Chapter
III-15

Platform-Related Issues

Platform-Related Issues

Igor Pro runs on Macintosh and Windows. This chapter contains information that is platform-specific and also information for people who use Igor on both platforms.

Windows-Specific Issues

On Windows, the name of the Igor program file must be "Igor64.exe" for the 64-bit version of Igor or "Igor.exe" for the 32-bit version, exactly. If you change the name, Igor extensions will not work because they will be unable to find Igor.

Cross-Platform File Compatibility

Version 3.1 was the first version of Igor Pro that ran on Windows as well as Macintosh.

Crossing Platforms

When crossing from one platform to another, page setups are only partially translated. Igor tries to preserve the page orientation and margins.

When crossing platforms, Igor attempts to do font substitution where necessary. If Igor can not determine an appropriate font it displays the font substitution dialog where you can choose the font.

Platform-specific picture formats are displayed as gray boxes when you attempt to display them on the non-native platform. PDF pictures are supported on Macintosh only and appear as gray boxes on Windows. Windows Metafile, Enhanced Metafile, and Windows bitmap (BMP) pictures are supported on Windows only and appear as gray boxes on Macintosh. The EPS, PNG, JPEG, TIFF, and SVG formats are platform-independent and are displayed on both platforms.

Transferring Files Using File Transfer Programs

Some transfer programs offer the option of translating file formats as they transfer the program from one computer to another. This translation usually consists of replacing each carriage return character with a carriage return/linefeed pair (Macintosh to Windows) or vice-versa (Windows to Macintosh). This is called a "text mode" transfer, as opposed to a "binary mode" transfer. This translation is appropriate for plain text files only. In Igor, plain text notebooks, procedure files, and Igor Text data files are plain text. All other files are not plain text and will be corrupted if you transfer in text mode. If you get flaky results after transferring a file, transfer it again making sure text mode is off.

If you have a problem opening a binary file after doing a transfer, compare the number of bytes in the file on both computers. If they are not the same, the transfer has corrupted the file.

File Name Extensions, File Types, and Creator Codes

This table shows the file name extension and corresponding Macintosh file type for Igor Pro files:

Extension	File Type	What's in the File
.pxp	IGsU	Packed experiment file
.pxt	IGsS	Packed experiment template (stationery)
.uxp	IGSU	Unpacked experiment file
.uxt	IGSS	Unpacked experiment template (stationery)
.ifn	WMT0	Igor formatted notebook (last character is zero)
.txt	TEXT	Igor plain notebook
.ihf	WMT0	Igor help file

Extension	File Type	What's in the File
.ibw	IGBW	Igor binary data file

The Macintosh creator code for Igor is 'IGR0' (last character is zero).

Experiments and Paths

An Igor experiment sometimes refers to wave, notebook, or procedure files that are stored separate from the experiment file itself. This is discussed under **References to Files and Folders** on page II-22. In this case, Igor creates a symbolic path that points to the folder containing the referenced file. It writes a **NewPath** command in the experiment file to recreate the symbolic path when the experiment is opened. When you move the experiment to another computer or to another platform, this path may not be valid. However, Igor goes to great lengths to find the folder, if possible.

Igor stores the path to the folder containing the file as a relative path, relative to the experiment file, if possible. This means that Igor will be able to find the folder, even on another computer, if the folder's relative location in the disk hierarchy is the same on both computers. You can minimize problems by using the same disk hierarchy on both computers.

If the folder is not on the same volume as the experiment file, then Igor can not use a relative path and must use an absolute path. Absolute paths cause problems because, although your disk hierarchy may be the same on both computers, often the name of the root volume will be different. For example, on the Macintosh your hard disk may be named "hd" while on Windows it may be named "C:".

If Igor can not locate a folder or file needed to recreate an experiment, it displays a dialog asking you to locate it.

Picture Compatibility

Igor displays pictures in graphs, page layouts, control panels and notebooks. The pictures are stored in the Pictures collection (Misc→Pictures) and in notebooks. Graphs, page layouts and control panels reference pictures stored in the Pictures collection while notebooks store private copies of pictures.

