file named ImpactFileName on the Datalogic device. The file format is not selected or affected by this transfer.

Code Examples

Here is a simple example of how to call methods in the Programs Interface.

```
Sub ShowPrograms()  
    Dim prog As ImpactLib.Programs  
    Dim lRes As Long  
    Dim lDevHnd As Long  
    Dim pFiles As ImpactLib.FileData  
  
    prog = New ImpactLib.Programs  
    pFiles = New ImpactLib.FileData  
  
    lRes = prog.GetFiles( lDevHnd, pFiles )  
    lRes = prog.GetLoaded( lDevHnd, pFiles )  
    lRes = prog.LoadOnStartup( lDevHnd, "fileName.vp", True )  
    lRes = prog.Load( lDevHnd, "fileName.vp" )  
    lRes = prog.Unload( lDevHnd, "progName" )  
    lRes = prog.Read( lDevHnd, "fileName.vp","localFileName.vp" )  
    lRes = prog.Write( lDevHnd,"localFileName.vp","fileName.vp" )  
End Sub
```


Operate Interface

Operate Interface Introduction

Using the Operate interface, you can control and monitor inspection operations on the Datalogic camera. This includes modifying camera parameters, setting the camera on and offline, running tasks, running and training tools, snapping images, and reading and writing tool inputs and outputs.

Operate Interface Methods

Auto White Balance

AutoWhiteBalance(IDevHnd)

Parameters:

IDevHnd..... The device handle for the connection

This method performs the automatic white balance operation on the Datalogic camera.

Get File Camera

GetFileCamera(IDevHnd, IFileCam)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

IFileCam A Long value indicating whether the mode is currently enabled (1 is True and 0 is False)

This method returns the camera's current File Camera Mode.

Set File Camera

SetFileCamera(IDevHnd, IFileCam)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

IFileCam A Long value indicating whether File Camera mode is to be enabled (1 is True and 0 is False)

This method sets the camera's File Camera Mode.

Get Online

GetOnline(IDevHnd, IIsOnline)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

IIsOnline..... A Long value indicating whether the camera is currently online (1 is online and 0 is offline)

This method returns the camera's current online status.

Set Online

SetOnline(IDevHnd, IIsOnline)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

IIsOnline..... A Long value indicating whether the camera is to be put online or offline (1 is online and 0 is offline)

This method sets the camera's current online status.

Online On Startup

OnlineOnStartup(IDevHnd, IGoOnline)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

IGoOnline..... A Long value indicating whether the camera is to go online when it starts (1 is True, 0 is False)

This method sets whether the camera is to go online when it starts. Use the GetDetails method to read the current setting. See "Get Details" on page 2-1 for more details.

Run Task

RunTask(IDevHnd, sProgram, sTask)

IDevHnd..... The Datalogic camera's connection handle

sProgram The name of the program

sTask The name of the task

This method runs the named task within the named program.

Run Tool

RunTool(IDevHnd, sProgram, sTask, sTool)

Parameters:

lDevHnd..... The Datalogic camera's connection handle
sProgram The name of the program
sTask The name of the task
sTool..... The name of the tool

This method runs the named tool in the named program and task.

Snap Image

SnapImage(lDevHnd, pPngImage)

Parameters:

lDevHnd..... The Datalogic camera's connection handle
pPngImage The new image as an ImpactLib.pngImage

This method snaps an image in the camera and returns it in the pngImage interface. This does not cause an ImageIn event in the camera so no tasks in the vision program run and no inspection processing is done. See "PngImage Interface Methods" on page 9-1 for more details about the pngImage interface.

Train Tool

TrainTool(lDevHnd, sProgram, sTask, sTool)

Parameters:

lDevHnd..... The Datalogic camera's connection handle
sProgram The name of the program
sTask The name of the task
sTool..... The name of the tool

This method trains the named tool in the named program and task.

Trigger

Trigger(lDevHnd, sImageID)

Parameters:

lDevHnd..... The Datalogic camera's connection handle
sImageID..... The image ID of the new image

This method produces a software trigger in the Datalogic camera. The camera will capture an image and cause an ImageInEvent. The Image ID is returned. Refer to Trigger ID in the Impact Software Reference Guide for more details about the Image ID parameter.

Read Digital Input

ReadDigitalInputs(IDevHnd, IDi)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

IDi A Long value mask of the digital inputs

This method returns a value that is a mask corresponding to the status of all the digital inputs. Each bit of the mask corresponds to one input. For example, a value of one (1) indicates that input one is on. A value of two (2) indicates that input two is on. The number of inputs is camera dependent. Refer to the Discrete Input tool in the Impact Software Reference Guide for more details about the IDi parameter.

Write Digital Outputs

WriteDigitalOutputs(IDevHnd, IDo, IMask)

Parameters:

IDevHnd..... The Datalogic camera's connection handle

IDo..... On and Off states for outputs as a bit mask

IMask..... The Outputs to have their states changed as a bit mask

This method sets the camera's digital outputs based on the value of the IMask parameter. To control the status of an output, the corresponding bit for that output in IMask must be one (1). The on or off status of the outputs that have the IMask bit set is determined by the value of the IDo mask parameter. If the output's IMask bit is one and the bit in IDo is one (1), the output will be turned on. If the bit in IDo is zero (0), the output will be turned off. See the following table for more details.

