

Platform-Related Issues

- Platform-Related Issues 394
- Windows-Specific Issues..... 394
- Cross-Platform File Compatibility..... 394
 - Experiment Files — Working with Earlier Versions 394
 - Crossing Platforms 394
 - Transferring Files Using File Transfer Programs..... 394
 - File Name Extensions, File Types, and Creator Codes..... 395
 - Experiments and Paths 395
 - Picture Compatibility 395
 - Page Setup Compatibility 396
 - Pre-Carbon Page Setup Records..... 397
- File System Issues 398
 - File and Folder Names 398
 - Path Separators 398
 - UNC Paths 399
 - Unix Paths 399
 - FlushFileBuffers 399
- Keyboard and Mouse Usage 400
 - Command Window Input 401
 - Other Input Issues 401
- Cross-Platform Text and Fonts 401
 - Character Set Compatibility 401
 - Text Styles 402
 - Carriage Returns and Linefeeds 402
 - Font Substitution..... 402
 - Example..... 403
- Cross-Platform Procedure Compatibility 404
 - File Paths 404
 - File Types and Extensions 404
 - Points Versus Pixels..... 406
 - Window Position Coordinates..... 407
- Notebook Issues 407
 - PNG Pictures in Notebooks 407

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 “Igor.exe”, exactly. If you change the name, Igor extensions will not work because they will be unable to find Igor.

If you press Shift while launching Igor, Igor will skip loading extensions. (This feature is mainly of interest to the programmers at WaveMetrics who launch and quit Igor dozens of times a day during development.) If you want to do this on Windows, you should wait until you see Igor’s startup window before pressing the Shift key because Windows has its own interpretation for Shift during launch of a program.

If you save an experiment while a graph, table, layout, panel or notebook window is minimized, when you reopen that experiment, the window will again be minimized. This feature is not supported for any other kinds of windows, including the command window and all procedure windows. Instead, these other types of windows are reopened in their normal size.

Cross-Platform File Compatibility

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

If you plan to use Igor on both platforms, it is a good idea to keep the same folder hierarchy for your Igor Pro files on both platforms. For example, if your Macintosh Igor files are in "hd:Igor Data Files: . . .", then it is best if you put your Windows Igor files in "C:\Igor Data Files \ . . .". Doing this will maximize the chances that Igor can find files referenced from Igor experiment files.

Experiment Files — Working with Earlier Versions

You may occasionally want to send an experiment file to a colleague who has an earlier version of Igor or open it on a computer with an earlier version of Igor.

If you use new features from a particular version of Igor Pro, you may get errors when you open an experiment file in an older version. Usually you can ignore or correct the errors and recover the file.

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 will display 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. This includes the Macintosh PICT format and its variants when displayed on Windows and the Windows Metafile and Enhanced Metafile formats when displayed on Macintosh. The EPS, PNG, JPEG, and TIFF 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

On Windows, the file name extension indicates the nature of a file. When you double-click a file, Windows uses the extension to determine which program to launch. Mac OS 9 used the Mac-specific file type property to determine the nature of the file and the Mac-specific file creator code to determine which program to launch. Mac OS X still supports the file type and creator code properties but de-emphasizes them in favor of the file name extension. For this reason it is best to use the correct file name extension regardless of platform.

The file name extension and corresponding Macintosh file type for Igor Pro files are:

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
.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-37. 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 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 is unable to find a folder that is referenced by a full path, it looks for the folder in the same place in the hierarchy, but on the root volume containing the experiment file. If this fails, it looks for the folder in the same place in the hierarchy, but on the root volume containing the Igor application file. Also, if the path points inside the Igor Pro Folder, then Igor looks for the folder in the same place in the hierarchy, but inside the Igor Pro Folder containing the Igor application file.

If all of these methods fail, Igor displays a dialog asking you to locate the folder.

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.

