

## Exporting Graphics (Macintosh)

Overview .....	98
Macintosh PICT Format .....	98
PDF Format .....	99
Encapsulated PostScript (EPS) Format .....	99
Platform-Independent Bitmap Formats .....	99
Choosing a Graphics Format .....	100
Exporting Graphics Via the Clipboard .....	100
Exporting Graphics Via a File .....	101
Exporting a Graphic File for Transfer to a Windows Computer .....	101
Exporting a Section of a Layout .....	102
Exporting Colors .....	102
Exporting and Printing Graphs of Large Data Sets .....	102
Graphs in Page Layouts .....	102
Font Embedding .....	102
PostScript Font Names (OS X) .....	103

### Overview

This chapter discusses exporting graphics from Igor graphs, page layouts and tables to another program on Macintosh. You can export graphics through the Clipboard by choosing Edit→Export Graphics, or through a file, by choosing File→Save Graphics.

Igor Pro supports a number of different graphics export formats. You can usually obtain very good results by choosing the appropriate format, which depends on the nature of your graphics, your printer and the characteristics of the program to which you are exporting.

Unfortunately, experimentation is sometimes required to find the best export format for your particular circumstances. This section provides the information you need to make an informed choice.

This table shows the available graphic export formats:

Export Format	Export Method	Notes
Quartz PDF	Clipboard, file	The standard format for OS X. Best format for general use. Generated via the operating system and consequently more capable than Igor PDF.
LowRes PDF	Clipboard, file	Mostly just a compatibility placeholder for the legacy Macintosh PICT format. May have specialized uses but generally should not be used. Use Quartz PDF instead.
Igor PDF	Clipboard, file	PDF generated by Igor's own code rather than by the OS. May have specialized uses but generally should not be used. Use Quartz PDF instead.
Bitmap PICT	Clipboard, file	Legacy Macintosh-specific vector format. Largely obsolete as of Mac OS X.  Resolution is 72 dpi. Expanded export can be used with some programs to simulate higher resolution.
EPS (Encapsulated Postscript)	File only	Platform-independent except for the screen preview.  Supports high resolution.  Useful only when printing on a PostScript printer, creating a PDF file, or exporting to PostScript-savvy program (e.g., Adobe Illustrator, Tex).
PNG (Portable Network Graphics)	Clipboard, file	Platform-independent bitmap format.  Uses lossless compression. Supports high resolution.
JPEG	Clipboard, file	Platform-independent bitmap format.  Uses lossy compression. Supports high resolution.
TIFF	Clipboard, file	Platform-independent bitmap format.  Supports high resolution but not compression.
QuickTime Formats	File only	Additional bitmap formats are added by QuickTime, if installed. QuickTime formats appear in lower half of Format menu in Save Graphics dialog.

---

### Macintosh PICT Format

PICT is the legacy pre-OS X Macintosh graphics format. In Igor Pro 6.1, it has been replaced by the PDF format. In an emergency, when you are exporting to an older program that does not support PDF (such as Microsoft Office prior to 2008,) you can cause Igor to revert to the old QuickDraw graphics mode. See **Graphics Technology** on page III-421 for details.

### PDF Format

PDF (Portable Document Format) is Adobe's platform-independent vector graphics format that has been adopted by Apple as the standard graphics format for OS X. This is the best format as long as your destination program supports it.

The Quartz PDF format is generated by the operating system and consequently is more capable than Igor's existing native PDF generator. For example, it does a better job of embedding fonts and fully supports imported pictures.

The Igor PDF format is generated by Igor's own code rather than by the OS. Due to the following limitations, it should not be used unless you need to export in CMYK color mode (See **Exporting Colors** on page III-102 for details.) Limitations are:

- If the window contains any drawings imported into Igor from other programs, they will be rendered in the PDF as opaque bitmap images.
- You will need to pay attention to fonts. See **Font Embedding** on page III-102 for details.

### Encapsulated PostScript (EPS) Format

Encapsulated PostScript is a widely-used, platform-independent vector graphics format consisting of PostScript commands in plain text form. It usually gives the best quality, but it works only when printed to a PostScript printer or exported to a PostScript-savvy program such as Adobe Illustrator. You should use only PostScript fonts (e.g., Helvetica).

Because PostScript is very complex, few programs can display it on screen. Consequently EPS includes an optional screen preview. In most programs, you need to print a document containing the EPS in order to see what you really have, as opposed to the preview that you see on screen.

On Macintosh, the EPS screen preview format is standard PICT, stored in the resource fork of the EPS file. If you transfer a Macintosh EPS file to Windows, the screen preview will be lost because Windows does not support the resource fork. So you will see a plain box instead of the preview on the screen of the Windows program, but the EPS should print correctly.