Integer to Binary

The IMask and IDo parameters are Long values, but the Output tool uses the equivalent bits to set the Datalogic camera's outputs. The values convert like this:

Output Number	6	5	4	3	2	1
Binary Value	2^5	2^4	2^3	2^2	2^1	2^0
Parameter Value	32	16	8	4	2	1

To select a single output, enter the corresponding parameter value. To combine multiple outputs, enter the sum of the parameter values for those outputs. For example,

32 selects output number 6

17 (the sum of 16 and 1) selects outputs 5 and 1

36 (the sum of 32 and 4) selects outputs 6 and 3

63 (the sum of all the integer values) selects all 6 outputs.

NOTE: The number of outputs is camera dependent. Not all cameras have 6 outputs.

IDo

If an output is enabled by its corresponding bit in the IMask value, the IDo parameter determines whether the tool turns an output on or off. If a bit in the IDo parameter is 1, the corresponding output is turned on; if the bit is 0 (zero), that output is turned off.

IMask

An output is enabled when the corresponding bit in its IMask value is set to 1 and disabled when it is set to 0 (zero). If an output is disabled (i.e. its IMask bit is zero), the State value will have no effect on that output.

Note: You should only enable those outputs you want to turn on or off. If an output is enabled, and its corresponding State value bit is not set correctly, the output may be in an undesired state after the tool runs.

Examples

Here are some examples combining the IDo and IMask parameters.

	Value	6	5	Out-puts 4	3	2	1
IMask	63 (Y = Enabled)	Y	Y	Y	Y	Y	Y
IDo	17	Of f	On	Off	Off	Off	On
IMask	36 (Y = Enabled)	Y			Y		
IDo	0 (– No change)	Of f	–	–	Off	–	–
IMask	28 (Y = Enabled)		Y	Y	Y		
IDo	28 (– No change)	–	On	On	On	–	–

Reading and Writing Tool Ports

All read and write operations to a Datalogic camera port use the same basic parameters with different data types (tool inputs and outputs are also called ports) depending on the port’s data type. The common parameters are:

Parameters:

- IDevHnd..... The Datalogic camera’s connection handle
- sProgram The name of the vision program
- sTask The desired task name
- sTool..... The desired tool name
- sPort The desired input or output port name
- subPort..... The desired subport name. Subports provide variations of the port’s data type. You can only read data from a subport.

Read and Write Method Call Conditions

- The Device Connection handle and Program name are required by all calls. Any remaining parameters that are not needed must be passed as an empty string.
- To access System Object ports, use the Program name “Vision System” and the desired System Object name as the Tool name. For example, to access the CPU Temperature port on the Camera System Object you would use the following parameters:
 Program Name: “Vision System”
 Task Name: ""
 Tool Name: “System”
 Port Name: “CPU Temperature”
- To access Task ports, leave the Tool name blank. The Task name must be present. For example, to access the Trigger ID port on the task you would use the following parameters:
 Program Name: the program name
 Task Name: the task name
 Tool Name: ""
 Port Name: “Trigger ID”
- A Port name is required for all calls.
- The Subport name is optional. You can only read data from a subport.
- Ports that are inputs may be read or written. Ports that are outputs may only be read.

Get Image Port

GetImagePort(IDevHnd, sProgram, sTask, sTool, sPort, pPngImage)

Parameters:

See “Reading and Writing Tool Ports” on page 4-5 for a description of the port selection parameters.

IDevHnd..... The Datalogic camera’s connection handle

pPngImage The image as an ImpactLib.pngImage

This method gets a single PNG image from the specified image port. The image can be displayed in an image display control appropriate to the programming language you are using. See “PngImage Interface Methods” on page 9-1 for more details about the pngImage interface.

Get Real Port

GetRealPort(IDevHnd, sProgram, sTask, sTool, sPort, sSubport, fData)

Parameters:

See “Reading and Writing Tool Ports” on page 4-5 for a description of the port selection parameters.

fData..... The port value as a Floating point number

This method reads the value of the specified Real port or subport and returns it in fData.

Set Real Port

SetRealPort(IDevHnd, sProgram, sTask, sTool, sPort, sSubport, fData)

Parameters:

See “Reading and Writing Tool Ports” on page 4-5 for a description of the port selection parameters.

fData The data as a Floating point number

This method writes the fData value to the specified Real port. The subport parameter must be blank. You cannot write values to a subport.

Get Integer Port

GetIntPort(IDevHnd, sProgram, sTask, sTool, sPort, sSubport, iData)

Parameters:

See “Reading and Writing Tool Ports” on page 4-5 for a description of the port selection parameters.

iData The data as an Integer

This method reads the value of the specified Integer port or subport and returns it in iData.

Set Integer Port

SetIntPort(IDevHnd, sProgram, sTask, sTool, sPort, sSubport, iData)

Parameters:

See “Reading and Writing Tool Ports” on page 4-5 for a description of the port selection parameters.

iData The data as an Integer

This method writes the iData value to the specified Integer port. The subport parameter must be blank. You cannot write values to a subport.

Get Boolean Port

GetBoolPort(IDevHnd, sProgram, sTask, sTool, sPort, sSubport, lData)

Parameters:

See “Reading and Writing Tool Ports” on page 4-5 for a description of the port selection parameters.

lData..... The data as a Long value (1 is True, 0 is False)

This method reads the value of the specified Boolean port or subport and returns it in lData.