Chapter III-15 — Platform-Related Issues

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

Format	How To Create	Notes
PICT	Paste or use Misc→Pictures	Macintosh only
PDF	Paste or use Misc→Pictures	Macintosh only
EMF (Enhanced Metafile)	Paste or use Misc→Pictures	Windows only
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. Requires PostScript printer. A screen preview is displayed on screen.

PICT was the standard format for Mac OS 9 graphics but has been supplanted by PDF on Mac OS X. EMF is the standard format for Windows graphics. PICT, PDF, EMF and BMP are platform-dependent and will display as gray boxes if you move the Igor experiment to the other platform. The other formats are platform-independent.

Although Igor does not display nonnative 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 Window 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, and the Windows metafile will be displayed correctly.

If you want PDF, PICT, EMF, or BMP/DIB 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.

Note: 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. However, this takes a lot of memory during the conversion and when the PNG is displayed. Because PNGs are compressed, they usually do not require excessive disk space when saved.

The ability to store pictures in JPEG, TIFF and EPS formats was added in Igor Pro 5. If you create an experiment with pictures in these formats, they will display as gray boxes if opened in older versions of Igor.

Prior to version 3.1, Igor stored graph and page layout pictures on the Macintosh in the experiment file's resource fork. When you transfer a file created by an Igor version older than 3.1 from the Macintosh to a PC, the resource fork is lost and Igor will display a placeholder rectangle in place of the picture.

Page Setup Compatibility

Page setup records store information regarding the size and orientation of the page. Page setups contain platform- and printer-dependent data. They are most important in regards to page layout windows but also affect printing of other kinds of windows.

On Macintosh, Igor stores page setups in experiment files, notebook files, and procedure files. 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 graphs and single page setup for all tables.

On Windows, Igor stores page setups the same as on Macintosh except that it does not store page setups in plain notebook or procedure files. This is true whether the file is embedded in a packed experiment file or is a stand-alone file. When you open a plain notebook or procedure file on Windows, Igor creates a new page setup record by copying the preferred page setup record as set by the Capture Notebook Prefs or Capture Procedure Prefs dialogs.

When you transfer an experiment from one platform to another, page setup records are only partially preserved. Igor attempts to preserve the page orientation and margins.

Prior to version 3.1, Igor stored page setup records on the Macintosh in the experiment file's resource fork. When you transfer a file created by an Igor version older than 3.1 from the Macintosh to a PC, the resource fork is lost and Igor has no way to know the orientation of the page setups. Therefore, when opening one of these old Macintosh experiment files on Windows, any page layouts in the experiment will use the page setup that you captured using the Capture Layout Prefs dialog or a default page setup if you have not captured a preferred page setup.

Pre-Carbon Page Setup Records

Carbon is a set of routines that Apple created for the transition from Macintosh OS 9 to OS X. The page setup record now used by the Macintosh operating system is called a Carbon page setup record. Prior to that the operating system used a different type of page setup record which we will call an "old Macintosh page setup record."

Experiments saved on Macintosh by pre-Carbon versions of Igor Pro (prior to version 4.05A) contain old page setup records. When you open one of these experiments under OS X, the old page setup record is passed to a system routine that is supposed to convert it to a Carbon page setup record. This usually works correctly but sometimes it corrupts the page setup record. The corruption usually appears as a nonsensical scaling value in the Page Setup dialog for the affected window. If the window is a page layout window, this results in a page that is grossly too small or too big. You can usually fix it by entering 100% for the scaling value in the page setup dialog.

Igor Pro 6 attempts to detect and fix this corruption if the scaling value is less than 50% or greater than 200%. It repairs the page setup record by replacing it with a new default page setup record and then sets the orientation of the new page setup record to match the orientation of the old record. All other properties, such as the scaling, of the old page setup record are lost.

If corruption occurs and the automatic repair is not successful, you can use the **SetIgorOption** operation (see page V-562) to control the old page setup record conversion. The command format is:

```
SetIgorOption RepairOldPageSetups = val
```

The parameter *val* is one of the following:

<i>val</i>	Effect
0	Never repair old page setup records.
1	Repair if the old page setup appears corrupted (default).
2	Always repair old page setup records.
3	Always present a page setup dialog.
4	Always repair and then present a page setup dialog.