On Windows, the screen preview format is TIFF and is embedded in the EPS file. If you transfer it to Macintosh, the preview may or may not be understood, depending on the program into which you are importing. Embedding the TIFF preview makes the Windows EPS file a binary file, which is not editable with a text editor.

Some poorly written applications are confused by the screen preview. They ignore the EPS rules and use the size of the preview image rather than the PostScript bounding box, resulting in improper recreation of the EPS graphic. If you get unsatisfactory results, try using Igor's Suppress Preview option. The resulting EPS will display as a plain box in most programs but will print correctly.

EPS files normally use the RGB encoding to represent color but you can also use CMYK. See **Exporting Colors** on page III-102 for details.

Igor Pro exports EPS files using PostScript language level 2. This allows much better fill patterns when printing and also allows Adobe Illustrator to properly import Igor's fill patterns. For backwards compatibility with old printers, you can force Igor to use level 1 by specifying `/PLL=1` with the SavePICT operation.

If the graph or page layout that you are exporting as EPS contains a non-EPS picture imported from another program, Igor exports the picture as an image incorporated in the output EPS file.

Igor Pro 5.02 and later can embed TrueType fonts as outlines. See **Font Embedding** on page III-102 for details.

### Platform-Independent Bitmap Formats

PNG (Portable Network Graphics) is a platform-independent bitmap format that uses lossless compression and supports high resolution. It is a superior alternative to JPEG or GIF. Although Igor can export and import PNG via the Clipboard, not all programs can paste PNG from the Clipboard.

## Chapter III-5 — Exporting Graphics (Macintosh)

JPEG is a lossy image format whose main virtue is that it is accepted by all web browsers. However, all modern web browsers support PNG so there is little reason to use JPEG. Although Igor can export and import JPEG via the Clipboard, not all programs can paste JPEGs.

TIFF is an Adobe format often used for digital photographs. Igor's implementation of TIFF export does not use compression. TIFF files normally use the RGB scheme to specify color but you can also use CMYK. See **Exporting Colors** on page III-102 for details. There is no particular reason to use TIFF over PNG unless you are exporting to a program that does not support PNG. Igor can export and import TIFF via the Clipboard and most OS X programs can import TIFF.

A number of additional bitmap file formats are supported through Apple's QuickTime. The formats added by QuickTime are listed in the bottom half of the Format pop-up menu in the Save Graphics dialog.

### Choosing a Graphics Format

Because of the wide variety of types of graphics, destination programs, printer capabilities, operating system behaviors and user-priorities, it is not possible to give definitive guidance on choosing an export format. But here is an approach that will work in most situations.

If the destination will accept PDF, then that is probably your best choice because of its high-quality vector graphics and platform-independence.

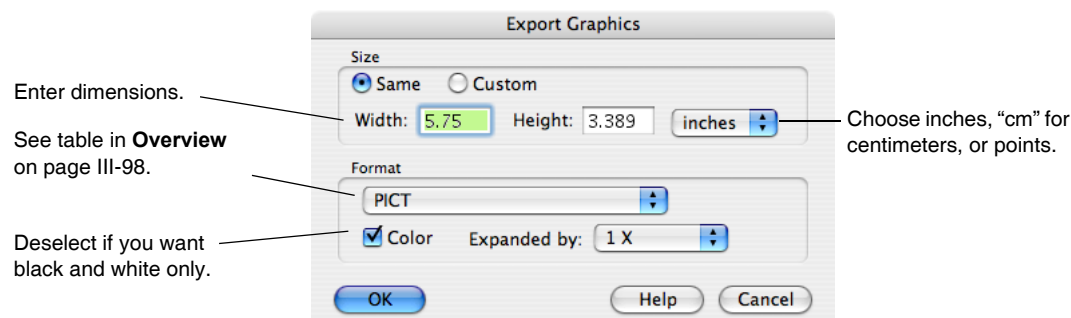
Encapsulated PostScript (EPS) is also a very high quality format and is the format of choice if you are:

- Exporting to a word processor for printing on a PostScript printer.
- Exporting to a word processor for creating a PDF file.
- Exporting to a PostScript-savvy drawing program such as Adobe Illustrator.

If EPS and PDF are not appropriate, your next choice would be a high-resolution bitmap. The PNG format is preferred because it is platform-independent and is compressed. If the application to which you are exporting does not support PNG, your next choice would be TIFF or JPEG.

### Exporting Graphics Via the Clipboard

To export a graphic from the active graph, page layout or table window via the Clipboard, choose Edit→Export Graphics. This displays the Export Graphics dialog. For a graph, it looks like this:



When you click the OK button, Igor will copy the graph, page layout, or table to the Clipboard. You can then switch to another program and do a paste.