Set Boolean Port

SetBoolPort(IDevHnd, sProgram, sTask, sTool, sPort, sSubport, lData)

Parameters:

See “Reading and Writing Tool Ports” on page 4-5 for a description of the port selection parameters.

lData The data as a Long value (1 is True, 0 is False)

This method writes the value of lData to the specified Boolean port. The subport parameter must be blank. You cannot write values to a subport.

Get String Port

GetStringPort(lDevHnd, sProgram, sTask, sTool, sPort, sSubport, sData)

Parameters:

See “Reading and Writing Tool Ports” on page 4-5 for a description of the port selection parameters.

sData The data as a String

This method reads the value of the specified String port or subport and returns it in sData.

Set String Port

SetStringPort(lDevHnd, sProgram, sTask, sTool, sPort, sSubport, sData)

Parameters:

See “Reading and Writing Tool Ports” on page 4-5 for a description of the port selection parameters.

sData The data as a String

This method writes the value sData to the specified String port. The subport parameter must be blank. You cannot write values to a subport.

Code Examples

Here is a simple example of how to call methods in the Operate Interface.

```
Sub ShowOperate()
    Dim oper As ImpactLib.Operate
    Dim lRes As Long
    Dim anyData As VariantType
    Dim lDevHnd As Long
    Dim isOnline As Boolean
    Dim fileCam As Byte
    Dim dio As Long
    Dim dioMask As Long
    Dim pngImage As ImpactLib.pngImage

    oper = New ImpactLib.Operate()

    lRes = oper.SetOnline( lDevHnd, True )
    lRes = oper.GetOnline( lDevHnd, isOnline )
    lRes = oper.RunTask( lDevHnd, "progname", "taskname" )
    lRes = oper.RunTool( lDevHnd, "progname", "taskname", "toolname" )
    lRes = oper.Trigger( lDevHnd )
    lRes = oper.GetImagePort( lDevHnd, "program", "taskname", "toolname", "port-
name", pngImage )
```

```
    lRes = oper.SetImagePort( lDevHnd, "program", "taskname", "toolname", "port-  
name", pngImage )  
    lRes = oper.SnapImage( lDevHnd, pngImage )  
    lRes = oper.GetFileCamera( lDevHnd, fileCam )  
    lRes = oper.SetFileCamera( lDevHnd, False )  
    lRes = oper.OnlineOnStartup( lDevHnd, True )  
    lRes = oper.ReadDigitalInputs( lDevHnd, dio )  
    lRes = oper.WriteDigitalOutputs( lDevHnd, dio, dioMask )  
    lRes = oper.TrainTool( lDevHnd, "program","taskname", "toolname" )  
End Sub
```


Device Events Interface

Device Events Interface Introduction

Events on the Datalogic device are reported back to the application using a Microsoft-standard Events interface. When your program connects to the Datalogic camera, a subscription is created to get a callback for the four most important events on the Datalogic device. These events are ImageIn, Bad Image, Online, and Offline.

The implementation for using these events is very dependent on the programming language being used.

Device Events Interface Methods

Impact Event

ImpactEvent(ByVal eEvent As ImpactLib.ImpactEventType)

Parameters:

eEvent An enumeration specifying the event that happened

When your program connects to a Datalogic camera, an event subscription is created. When one of the following events occurs on the camera, a callback with this signature is called from the ActiveX control. Your program can create a routing to process this callback and determine the event type based on its enumeration.

System Event Types

- Image In: This event occurs when the Datalogic camera has completed acquiring an image.
- Bad Image: This event occurs if the acquired image is incomplete or corrupt.
- Online: This event occurs when the camera goes from offline to online.
- Offline: This event occurs when the camera goes from online to offline.
- Connection Lost: This event occurs when the ActiveX software cannot communicate with the Datalogic device after a period of time.
- Connection Restored: This event occurs when the Datalogic device communication is restored and the user session is still viable. This means there was a temporary network outage or that the Datalogic device was too busy to respond for an extended period.
- Device Reconnected: This event occurs when the same Datalogic device was found but a new user is logged in. This is likely due to the Datalogic device being powered off and on.

Custom Event Types

There are eight custom events available from using the VPM Scheduled Event tool. If tool sets the Event ID from zero through seven, and the event is subscribed, it will be sent to the event handler. You can put these

events wherever you want in your VPM programs to signal that specific actions have occurred. For example, you could have a special event for a bad part or some other specific failure type. The event could also be used to signal the program to get data less frequently than every image. For example, you could send a Scheduled Event 0 for every tenth image snapped.

Set Device Handle

SetDeviceHandle(IDevHnd)

Parameters:

IDevHnd..... The device handle for the connection

Calling this method allows ActiveX to direct incoming events for the device handle to be directed to the correct event handler. This way events from a specific device are passed to a single ImpactEvent() callback.

Set Subscription

SetSubscription(IDevHnd, eventType, subscribe)

Parameters:

IDevHnd..... The device handle for the connection

eventType The event from ImpactEventType

subscribe A Long for subscribing (true) or unsubscribing (false)

Calling this method controls whether the Datalogic device sends the specified event to the event handler for this connection. This only affect the handler specified through SetDeviceHandle().

Code Examples

Here is a simple example of how to process events in the Event Interface.