This table shows the graphic formats that Igor can use to store pictures:

Format	How To Create	Notes
PDF	Paste or use Misc→Pictures	Macintosh only
EMF (Enhanced Metafile)	Paste or use Misc→Pictures	Windows only See **Graphics Technology on Windows** on page III-445 for information about different types of EMF pictures.
BMP (bitmap)	Use Misc→Pictures	Windows Only. BMP also called DIB (device-independent bitmap).
PNG (Portable Network Graphics)	Use Misc→Pictures	Cross-platform bitmap format
JPEG	Use Misc→Pictures	Cross-platform bitmap format
TIFF (Tagged Image File Format)	Use Misc→Pictures	Cross-platform bitmap format
EPS (Encapsulated PostScript)	Use Misc→Pictures	High resolution vector format. EPS is largely obsolete. Use PDF instead.
SVG (Scalable Vector Graphics)	Use Misc→Pictures	Cross-platform high resolution vector format.

Formats that are not supported on the current platform are drawn as gray boxes.

Although Igor does not display non-native graphic formats, it does preserve them. For example, you can create an experiment on Macintosh and paste a Macintosh PDF into a page layout, graph, or notebook window. If you save the experiment and open it on Windows, the PDF will be displayed as a gray box. You can now paste a Windows metafile into the page layout, graph, or notebook window. If you save the experiment and open it on Macintosh, the Windows metafile will be displayed as a gray box but the PDF will be displayed correctly. If you now save and open the experiment on Windows again, the Windows metafile will be displayed correctly.

If you want platform-specific pictures to be displayed correctly on both platforms, you must convert the pictures to PNG. To convert to PNG, use the Pictures dialog (Misc menu) for pictures in graphs and page layouts, or for pictures in notebooks, use the Special submenu in the Notebook menu.

Converting a picture to PNG makes it a bitmap format and may degrade resolution. This is fine for graphics intended to be viewed on the screen but not for graphics intended to be printed at high resolution. You can convert to a high resolution PNG without losing much picture quality.

Page Setup Compatibility

Page setup records store information regarding the size and orientation of the page. Prior to Igor Pro 7, page setups contained platform-depedent and printer-depedent data. This is no longer the case, but as a consequence, only minimal information is stored.

In each experiment file, Igor stores a separate page setup for each page layout, notebook, and procedure window, and stores a single page setup for all tables.

File System Issues

This section discusses file system issues that you need to take into account if you use Igor on both Macintosh and Windows.

File and Folder Names

On Windows, the following characters are illegal in file and folder names: backslash (\), forward slash (/), colon (:), asterisk (*), question mark (?), double-quote ("), left angle bracket (<), right angle bracket (>), vertical bar (|). On Macintosh, the only illegal character is colon.

This means, for example, that you can not create a file with a name like “Data 1/23/98” on Windows. You can create a file with this name on Macintosh. If you write an Igor procedure that generates a file name like this, it will run on Macintosh but fail on Windows.

Therefore, if you are concerned about cross-platform compatibility, you must not use any of the Windows illegal characters in a file or folder name, even if you are running on Macintosh. Also, don’t use period except before a file name extension.

File and folder names in Windows can theoretically be up to 255 characters in length. Because of some limitations in Windows and also in Igor, you will encounter errors if you use file names that long. However, both Igor and Windows are capable of dealing with file names up to about 250 characters in length. It is unlikely that you will approach this limit.

An exception to this is that Igor limits names of XOP files to 31 characters, plus the “.xop” extension. Igor will not recognize an XOP file with a longer name.

Paths in Windows are limited to 259 characters in length. Neither Windows nor Igor can deal with a path that exceeds this limit. For example, if you create a directory with a 250 character name and try to create a file with a 15 character name, neither Windows nor Igor will permit this.

This boils down to the following: Feel free to use long file and directory names, but expect to see errors if you use outrageously long names or if you have directories so deeply nested that paths approach the theoretical limit.