The effect of SetIgorOption lasts only until you quit Igor.

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. Furthermore, you should not use "high ASCII" characters (see **Character Set Compatibility** on page III-401), because they don't translate across platforms. Also, don't use period except before a file name extension.

Prior to Igor Pro 6.1, the Macintosh version of Igor could not handle file or folder names exceeding 31 characters. Igor Pro 6.1 and later support long file names (up to 255 characters) on Macintosh as well as Windows. Most WaveMetrics XOPs now also support long file names on Macintosh. However XOP file names are still limited to 31 characters plus the ".xop" extension.

The following Igor Pro features will not work with long file names on Mac OS X because Igor calls Apple routines that do not support long file names for these features:

- NewMovie
- ImageSave when using QuickTime
- ImageLoad when using QuickTime

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

```
hd:Igor Pro 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 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>really 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-style path, even on Windows:

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

Igor will convert the Macintosh-style path to a Windows-style path before using it. This avoids the backslash ambiguity.

For a complete list of escape sequences, see **Escape Characters 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-472 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 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 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 paths instead.

FlushFileBuffers

On Windows, by default Igor calls the Windows FlushFileBuffers routine before closing a file. FlushFileBuffers flushes data cached in a hard drive's circuitry for the file being closed, ensuring that the data is written to disk. This guarantees the file's integrity in the event of a power failure.

When writing a lot of small files one right after another, for example when saving an unpacked experiment with a very large number of waves or data folders, calling FlushFileBuffers can adversely affect perfor-

Chapter III-15 — Platform-Related Issues

mance. In this case, you can use `SetIgorOption` to disable calling `FlushFileBuffers`. This is rarely necessary and should be done only in the circumstances described in this paragraph.

To turn `FlushFileBuffers` off execute:

```
SetIgorOption UseFlushFileBuffers=0           // Turn off
```

To turn `FlushFileBuffers` on execute:

```
SetIgorOption UseFlushFileBuffers=1         // Turn on (default)
```

To query the state execute:

```
SetIgorOption UseFlushFileBuffers=?; Print V_Flag // Query
```

This does nothing on Macintosh.

`SetIgorOption` is usually called from the command line. It can not be executed from a user-defined function. Its effect lasts until Igor quits.

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	Macintosh	Windows
Shift	Shift	Return	Enter
Command	Ctrl	Enter	Enter
Option	Alt	Control-click	Right-click
Control	<not mapped>		

There are a few exceptions to this mapping. For example, Option-Tab enters a tab into a text wave in a table. Alt-Tab does not do this on Windows because Alt-Tab is reserved by the Windows operating system.

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

You can quickly move the command window (or any built-in Igor window) to its preferred position. On Macintosh, press Option and click the zoom button. On Windows, press Alt and click the maximize button.

Other Input Issues

The accelerators (keyboard shortcuts) for Indent Left and Indent Right in the Edit menu are Command-[and Command-] on Macintosh but are Ctrl+Shift+L and Ctrl+Shift+L R on Windows. This is because Windows does not allow using Ctrl+[or Ctrl+] as an accelerator.

Here are some shortcuts related to the Igor help system:

Action	Macintosh	Windows
Invoke Igor Help Browser	Help	F1
Insert operation or function template	Shift-Help	Ctrl+F1
Go to help for operation or function	Shift-Option-Help	Ctrl+Alt+F1

Cross-Platform Text and Fonts

Character Set Compatibility

The Macintosh and Windows character encodings are the same for the common characters, such as upper and lower case letters, numbers and common punctuation. These are sometimes called the “low ASCII” characters because they have ASCII codes below 128 and appear in the first half of the ASCII character encoding table. The remaining characters, such as accented vowels, bullets and other symbols, are called “high ASCII” characters. The encoding for high ASCII characters is different on the Macintosh and Windows. For example, the character code for a bullet on the Macintosh is the character code for a yen symbol on Windows.

In most cases, Igor does an automatic translation to compensate for these encoding differences. For example, when Igor opens a formatted notebook file or a packed experiment file containing notebooks and procedures, Igor automatically translates. There are some characters for which no translation is possible because the characters are missing from one platform or the other. For example, the Macintosh has a “Š” character but Windows does not. For these characters, Igor does not do any translation. The character will appear incorrect on the nonoriginating platform but will appear correct if the file is moved back to the originating platform.

For some fonts, this translation causes a problem, because the fonts do not use the normal character encodings. Symbol font is an example. It has the same encoding on both platforms and therefore Igor does not do any translation.

When Igor opens a plain text notebook or a procedure file that is a stand-alone file (i.e., is not stored in a packed experiment file) it has no way to know which platform the file came from and therefore can not do character translation. This may result in funny characters appearing, such as a yen symbol where you expect a bullet symbol. If you use such characters in procedure files, this problem may cause errors when you compile or execute the procedures. The only solution is to avoid using high ASCII characters in plain text files.

Like the Symbol font, Asian fonts use the same character encoding on Macintosh and Windows and so Igor does not translate text governed by an Asian font. When opening an experiment using Asian text, it is

Chapter III-15 — Platform-Related Issues

important that your preferred procedure file font be an Asian font. This is the factory default if you are running on an Asian version of the Macintosh or Windows operating system. This is important because Igor uses the preferred procedure file font in cases where it has no other way to determine which font should be used for the file, such as when you open a stand-alone procedure file on Macintosh or any procedure file (even packed into an experiment file) on Windows.

Text Styles

The outline and shadow text styles are available on Macintosh only. On Windows, these styles have no effect and Igor disables Outline and Shadow items in menus and Outline and Shadow checkboxes in dialogs.

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 Macintosh), 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 CR on Mac OS and CRLF on Windows. As of Igor Pro 5.04, 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.

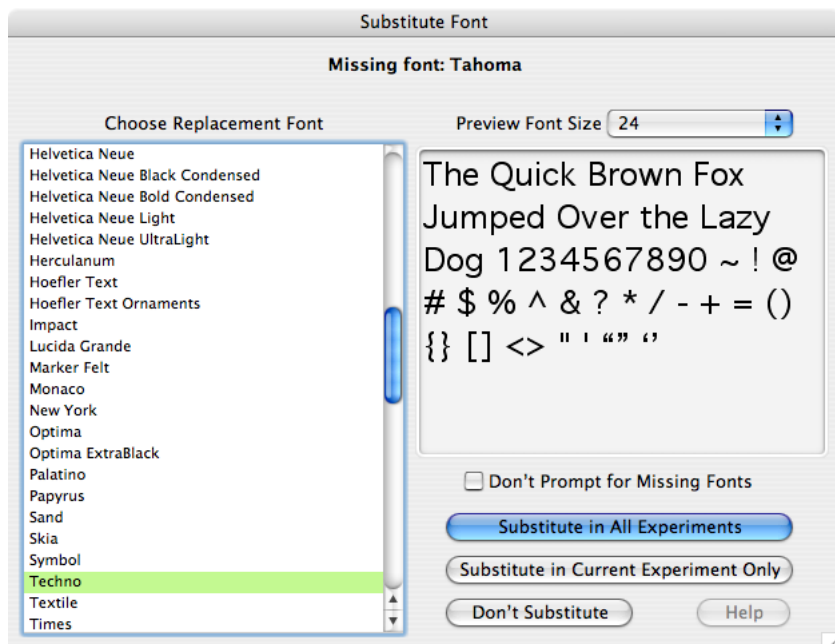
By default, the **FReadLine** operation treats CR, LF, or CRLF as terminators. Use this to write a procedure that can read lines from a text file without caring whether it is a Macintosh, Windows, or Unix file.

Font Substitution

When a font specified in a command or document is not installed Igor will apply 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:



Note: 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, which is also present in the Edit Font Substitutions Dialog.

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).

If you prefer to replace Geneva with Verdana, you can use the Edit Font Substitution Dialog (accessed from the Misc menu) to edit the user-level substitutions, which take precedence over any built-in 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 (and 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.

Example

Suppose an opened experiment's recreation procedures contain commands like these:

```
Display                               // create a blank graph
ModifyGraph gFont="this font doesn't exist"
```

Suppose also that we haven't created an entry in the Edit Font Substitution dialog for a missing font named "this font doesn't exist".

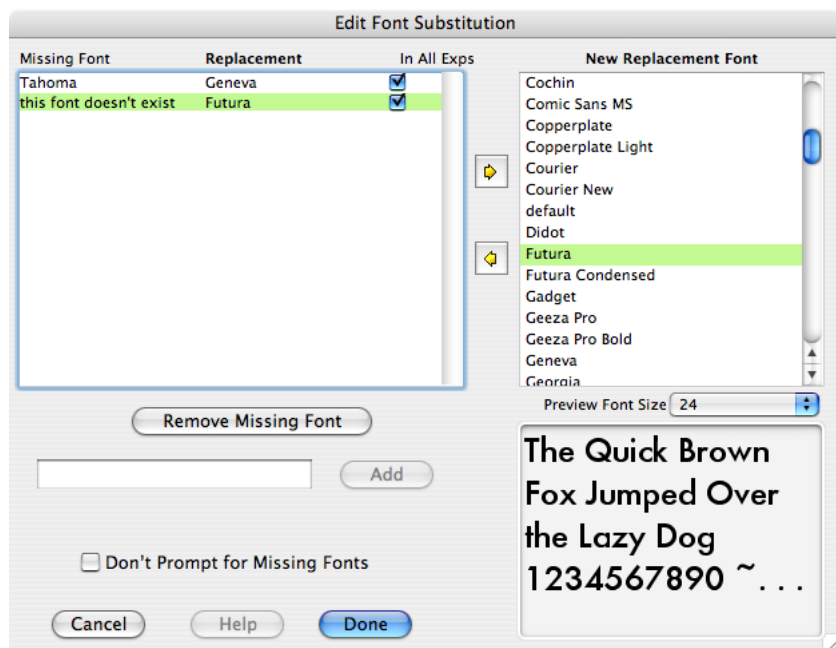
Because the `ModifyGraph gFont` command refers to a nonexistent font, the Substitute Font Dialog appears. Suppose we select a replacement font of Futura.

There are three consequences:

- 1) The recreation macro of the created graph looks like this:

```
Window Graph0() : Graph
    PauseUpdate; Silent 1          // building window...
    Display /W=(5,44,400,252)
    ModifyGraph gFont="Futura"
EndMacro
```

- 2) If *another* `ModifyGraph gFont="this font doesn't exist"` command is executed, the Substitute Font dialog does *not* appear, but silently converts the font to Futura.
- 3) Igor adds the font substitution to the Missing Font list you see in the Edit Font Substitution Dialog:



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, using the **IgorInfo(2)** function, and act accordingly.

File Paths

As described under **Path Separators** on page III-398, the Macintosh uses colons to separate elements of a file path while Windows uses backslashes. So that you can write a single procedure that works on both platforms, 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-398. 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-472 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 controls the icon displayed for the file and which programs can open the file.

In Mac OS X, the file type can be represented by the file type property or by the file name extension. Either or both can be used, with the extension preferred.

On Windows, the file type is indicated by the file name extension and files do not have file type properties.

For the most part, in Igor programming you do not need to worry about file types or extensions. However, there are some Igor operations and functions, such as **Open** and **IndexedFile**, which take a string parameter containing a file type or list of file types. For example, these commands present a dialog from which a user can choose plain text files or Igor formatted notebook files:

```
Variable fileRef
Open/R/T="TEXTWMT0" fileRef as ""
Close fileRef
```

If you use these operations and functions in procedures, then you need to understand how Igor maps file name extensions to file types.

Igor associates a file type with a file name extension. For example, Igor considers “.txt” files to be of type TEXT and considers “.ifn” files to be of type WMT0. The following table shows how Igor maps from extension to file type:

Windows Extension	Macintosh File Type	What's in the File
.uxp	IGSU	Unpacked experiment file
.uxt	IGSS	Unpacked experiment template (stationery)
.pxp	IGsU	Packed experiment file
.pxt	IGsS	Packed experiment template (stationery)
.ibw	IGBW	Binary wave file
.itx	IGTX	Igor Text file
.xop	IXOP	Igor extension file
.ibv	IGB-	Igor binary variable
.ibt	IGT-	Igor binary misc things
.txt	TEXT	Text file
.ifn	WMT0	Igor formatted notebook file
.ift	WMTS	Igor formatted notebook template (stationery)
.eps	EPSF	Encapsulated PostScript file
.rtf	RTF	Rich Text file
.tif	TIFF	TIFF file
.jpg	JPEG	JPEG file
.tga	TPIC	Targa file
.gif	GIFf	GIF file
.sgi	SGI	Silicon Graphics file
.png	.PNG	Portable Network Graphics file
.psd	8BPS	Photoshop file
.pct	PICT	Macintosh PICT file
.bmp	BMPp	Windows bitmap file
.*	????	Any type file

In addition to the standard Windows extensions listed above, Igor also recognizes files with the “.bwav” and “.away” extensions. These are recognized as equivalent to “.ibw” and “.itx” for backward compatibility with old versions of Igor.

Chapter III-15 — Platform-Related Issues

Igor also maps the following extensions to the associated file type. However, the file types listed in this table are fictitious. They are not file types that are really used on Macintosh but they still work, for example, in the Open operation.

Windows Extension	Fictitious Macintosh File Type	What's in the File
.ihf	IHLP	Igor help file (WMT0 files on Macintosh)
.ipf	IPRC	Procedure file (TEXT files on Macintosh)
.dat	.DAT	Data file
.csv	.CSV	Comma-separated file
.emf	.EMF	Enhanced metafile
.wmf	.WMF	Windows metafile
.bmp	.BMP	Bitmap file
.png	.PNG	PNG file
.htm	HTML	HTML file

Igor maps all extensions not listed above to the file type '????' which signifies “unknown file type”. This includes files that have no extension.

The `TextFile` function will recognize only files of type TEXT. On Windows, a file must have the “.txt” extension to be of type TEXT.

The `IndexedFile` function requires that you supply a file type or a file extension as a parameter. Thus, the following commands will print the name of the first file in the Igor directory whose type is TEXT (extension is “.txt”):

```
Print IndexedFile(Igor, 0, "TEXT")
Print IndexedFile(Igor, 0, ".txt")
```

This will print the name of the second file of any type:

```
Print IndexedFile(Igor, 1, "????")
```

Points Versus Pixels

Most measurements of length in Igor are in terms of points. A point is roughly 1/72 of an inch.

A pixel is the area taken up by the smallest displayable dot on an output device such as a CRT or a printer. The physical width and height of a pixel depends on the device. CRTs typically display 60 to 120 pixels per inch (commonly called “dots per inch” or DPI). Printers typically display 150 to 2400 DPI.

The physical size of a screen pixel can vary from one CRT to another. The size of a pixel on a given CRT can often be adjusted by the user by twiddling knobs on the back of the CRT or using buttons on the front panel. Because of this, software programs don't know about or worry about the physical size of a pixel but instead deal with an assumed or “logical” size.

On the Macintosh, the logical width and height of a screen pixel is always 1/72 of an inch, which equates to a logical resolution of 72 DPI. The user can change the physical resolution but not the logical resolution. If a monitor's physical resolution is close to 72 DPI, there will be a pretty good match between the logical and physical sizes. On such monitors, if you make a graph 5 inches (360 points) wide, it will appear roughly 5 inches wide.

On Windows, the default logical width and height of a screen pixel is 1/96 of an inch which equates to a logical resolution of 96 DPI. The user can change both the physical resolution and the logical resolution using the Windows Display Properties control panel. It is common to have a significant discrepancy between the logical and physical sizes. In other words, if you make a graph 5 inches (360 points) wide, it may appear 6 or 7 inches wide. If you want the physical size of an object on the screen to match its logical size, you need to find an appropriate combination of physical and logical resolutions.

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, PlayMovie and MoveWindow.

The exception is the control panel window. To make it easier to create control panels that look the same on Macintosh and Windows, the NewPanel operation interprets its /W coordinates as pixels and operations that create or modify controls also interpret coordinates as pixels.

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 the 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 the Macintosh, a point is always interpreted to be one pixel.

On Windows, the correspondence between a point and a pixel can be controlled by the user, as described under **Points Versus Pixels** on page III-406. 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. The exception is that control panels will not change because Igor interprets their coordinates as pixels.

Notebook Issues

On Macintosh, Igor stores the settings (font, size, style, etc.) for a plain text file in the file's resource fork. The data fork contains just the plain text. On Windows, text files have no resource fork. Therefore there is no way for Igor to store settings for a plain text file on Windows. When you open a plain text notebook or an experiment containing a plain text notebook on Windows, Igor uses preferences to set the notebook's text format. Thus, text format changes that you make to a plain text notebook are lost on Windows unless you capture them as your preferred text format.

PNG 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. Although EPS is a cross-platform format, the EPS screen preview on Macintosh is PICT, which is not cross-platform.

If you have pictures in PICT or EMF format 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. If you select just one picture and do the conversion, Igor will let you undo it. However, if you select anything other than a single picture, the conversion will not be undoable. You can interrupt a conversion by pressing Command-period (*Macintosh*) or Ctrl+Break (*Windows*) and holding until Igor stops the conversion.

Chapter III-15 — Platform-Related Issues

When you create a PNG file from within Igor, you can create it at screen resolution or at higher resolution. If you want crisp screen display when viewed at 100 percent magnification, use screen resolution when you create the PNG file. If you want better quality when viewed at higher magnifications and when printed, use higher than screen resolution when you create the PNG file. Using higher resolution requires a lot of memory during the conversion and when the picture is displayed but usually does not require excessive disk space when saved because PNGs are compressed. Once you have created a PNG file, you can load it into another program or into Igor using Misc→Pictures.

The next three paragraphs explain how Igor treats screen resolution PNGs differently from high resolution PNGs. You can skip this if you don't care about the technical details.

Normally, Igor stretches pictures if necessary to preserve the logical size of the pictures at different screen resolutions. However, PNG, being a bitmap type of picture, looks best when displayed without stretching. Therefore, Igor treats PNGs in notebooks specially. If the PNG was created at screen resolution, Igor displays the PNG without stretching. This gives a crisp, undistorted display when viewed at 100 percent magnification, but the physical size of the picture is not preserved. This is usually what you want for screen dumps and for graphics intended mostly for viewing on screen at 100% magnification as opposed to printing or viewing on screen at higher magnifications.

For example, suppose you create a 5 inch by 4 inch graph on the Macintosh and export that graph as a screen-resolution PNG. You paste the PNG into a notebook. On the Macintosh, the screen resolution is 72 dpi, so the PNG will be 72 dpi. Thus, the dimensions of the PNG in pixels is 360 by 288 (5*72 by 4*72). If you save the notebook and open it on Windows, Igor notices that the PNG is screen resolution and will therefore displays it using 360 by 288 pixels. This means that Igor does not need to stretch the PNG so it displays crisply. However, because Windows screen resolution is typically 96 dpi, the logical size of the PNG on Windows will be smaller than 5 inches by 4 inches by a factor of 72/96.

If instead of creating a screen-resolution PNG, you create a high-resolution PNG, Igor will stretch the PNG by a factor of 96/72 on Windows. This makes the logical size of the PNG the same on both platforms but it will be somewhat distorted on Windows because of the stretching.