For a page layout or table, the dialog is the same except that you can't enter the width and height.

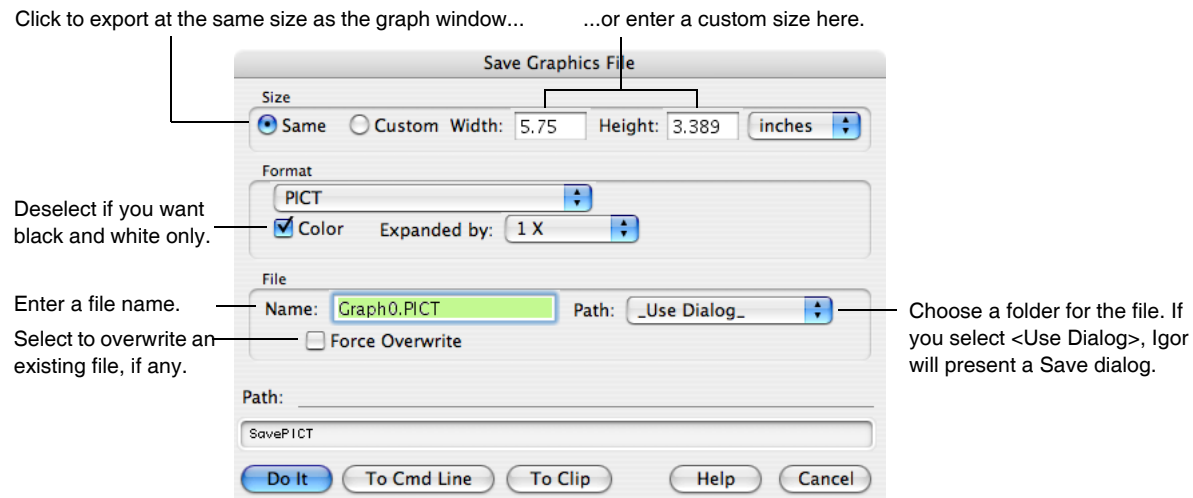
When a graph or page layout is active and in operate mode, choosing Edit→Copy copies to the Clipboard whatever format was last used in the Export Graphics dialog. (This is new as of Igor Pro 6.01. Previously the format was always a 1X standard vector PICT.) For a table, Edit→Copy copies the selected numbers to the Clipboard and does not copy graphics.

When a page layout has an object selected or when the marquee is active, choosing Edit→Copy copies an Igor object in a format used internally by Igor along with a 1x standard vector PICT and does not use the format from the Export Graphics dialog

Igor can export PNG images to the Clipboard and can then paste them back in. On the Macintosh, the Clipboard type is 'PNGf' but because there is no standard for PNG on the Clipboard it is therefore unlikely that other programs can import them except as files.

## Exporting Graphics Via a File

To export a graphic from the active graph, page layout or table window via a file, choose File→Save Graphics. The Save Graphics File dialog for a graph looks like this:



The controls in the Format area of the dialog change to reflect options appropriate to each export format.

When you click the Do It button, Igor will write the graphic to a file. You can then switch to another program and import the file.

If you select <Use Dialog> from the Path list, Igor will present a Save dialog from which you can choose a folder in which to save the file.

The controls in the Format area of the dialog change to reflect options appropriate to each export format.

## Exporting a Graphic File for Transfer to a Windows Computer

The best method for transferring Igor graphics to a Windows computer is to transfer the entire Igor experiment file, open it in Igor for Windows, and export the graphic via one of the Windows-compatible methods available in Igor for Windows.

If your graph or layout contains embedded pictures in PDF or PICT format, you will need to convert them to the cross-platform format, PNG, because PDFs and Macintosh PICTs are not displayed in Igor for Windows. If any embedded pictures are in JPEG, TIFF, or EPS formats, these will work without conversion. See Chapter III-15, **Platform-Related Issues**, especially the section **Picture Compatibility** on page III-395.

If you don't have a copy of Igor for Windows available, you have these choices:

1. Export PDF if the destination program can accept it.
2. Export an EPS file. This works only if the Windows program can import EPS files, and requires that it be printed on a PostScript printer.  
Furthermore, the PICT screen preview that Igor puts into a Macintosh EPS file is not displayable on Windows. Use the Suppress Preview checkbox to eliminate the preview. This will render the graphic unviewable on-screen, but it will still print (on a PostScript printer) just fine.
3. Export a PNG.

### Exporting a Section of a Layout

To export a section of a page layout, use the marquee tool to identify the section and then choose Export Graphics (Edit menu) or Save Graphics (File menu). If you don't use the marquee, Igor exports the area of the layout that is in use plus a small margin.