Path Separators

The Macintosh HFS file system uses a colon to separate elements in a file path. For example:

```
hd:Igor Pro 7 Folder:Examples:Sample Graphs:Contour Demo.pxp
```

The Windows file system uses a backslash to separate elements in a file path. For example:

```
C:\Igor Pro 7 Folder\Examples\Sample Graphs:Contour Demo.pxp
```

Some Igor operations (e.g., LoadWave) allow you to enter file paths. Igor accepts Macintosh-style or Windows-style paths regardless of the platform on which you are running.

Note: Igor uses the backslash character as an escape character in literal strings. This can cause problems when using Windows-style paths.

For example, the following command creates a textbox with two lines of text. "\r" is an escape code inserts a carriage return character:

```
Textbox "This is line 1.\rThis is line 2."
```

Because Igor interprets a backslash as an escape character, the following command will not execute properly:

```
LoadWave "C:\Data Files\really good data.ibw"
```

Instead of loading a file named "really good data.ibw", Igor would try to load a file named "Data Files<CR>eally good data.ibw", where <CR> represents the carriage return character. This happens because Igor interprets "\r" in literal strings to mean carriage return.

To solve this problem, you must use "\\" instead of "\" in a file path. Igor will correctly execute the following:

```
LoadWave "C:\\Data Files\\really good data.ibw"
```

This works because Igor interprets "\\" as an escape sequence that means "insert a backslash character here".

Another solution to this problem is to use a Macintosh HFS-style path, even on Windows:

```
LoadWave "C:Data Files:really good data.ibw"
```

Igor converts the Macintosh HFS-style path to a Windows-style path before using it. This avoids the backslash issue.

For a complete list of escape sequences, see **Escape Sequences in Strings** on page IV-13.

If you are writing procedures that need to extract sections of file paths or otherwise manipulate file paths, the **ParseFilePath** function on page V-621 may come in handy.

UNC Paths

"UNC" stands for "Universal Naming Convention". This is a Windows convention for identifying resources on a network. One type of network resource is a shared directory. Consequently, when running under Windows, in order to reference a network directory from an Igor command, you need to use a UNC path.

The format of a UNC path that points to a file in a folder on a shared server volume or directory is:

```
"\\server\share\directory\filename"
```

"server" is the name of the file server and "share" is the name of the top-level shared volume or directory on that server.

Because Igor treats a backslash as an escape character, in order to reference this from an Igor command, you would have to write:

```
"\\\\server\\share\\directory\\filename"
```

As described in the preceding section, you could also use Macintosh HFS path syntax by using a colon in place of two backslashes. However, you can not do this for the "\\server\share" part of the path. Thus, using Macintosh HFS syntax, you would write:

```
"\\\\server\\share:directory:filename"
```

Unix Paths

Unix paths use the forward slash character as a path separator. Igor does not recognize Unix paths. Use Macintosh HFS paths instead.

Keyboard and Mouse Usage

This section describes how keyboard and mouse usage differs on Macintosh versus Windows. It is intended to help Igor users more easily adapt when switching platforms.

There are three main differences between Macintosh and Windows input mechanisms:

1. The Macintosh mouse may have one button and the Windows mouse has two.
2. The Macintosh keyboard has four modifier keys (Shift, Command, Option, Control) while the Windows keyboard has three (Shift, Ctrl, Alt).
3. The Macintosh keyboard has Return and an Enter keys while the Windows keyboard (usually) has two Enter keys.

For the most part, Igor maps between Macintosh and Windows input as follows:

Macintosh	Windows
Shift	Shift
Command	Ctrl
Option	Alt
Control	<not mapped>

Macintosh	Windows
Return	Enter
Enter	Enter
Control-click	Right-click

In notebooks, procedure windows and help windows, pressing Control-Return or Control-Enter executes the selected text or, if there is no selection, to execute the line of text containing the caret.