Visual Basic

```
Public WithEvents ImpEvt As ImpactLib.DeviceEvents
,
'Create an event handler and tie it to a device handle
,
Private Sub CreateHandler()
    Dim bSubscribed as Boolean
    ImpEvt = New ImpactLib.DeviceEvents
    ImpEvent.SetDeviceHandle ( IDevHnd )
    ImpEvent.SetSubscription ( IDevHnd, ImpactEventType.ImpactEventOnline,
    true )
    ImpEvent.GetSubscription ( IDevHnd, ImpactEventType.ImpactEventOnline,
    bSubscribed )
End Sub
,
'Called when an event is received
,
Private Sub ImpEvt_ImpactEvent( ByVal status As ImpactLib.ImpactEventType )
    If ( status = ImpactEventType.ImpactEventOnline ) Then
        sTemp = "Online Event"
```

```
ElseIf ( status = ImpactEventType.ImpactEventOffline ) Then
    sTemp = "Offline Event"
ElseIf (status = ImpactEventType.ImpactEvenOther ) Then
    sTemp = "Other Event"
ElseIf (status = ImpactEventType.ImpactUnknown ) Then
    sTemp = "Unknown Event"
Else
    sTemp = "Bad event enumeration"
End If
```


Configuration Utility Interface

Configuration Utility Interface Introduction

The Configuration Utility interface provides access to the timers used to judge connection health and two helper functions that map events and errors to strings.

Configuration Utility Interface Properties

Connection Timeout

Parameter:

lDevHnd..... The device handle for the connection

This is a get and set property that returns the number of seconds before the connection will be flagged as "Not connected." After that time the ActiveX software will attempt to find the Datalogic device and re-establish communications with it. In unstable environments it may be useful to increase the value from the default of 10 seconds. The connection time out is automatically increased to any user-set command time out.

Command Timeout

Parameter:

lDevHnd..... The device handle for the connection

This is a get and set property that returns the number of seconds any command may take before the communication will time out. The default setting is one second. This may not be enough time if the Datalogic device is very busy processing images, especially if the processing approaches one second. This value applies to all commands.

Certain methods have a longer time out value default of ten seconds. These methods will use the ten seconds as a minimum. Methods with the long time out are:

Programs Interface

- GetLoaded
- GetFiles
- Load
- Unload
- Write
- Read

Operate Interface

- TrainTool
- Info Interface
- GetDetails
 - SetCameraCal
 - GetCameraCal
 - SyncTimeToPC

The Command Timeout value set will apply to all subsequent interactions with the Datalogic device using the connection specified by the device handle.

Next Command Timeout

Parameter:

IDevHnd..... The device handle for the connection

that returns the number of seconds just the next command may take before the communication will time out.

This is a get and set property that returns the number of seconds that just the next command may take before the communication will time out. After the next command on the connection specified by the device handle, the default timeout will be used for subsequent commands.

Event Description

Parameter:

eEvent An ImpactEventType enumeration

This is a get property that returns an English sentence describing the event. It is returned as an English string and cannot be internationalized.

Error Description

Parameter:

status An ImpactErrorCode enumeration

This is a get property that returns an English sentence describing the error. It is returned as an English string and cannot be internationalized.

Code Examples

Here is a simple example of how to use the Configuration Utility Interface Properties.

Visual Basic

```
Private Sub ShowConfigUtility()  
    Dim cnfgUtil As ImpactLib.ConfigUtility  
    Dim sString as String  
    Set fileData = New ImpactLib.ConfigUtility
```

```
cnfgUtil.ConnectionTimeout(1DevHnd) = 40
cnfgUtil.CommandTimeout(1DevHnd) = 3
cnfgUtil.NextCommandTimeout(1DevHnd) = 30
sString = cnfgUtil.EventDescription( ImpactEventType.ImpactEventUnknown )
sString = cnfgUtil.ErrorDescription( ImpactErrorCode.ImpactOK)
End Sub
```


File Data Interface

File Data Interface Methods Introduction

The File Data interface encapsulates data about a set of files and the data is presented as a set of properties. The number of files present in the data is given by the “size” property and the specific file being referenced is accessed with the “index” property. When you set the index to the desired file, that file’s properties can be accessed.

You cannot set the file attributes through this interface. If you want to change the values, you will need to use other method calls or VPM.

File Data Interface Methods

Property size

- This property holds the number of files on the camera. The value is zero when no files are present.

Property index

- This property specifies which set of file information is being accessed. Set this index to access information about a specific file in the interface. The first element in the interface is zero (0) and the last element is the value size-1. Index values outside that range are invalid.

Property fileName

- The indexed file name as a string.

Property programName

- The name of the program contained in the indexed file as a string.

Property loaded

- This property indicates whether the indexed file is currently loaded on the camera. The value is True if the file is loaded and False if it is not.

Property loadOnStartup

- This property indicates whether the indexed file will load on startup. The value is True if the file will load on startup and False if it will not.

Property dateModified

- This property indicates the date the file was written to or changed on the Datalogic device.

Property fileSize

- This property indicates the size (in bytes) of the file on the Datalogic device.

Code Examples

Here is a simple example of how to access files on the Datalogic camera using the File Data Interface.