Command Window Input

This table compares command window mouse actions:

Action	Macintosh	Windows
Copy history selection to command line	Option-click	Alt+click
Copy history to command and start execution	Command-Option-click	Ctrl+Alt+click
Invoke contextual menu	Control-click	Right-click

Cross-Platform Text and Fonts

Text Encoding Compatibility

Prior to Igor7, Igor used system text encoding. On Macintosh, this was usually MacRoman. On Windows, it was usually Windows-1252. On Japanese systems, it was Shift JIS on both platforms.

As of Igor7, Igor uses UTF-8 text encoding internally on both Macintosh and Windows.

When opening old files, Igor must convert from the file's text encoding to UTF-8 for storage in memory.

Dealing with various text encodings is a complex issue. See **Text Encodings** on page III-409 for details.

Carriage Returns and Linefeeds

The character or character pattern that marks the end of a line of text in a plain text file is called the "line terminator". There are three common line terminators, carriage return (CR, ASCII 13, used on old Macin-

tosh systems), linefeed (LF, ASCII 10, used on Unix) and carriage return plus linefeed (CRLF, used on Windows).

When Igor Pro opens a text file (procedure file, plain text notebook or plain text data file), it accepts CR, LF or CRLF as the line terminator.

If you create a new procedure file or plain text notebook, Igor writes LF on Mac OS and CRLF on Windows. If you open an existing plain text file, edit it and then save it, Igor preserves the original terminator as determined by examining the first line in the file.

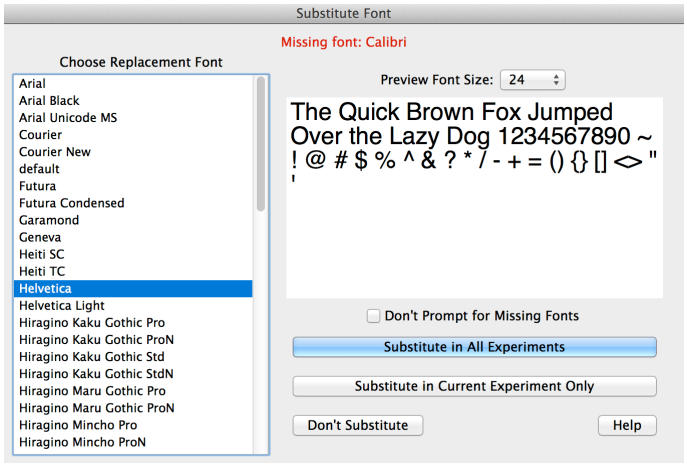
By default, the **FReadLine** operation treats CR, LF, or CRLF as terminators. Use this to write a procedure that can read lines from a text file without caring whether it is a Macintosh, Windows, or Unix file.

Font Substitution

When a font specified in a command or document is not installed, Igor applies font substitution to choose an installed font to use in place of the missing font. Dealing with these missing fonts often occurs when transferring a Windows-originated document to Macintosh or vice versa.

Igor employs two levels of font substitution: user-level editable substitution and built-in uneditable substitution.

The first level is an optional user-level font substitution facility that you will usually encounter for the first time when Igor displays the Substitute Font Dialog while opening an experiment or file. Use the dialog to choose a temporary or permanent replacement for the missing font:



The second level is Igor's built-in substitution table which substitutes between fonts normally installed on various Macintosh and Windows operating systems. For example, it substitutes Arial (a standard Windows font) for Geneva (a standard Macintosh font) if Geneva is not installed (which it usually isn't on a Windows computer).

The user-level substitutions have priority over the built-in substitutions.

The Substitute Font dialog appears when neither level of font substitution specifies a replacement for the missing font. You can prevent this dialog from appearing by selecting the Don't Prompt for Missing Fonts checkbox in the Substitute Font dialog.

You can manage font substitutions without waiting for a missing font situation to occur by choosing Misc-Edit Font Substitutions.