Visual Basic

```
,  
'Read loaded file data and unload all files  
,  
Private Sub ReadFileData()  
    Dim fileData As ImpactLib.FileData  
    Set fileData = New ImpactLib.FileData  
  
    lRes = prog.GetLoaded(devHnd, fileData)  
  
    ' Go from zero to size minus one  
    For iCnt As Integer = 0 To fileData.size - 1  
        fileData.index = iCnt  
        lRes = prog.Unload(devHnd, fileData.programName)  
    Next  
End Sub
```

Device Details Interface

Device Details Interface Methods Introduction

The Device Details interface contains information about one or more Datalogic Cameras. The data is presented as a set of properties. The number of cameras (or devices) present is given by the “size” property and the specific device being referenced is accessed with the “index” property. When you set the index to the desired device, that device’s properties can be accessed.

You cannot set the device detail attributes through this interface. If you want to change the values, you will need to use other method calls or VPM.

Device Details Interface Methods

Property size

- This property holds the number of Datalogic devices found through the interface. The value is zero when no devices are present.

Property index

- This property specifies which set of Datalogic device information is being accessed. Set this index to access information about a specific device in the interface. The first element in the interface is zero (0) and the last element is the value size-1. Index values outside that range are invalid.

Property swRev

- The indexed device’s software version, as a string.

Property deviceName

- The indexed device’s Device Name, as a string.

Property deviceDescription

- The indexed device’s Device Comment property as a string.

Property IPAddress

- The indexed device’s IP Address ,as a string.

Property onLineOnStartup

- The indexed device’s BOOLEAN property that indicates whether the device will go online at startup or not. The value is True if the Datalogic device is set to go online at startup and False if it is not.

Property serialNumber

- The indexed device's serial number, as a string.

Property modelNumber

- The indexed device's model, as a string. A value of "1" indicates a C-Series, a "2" indicates a T-Series, a "3" indicates an A-Series, and "Emulator" indicates the Emulator.

Property cameraModel

- The indexed device's camera model, as a string. This descriptive string includes the camera resolution. For example, "T20 640x480."

Property cameraModelNumber

- The indexed device's camera model, as a long.

Code Examples

Here is a simple example of how to access details about the Datalogic camera using the Device Details Interface.

Visual Basic

```
,
'Get list of DEVICES and set data into combo box
,
Private Sub loadComboBox()
    Dim disc As ImpactLib.Discovery
    Dim i As Integer
    disc = New ImpactLib.Discovery
    devInfo = New ImpactLib.DeviceDetails

    disc.Find(devInfo)

    ' Check for no devices
    If devInfo.size = 0 Then
        IPSelector.Text = "No devices found"
    Else
        'Loop from zero to size minus one
        For i = 0 To devInfo.size - 1
            devInfo.index = i
            IPSelector.Items.Add(devInfo.IPAddress)
        Next
        IPSelector.SelectedIndex = 0
    End If
End Sub
```

PngImage Interface

PngImage Interface Introduction

The PngImage interface encapsulates an IMAGE in PNG format. It includes a method to extract the PNG as an IPictureDisp which is a Microsoft-standard format for passing images through COM and ActiveX.

You can read and set the image attributes through this interface.

PngImage Interface Methods

Property pngDesc

This property specifies the description of the image, as a string, when the image is written to the Datalogic device, or the image identifier when the image is read from the Datalogic device.

Property byteCnt

This property indicates the size of the image, in bytes, as a Long value.

Property pngData

This property accesses the binary PNG data using a pointer to a Byte array

ConvertToIpictureDisp

ConvertToIPictureDisp(IPictureDisp * picture)

Parameters:

 picture A picture to a user-allocated picture control

This method converts the PNG image contained in the interface into an IPictureDisp picture. Microsoft provides methods to convert an IPictureDisp interface into other forms, if needed. This method will vary with the programming language used and the version of that language.

Code Examples

Here is a simple example of how to access images on the Datalogic camera using the PngImage Interface.

Visual Basic

```
Dim pngImage as ImpactLib.IpngImage
Dim picDisp as stdole.IPictureDisp

pngImage = New ImpactLib.IpngImage

'Get the image
```

```
lRes = impactOper.GetImagePort(parentDevForm.GetDevHnd(), sProg, Task, sRead-  
Tool, sReadPort, pngImg)  
  
'If the call is good, then continue  
If lRes = ImpactErrorCode.ImpactOK Then  
  
    'Convert the PNG to an IPictureDisp  
    picDisp = pngImg.ConvertToIPictureDisp()  
  
    'Now convert the IPictureDisp to an Image and assign it to the control  
    If (Not IsNothing(picDisp)) Then  
        If (picDisp.Handle <> 0 And picDisp.Width > 0 And picDisp.Height > 0 And  
picDisp.hPal >= 0 And picDisp.Type = 1) Then  
  
            Dim hBitmap As IntPtr = New System.IntPtr(picDisp.Handle)  
            If hBitmap <> 0 Then  
                ImagePictureBox.Image = Bitmap.FromHbitmap(hBitmap)  
            End If  
  
        End If  
    End If  
End If
```

Appendix

Appendix Introduction

This chapter provides additional information about several aspects of implementing the Impact ActiveX interface.

Error Codes

If a method call fails, an error code is returned describing the reason for the failure. Error codes are common across all methods and interfaces. A return code of zero indicates that the method was successful.