The user-level font substitution table is maintained in the "Igor Font Substitutions.txt" text file in Igor's preferences folder. The file format is:

```
<name of missing font to replace> = <name of font to use instead>
```

one entry per line. For example:

```
Palatino=New Times Roman
```

spaces or tabs are allowed around the equals sign.

When a missing font is replaced, Igor uses the name of the replacement font instead of the name of the font in the command.

The name of the missing font is replaced only in the sense that the altered or created object (window, control, etc.) uses and remembers only the name of the replacement font. Recreation macros, including experiment recreation procedures, use the name of the replacement font when the experiment is saved. The command, however, is unaltered and still contains the name of the missing font.

Cross-Platform Procedure Compatibility

Igor procedures are about 99.5% platform-independent. For the other 0.5%, you need to test which platform you are running on and act accordingly. You can use ifdefs to achieve this. For example:

```
Function Demo()
   #ifdef MACINTOSH
      Print "We are running on Macintosh"
   #else
      #ifdef WINDOWS
         Print "We are running on Windows"
      #else
         Print "Unknown OS"
      #endif
   #endif
End
```

File Paths

As described under **Path Separators** on page III-401, Igor accepts paths with either colons or backslashes on either platform.

The use of backslashes is complicated by the fact that Igor uses the backslash character as an escape character in literal strings. This is also described in detail under **Path Separators** on page III-401. The simplest solution to this problem is to use a colon to separate path elements, even when you are running on Windows.

If you are writing procedures that need to extract sections of file paths or otherwise manipulate file paths, the **ParseFilePath** function on page V-621 may come in handy.

File Types and Extensions

On Mac OS 9, all files had a file type property. This property is a four letter code that is stored with the file by the Macintosh file system. For example, plain text files have the file type TEXT. Igor binary wave files have the file type IGBW. The file type property controlled the icon displayed for the file and which programs could open the file.

The file type property is no longer used on Macintosh. On Mac OS X, as well as on Windows, the file type is indicated by the file name extension.

For backward compatibility, some Igor operations and functions, such as **IndexedFile**, still accept Macintosh file types. New code should use extensions instead.

Points Versus Pixels

A pixel is the area taken up by the smallest displayable dot on an output device such as a display screen or a printer. The physical width and height of a pixel depend on the device.

In Igor, most measurements of length are in terms of points. A point is roughly 1/72 of an inch. 72 points make up 1 "logical inch". Because of hardware differences and system software adjustments, the actual size of a logical inch varies from screen to screen and system to system.

Window Position Coordinates

With one exception, Igor stores and interprets window position coordinates in units of points. For example the command

```
Display/W=(5, 42, 405, 242)
```

specifies the left, top, right, and bottom coordinates of the window in points relative to a reference point which is, roughly speaking, the top/left corner of the menu bar. Other Igor operations that use window position coordinates in points include Edit, Layout, NewNotebook, NewGizmo and MoveWindow.

The exception is the control panel window when running on a standard resolution screen. This is explained under **Control Panel Resolution on Windows** on page III-405.

Most users do not need to worry about the exact meaning of these coordinates. However, for the benefit of programmers, here is a discussion of how Igor interprets them.

On Macintosh, the reference point, (0, 0), is the top/left corner of the menu bar on the main screen. On Windows, the reference point is 20 points above the bottom/left corner of the main Igor menu bar. This difference is designed so that a particular set of coordinates will produce approximately the same effect on both platforms, so that experiments and procedures can be transported from one platform to another.

The coordinates specify the location of the content region, in Macintosh terminology, or the client area, in Windows terminology, of the window. They do not specify the location of the window frame or border.

On Macintosh, a point is always interpreted to be one pixel except on Retina displays where it is two.

On Windows, the correspondence between a point and a pixel can be controlled by the user using system settings. Since Igor stores window positions in units of points, if the user changes the number of pixels per point, the size of Igor windows in pixels will change.

Control Panel Resolution on Windows

In the past, by default, Windows screens ran at 96 DPI (dots-per-inch) resolution. In recent years, high-resolution displays, also called UltraHD displays and 4K displays, have become common.