Error codes are defined by the `ImpactLib.ImpactErrorCode` enumeration. There are many possible error responses with all calls giving multiple options. The most common errors are listed here.

- `ImpactOK`: Success
- `ImpactCommunicationError`: Failed to communicate with the camera. The camera was found but communication failed.
- `ImpactSmartCameraNotFound`: Failed to find the camera. The camera may be unplugged or the power turned off.
- `ImpactBadHandle`: The `IDevHnd` parameter is invalid. The camera with that handle does not exist on the network.
- `ImpactBadPort`: The program, task, tool, port, or subport parameter is invalid. The object being addressed does not have the parameter indicated in the method call.
- `ImpactBadPortType`: There is a mismatch between the call type and the port type. For example, trying to read an Integer value with a String method call.
- `ImpactEmptySysLog`: An attempt was made to read or clear the system log and it is empty.
- `ImpactCommandNotImplemented`: The method contains an invalid command.
- `ImpactBadCookie`: A cookie file is missing or unreadable.
- `ImpactNotLoggedIn`: If a Datalogic camera has security enabled, the Connect method must provide a username and password to log on.
- `ImpactAborted`: The command was terminated.
- `ImpactBadPngFile`: The method attempted to read or write a bad image file.
- `ImpactBasicCommandNotImplemented`: The attempt was made to run a Basic tool that contains invalid code.
- `ImpactBreak`: An attempt was made to run a Break tool outside of a loop.
- `ImpactCommFailure`: Communications failed with the camera indicated.

- **ImpactDuplicateName:** An attempt was made to load a vision program file on the camera that already has that file or program loaded.
- **ImpactFailed:** The method call failed.
- **ImpactInvalidFileType:** An attempt was made to read or write an invalid file type.
- **ImpactInvalidParm:** The method call contains an invalid parameter.
- **ImpactInvalidRoiSize:** An attempt was made to run a tool that has an invalid Region of Interest (ROI) defined.
- **ImpactInvalidToolType:** The tool parameter is invalid. The tool being accessed cannot be accessed using this method.
- **ImpactInvalidType:** The program, task, tool, port, or subport type parameter is invalid.
- **ImpactNoBasicCode:** The attempt was made to run a Basic tool that contains no code.
- **ImpactNoImage:** An attempt was made to run an image processing tool when it contained no image.
- **ImpactNoModel:** An attempt was made to run an image processing tool when the model had not been trained.
- **ImpactNoStartupApp:** An attempt was made to run a vision program on startup that did not exist.
- **ImpactNotAuthorized:** The method is not authorized.
- **ImpactNotFound:** An attempt was made to access file or program that does not exist.
- **ImpactNotLicensed:** The Datalogic camera does not contain the correct license.
- **ImpactNotOffline:** An attempt was made to set a camera off line that was already off line.
- **ImpactNotRunning:** The Datalogic camera is not running.
- **ImpactNullDataRef:** An attempt was made to run a tool that contains a reference to non-existent data.
- **ImpactObjectDoesNotExist:** An attempt was made to access an object that does not exist.
- **ImpactOpenForWrite:** The indicated file is open for write access.
- **ImpactOutOfMemory:** The Datalogic camera's memory capacity has been exceeded.
- **ImpactOutOfRange:** An attempt was made to write a value that was too large or too small to a port.
- **ImpactReadOnly:** An attempt was made to modify a file that is marked Read Only.
- **ImpactRoiOffImage:** An attempt was made to run a tool that has Region of Interest (ROI) off the image.
- **ImpactTemplateModelRectMismatch:** The Model Rectangle property in a Template tool does not match.
- **ImpactTooManyUsers:** An attempt was made to connect to a Datalogic camera that already has the maximum number of users connected.
- **ImpactToolMissingInputList:** The Input List property is missing for a tool that requires it.
- **ImpactUnableToTrainModel:** A tool with a Model could not be trained.
- **ImpactUnresolvedSymbol:** One of the tools in the vision program reported an internal error.
- **ImpactModelImagePixelSizeMismatch:** A tool with a Model could not run on the input image.
- **ImpactOcrSubCharSizeError:** A character size error was reported by the OCR tool.

- **ImpactNotEnoughEdges:** An attempt was made to run a tool, but the ROI did not find enough edges.
- **ImpactImagePixelSizeMismatch:** Image Pixel Size Mismatch';
- **ImpactImageTooLarge:** An attempt was made to display an image that is too large.
- **ImpactPointMatchError:** An error was reported from a tool that uses point matching.
- **ImpactDataMatrixSizeNotSet:** An attempt was made to run the Data Matrix tool without setting a valid size.
- **ImpactCircleRefineSegmentLengthError:** The Circle Edge Refinement tool has an invalid Segment Length setting.

Code Examples

Method calls vary depending on the programming language being used. This section contains some simple code examples.

NOTE: These are only examples. They are not intended as complete programs and they will not necessarily perform as they are written.

Visual Basic

Here is a simple example of how to connect to a Datalogic camera using Visual Basic.

```
Sub ShowDiscovery()  
    rem this is the IDiscovery interface to the Datalogic device  
    Dim disc As ImpactLib.Discovery  
    rem variable to hold error codes  
    Dim lRes As Long  
    Dim uCnt As Long  
    rem this interface is used to get information about Datalogic devices  
    Dim vDevices As ImpactLib.DeviceDetails  
    rem this handle is used to refer to the Datalogic device currently connected  
    Dim lDevHnd As Long  
  
    disc = New ImpactLib.Discovery  
  
    lRes = disc.Find( vDevices )  
    lRes = disc.CheckConnection( "ipaddr" )  
    lRes = disc.Connect( "192.168.0.1", "User", "Pass", lDevHnd )  
    lRes = disc.CloseConnection( lDevHnd )  
End Sub
```

C++

The ActiveX control for C++ is a COM object with the type library included in the dll, so a header file is not needed. To create the COM header files from the dll, use the `#import` preprocessor directive. Note that you are doing COM programming and not just calling a C++ API, but that is the nature of using ActiveX.

The syntax for the import directive is:

```
#import "pathname\impact.dll"
```

where "pathname" is the "install folder" where `impact.dll` is installed:

C:\install folder\Impact\Vision Device APIs\ActiveX

Here is a simple example of how to discover the number of Datalogic cameras connected to the network.

```
#import "impact.dll"
CoInitialize(NULL);
ImpactLib::IDiscoveryPtr pDisc = ImpactLib::IDiscoveryPtr("Impact.Discovery");
ImpactLib::IDeviceDetailsPtr pDevDetails = ImpactLib::IDeviceDetailsPtr("Impact.DeviceDetails");
pDisc->Find(pDevDetails);
int n = pDevDetails->Getsize(); // how many impacts are on the network?
pDevDetails.Release();
pDisc.Release();
CoUninitialize();
```

Using the Emulator

You can use the Impact Emulator to develop and test your ActiveX applications. It will respond to API commands and queries like a Datalogic hardware vision device.

Multiple Emulators

The Find method call will return the Emulator's IP Address the same as it does a hardware vision device. Also, an ActiveX application program can connect to more than one Impact Emulator at a time just like hardware devices. You may want to use this option if you are simulating an inspection that uses multiple devices.

However, the difference between a hardware device and an Emulator is that, when multiple Emulators are running on the same PC, all the Emulators will have the same IP Address but each one must have a different port number. The port number must be entered on the command line that starts the Emulator.

NOTE: When you are using multiple Emulators with ActiveX, you can only use the port numbers from 9933 to 10032.

Refer to the Impact Reference Guide (Publication # 843-0093) for details about using multiple Emulators.

No Network Connection

When you are using a development computer that is not connected to a network, your ActiveX application program will not find any Impact Emulators after you start them. This is because an Emulator automatically uses the default loopback IP Address of 127.0.0.1 and there is no network route for the Emulator to respond to the ActiveX API call to that localhost address.

If you want to use the Emulator for program development and testing, the simplest solution is to connect your development computer to a network. Since this is not always possible, you can either connect a hardware Ethernet loopback adapter to the PC, or install the Microsoft Loopback Adapter. You can build or buy a hardware Ethernet loopback adapter. The Microsoft Loopback Adapter is included as part of Microsoft Windows.

To install the Microsoft loopback adapter in Windows XP

1. Click **Start**, then click **Control Panel**.
2. Double-click the **Add Hardware** icon, then click **Next**.
3. Click **Yes, I have already connected the hardware**, then click **Next**.
4. At the bottom of the list, click **Add a new hardware device**, and then click **Next**.
5. Click **Install the hardware that I manually select from a list**, and then click **Next**.
6. Click **Network adapters**, and then click **Next**.
7. In the **Manufacturer** box, click **Microsoft**.
8. In the **Network Adapter** box, click **Microsoft Loopback Adapter**, and then click **Next**.
9. Click **Finish**.

To install the Microsoft loopback adapter in Windows Vista

1. Click **Start**, then click **Control Panel**.
2. Double-click the **System** icon, then click the **Hardware** tab.
3. Click the **Device Manager** button.
4. Right click on the topmost item in the hardware list (it should be your computer name).
5. Select **Add legacy hardware**, from the list.
6. Click **Install the hardware that I manually select from a list**, and then click **Next**.
7. Click **Network adapters**, and then click **Next**.
8. In the **Manufacturer** box, click **Microsoft**.
9. In the **Network Adapter** box, click **Microsoft Loopback Adapter**, and then click **Next**.
10. Click **Finish**.

After the adapter is installed, you can configure it using Network Connections in the Control Panel. Use the following TCP/IP settings:

- IP Address: 192.0.2.1
- Subnet Mask: 255.255.255.0
- Loopback adapter is last in the Network Connection order (under Advanced - Advanced Settings in the menu)

Multiple Vision Devices

A single ActiveX application program can connect to multiple Datalogic vision devices and process events from each of those devices. Each connection must have a separate event handler method that is tied back to the connection by calling the SetDeviceHandle() method.

Subsequent application programs can connect to the same Datalogic vision devices, but they cannot receive events from them. Those applications can read and write data on the devices, but they will receive no events from the devices.