In Igor6 and before, commands that create control panels and controls use screen pixel units to specify sizes and coordinates. This results in tiny controls on high-resolution displays.

In Igor7 and later, control panels and control sizes and coordinates can be specified in units of points instead of pixels. Since a point is a resolution-independent unit, this results in controls that are the same logical size regardless of the physical resolution of the screen.

By default, panels are drawn using pixels if the screen resolution is 96 DPI but using points for higher-DPI settings. This gives backward compatibility on standard screens and reasonably-sized controls on high-resolution screens.

You can change the way control panel coordinates and sizes are interpreted using this using:

```
SetIgorOption PanelResolution = <resolution>
```

<resolution> controls how Igor interprets coordinates and sizes in control panels. It must be one of these values:

0:	Coordinates and sizes are treated as points regardless of the screen resolution.
1:	Coordinates and sizes are treated as pixels if the screen resolution is 96 DPI, points otherwise. This is the default setting in effect when Igor starts.
72:	Coordinates and sizes are treated as pixels regardless of the screen resolution (Igor6 mode).

This setting is not remembered across Igor sessions.

The resolution for a panel is set when the panel is created so changing the panel resolution does not affect already-existing control panels.

Programmers are encouraged to test their control panels at various resolution settings.

For the most part, Igor6 control panels will work fine in Igor7 and later. However, if you have panels that rearrange their components depending on the window size, you probably have some code that uses the ratio of 72 and ScreenResolution. For example, here is a snippet from the WaveMetrics procedure Axis Slider.ipf:

```
case "Resize":
   GetWindow kwTopWin,gsize         // Returns points
                                    // Controls are positioned in pixels
   grfName= WinName(0, 1)
   V_left *= ScreenResolution / 72  // Convert points to pixels
   V_right *= ScreenResolution / 72
   Slider WMAxSlSl,size={V_right-V_left-kSliderLMargin,16}
   break
```

To make this work properly at variable resolution, we use the **PanelResolution** function, added in Igor Pro 7.00:

```
case "Resize":
   GetWindow kwTopWin,gsize         // Returns points
   grfName= WinName(0, 1)
   V_left *= ScreenResolution / PanelResolution(grfName)     // Variable
   V_right *= ScreenResolution / PanelResolution(grfName)    // resolution
   Slider WMAxSlSl,size={V_right-V_left-kSliderLMargin,16}
   break
```

The PanelResolution function, when called with "" as the parameter returns the current screen resolution or, if in Igor6 mode, 72. When called with the name of a control panel or graph, it returns the resolution that was in effect when the panel or graph was created.

If you want your procedures to work with Igor6, you should insert this into each procedure file that needs the PanelResolution function:

```
#if Exists("PanelResolution") != 3
Static Function PanelResolution(wName)    // For compatibility with Igor7
   String wName
   return 72
End
#endif
```

To find other examples, use the Help Browser to search procedure files for PanelResolution.

Pictures in Notebooks

If you want to display notebook pictures on both platforms, they must be in one of these cross-platform formats: PNG, JPEG, TIFF, SVG.

If you have pictures in other formats and you want them to be viewable on both platforms, you should convert them to PNG. Igor uses PNGs for Igor help files which need to be platform-independent.

There are two ways to create a PNG picture in an Igor notebook. You can load it from a file using Misc→Pictures and then place it in a notebook or you can convert a picture that you have pasted into a notebook using Notebook→Special→Convert to PNG.

The Convert to PNG command in the Notebook→Special menu converts the selected picture or pictures into PNG. It skips selected pictures that are already PNG or that are foreign (not native to the platform on which you are running). You can determine the type of a picture in a notebook by clicking in it and looking at the notebook status line.

The Convert to PNG dialog allows you to choose the desired resolution. Usually 4x or 8x is best.

When you create a PNG file from within Igor, from a graph window for example, you can create it at screen resolution or at higher resolution. For good quality when viewed at higher magnifications and when printed, use 4x or 8x.