ACTIVE X REFERENCE

Index

A

Adapter
 loopback 10-4
 Appendix 10-1
 AutoWhiteBalance 4-1

B

Basic, Visual 10-3

C

C++ 10-3
 Calibration 2-2
 Camera
 check connection 1-1
 connect 1-1
 find 1-1
 online on startup 4-2
 read calibration 2-2
 read details 2-1
 read online status 4-2
 set calibration 2-2
 set online status 4-2
 snap image 4-3
 trigger 4-3
 white balance 4-1
 CheckConnection 1-1
 ClearSysLog 2-1
 Closeconnection 1-2
 Code examples 10-3
 configuration utility interface 6-2
 device details interface 8-2
 discovery interface 1-2
 event interface 5-2
 file data interface 7-2
 information interface 2-2
 operate interface 4-8
 PngImage interface 9-1
 programs interface 3-3
 Command
 timeout property 6-1
 Configuration utility interface 6-1
 Connect 1-1
 Connection
 check 1-1
 close 1-2
 timeout property 6-1

D

Data types Intro-3
 Delete 3-1

file 3-1
 Device
 details 8-1
 multiple connections 10-5
 security Intro-4
 Device details interface 8-1
 Device events interface 5-1
 Digital
 input read 4-4
 outputs set 4-4
 Discovery interface 1-1
 DLLVersion 1-2
 DPWS
 server Intro-2

E

Emulator
 accessing 10-4
 multiple 10-4
 no network 10-4
 Error codes Intro-3
 all 10-1
 Event
 description property 6-2
 method 5-1

F

File camera
 read mode 4-1
 set mode 4-1
 File data interface 7-1
 Files
 delete 3-1
 load 3-2
 load on startup 3-2
 read details 3-1
 read loaded 3-1
 save from camera 3-2
 save to camera 3-3
 unload 3-2
 Find 1-1

G

GetBoolPort 4-7
 GetCameraCal 2-2
 GetDetails 2-1
 GetFileCamera 4-1
 GetFiles 3-1
 GetImagePort 4-6
 GetIntPort 4-7
 GetLoaded 3-1
 GetOnline 4-2
 GetRealPort 4-6
 GetStringPort 4-8

GetSysLog 2-1
Glossary Intro-5

I

Image
 read 4-6
Impact documentation Intro-4
Information interface 2-1
Input
 digital read 4-4
Interface
 programming Intro-3
IP address 1-1, 8-1
 loopback 10-4

L

Language
 non-English Intro-4
Load 3-2
LoadOnStartup 3-2
Log
 clear 2-1
Loopback
 adapter 10-4
 IP address 10-4

M

Method
 AutoWhiteBalance 4-1
 CheckConnection 1-1
 ClearSysLog 2-1
 CloseConnection 1-2
 Connect 1-1
 Delete 3-1
 DLLVersion 1-2
 Find 1-1
 GetBooleanPort 4-7
 GetCameraCalibration 2-2
 GetDetails 2-1
 GetFileCamera 4-1
 GetFiles 3-1
 GetImagePort 4-6
 GetIntegerPort 4-7
 GetLoadedFiles 3-1
 GetOnline 4-2
 GetRealPort 4-6
 GetStringPort 4-8
 GetSystemLog 2-1
 ImpactEvent 5-1
 Load 3-2
 LoadOnStartup 3-2
 OnlineOnStartup 4-2
 Read 3-2
 ReadDigitalInput 4-4
 RunTask 4-2
 RunTool 4-2

SetBooleanPort 4-7
SetCameraCalibration 2-2
SetDeviceHandle 5-2
SetFileCamera 4-1
SetIntegerPort 4-7
SetOnline 4-2
SetRealPort 4-7
SetStringPort 4-8
SetSubscription 5-2
SnapImage 4-3
SyncTimeToPC 2-2
TrainTool 4-3
Trigger 4-3
Unload 3-2
Write 3-3
 WriteDigitalOutputs 4-4
Multiple connections 10-5
Multi-thread access Intro-3

N

Network connection 10-4
Nextcommand
 timeout property 6-2
Notations Intro-3

O

Online
 read status 4-2
 set on startup 4-2
 set status 4-2
OnlineOnStartup 4-2
Operate interface 4-1
Output
 digital write 4-4

P

PngImage interface 9-1
Ports
 non-English Intro-4
 read boolean 4-7
 read image 4-6
 read integer 4-7
 read real 4-6
 read string 4-8
 reading and writing 4-5
 write boolean 4-7
 write integer 4-7
 write real 4-7
 write string 4-8
Programs interface 3-1
Property
 command timeout 6-1
 connection timeout 6-1
 event description 6-2
 next command timeout 6-2

R

Read
 digital input 4-4
 file 3-2
 port 4-5
ReadDigitalInput 4-4
RunTask 4-2
RunTool 4-2

S

Security Intro-4
Server
 DPWS Intro-2
 enable/disable Intro-2
SetBoolPort 4-7
SetCameraCal 2-2
SetDeviceHandle 5-2
SetFileCamera 4-1
SetIntPort 4-7
SetOnline 4-2
SetRealPort 4-7
SetStringPort 4-8
SetSubscription 5-2
SnapImage 4-3
SyncTimeToPC 2-2
System log 2-1
 read 2-1

T

Task
 run 4-2
Time, sync 2-2
Timeout
 command 6-1
 connection 6-1
 next command 6-2
Tool
 run 4-2
 train 4-3
TrainTool 4-3
Translation Intro-4
Trigger 4-3

U

Unload 3-2

V

Vision device
 multiple connections 10-5
Visual Basic 10-3

W

Write
 digital output 4-4
 file 3-3
 port 4-5
WriteDigitalOutputs 4-4