### Exporting Colors

The PDF, EPS and TIFF graphics formats normally use the RGB scheme to specify color. Some publications require the use of CMYK instead of RGB, although the best results are obtained if the publisher does the RGB to CMYK conversion using the actual characteristics of the output device. For those publications that insist on CMYK, you can use the SavePICT /C=2 flag

### Exporting and Printing Graphs of Large Data Sets

When exporting or printing an image plot, Igor normally uses a fast algorithm but has to resort to a much slower, memory-intensive algorithm if the image contains holes (the displayed matrix wave contains NaNs) or if it is displayed on a nonuniform grid (you specified X and Y coordinate waves when creating the plot).

The best method of exporting a graph for placement in another program is normally a vector format such as HiRes PICT or EPS. However, graphs containing large images or very large waveforms may take a long time to export or print. You may be able to solve this by using a high-resolution bitmap format for such graphs. This has the added advantage of working well with non-PostScript printers.

### Graphs in Page Layouts

If you discover that a page layout is taking much too long to print, you can print graph objects in the layout layer of the layout using a high-resolution bitmap rather than the usual vector method. Use this only in an emergency when a printer driver has bugs that affect normal operations and when printing graphs with very large numbers of data points. There are drawbacks to the bitmap method. A large amount of memory will be needed and patterns will be too small to be useful. Also, the quality of lines (especially dashed lines) may be degraded.

To force Igor to print graph objects in a page layout using the bitmap method, execute the following on the command line:

```
Variable/G root:V_PrintUsingBitmap= 1
```

You may want to set this variable to zero after printing a problem layout so as not to affect other layouts in the same experiment.

### Font Embedding

You can embed TrueType fonts in EPS files and in PDF files. This means you can print EPS or PDF files on systems lacking the equivalent PostScript fonts. This also helps for publications that require embedded fonts.

Font embedding is done automatically for the Quartz PDF format and you do not need to bother with this section unless you are using EPS or Igor PDF formats.

There are three levels of font embedding: No embedding, embed only nonstandard fonts, and embed all fonts. For most purposes, embed only nonstandard fonts is the best choice.

Igor embeds TrueType fonts as synthetic PostScript Type 3 fonts derived from the TrueType font outlines. Only the actual characters used are included in the fonts and only single byte fonts are supported.

You should not use font embedding if you plan on exporting to a drawing program such as Adobe Illustrator and wish to edit the text in that program.

Not all fonts and font styles on your system can be embedded. Some fonts may not allow embedding and others may not be TrueType or may give errors. Be sure to test your EPS files on a local printer or by import-

ing into Adobe Illustrator before sending them to your publisher. You can test your PDF files with Adobe Reader. You can also use the “TrueType Outlines.pxp” example experiment to validate fonts for embedding. You will find this experiment file in your Igor Pro Folder in the “Examples:Testing & Misc:” folder.

For EPS, the “embed only nonstandard fonts” method determines if a font is nonstandard by attempting to look up the font name in the TTPSFNames table described in **PostScript Font Names (OS X)** on page III-103 after doing any font substitution using the TTtoPS table. In addition, if a nonplain font style name is the same as the plain font name, then embedding is done. This means that standard PostScript fonts that do not come in italic versions (such as Symbol), will be embedded for the italic case but not for the plain case.

For PDF, “embed only nonstandard fonts” embeds fonts other than the basic fonts guaranteed by the PDF specification to be built-in to any PDF reader. Those fonts are Helvetica and Times in plain, bold, italic and bold-italic forms as well as Symbol and Zapf Dingbats only in plain style. If embedding is not used or if a font can not be embedded, fonts other than those just listed will be rendered as Helvetica and will not give the desired results

## PostScript Font Names (OS X)

When generating PostScript, Igor needs to generate proper PostScript font names. This presents problems under Macintosh OS X. Igor also needs to be able to substitute PostScript fonts for non-PostScript fonts.

If you use only the basic fonts in the following table or if you use **Font Embedding** on page III-102, then you do not have to read any further. Igor has built-in PostScript font names for these as well as a built-in translation table that you use to specify standard system TrueType fonts but get proper PostScript fonts when exporting. The built-in name translations are:

TrueType Name	PostScript Name
Helvetica	Helvetica
Arial	Helvetica
Helvetica-Narrow	Helvetica-Narrow
Arial Narrow	Helvetica-Narrow
Palatino	Palatino
Book Antiqua	Palatino
Bookman	Bookman
Bookman Old Style	Bookman
Avant Garde	AvantGarde
Century Gothic	AvantGarde
New Century Schlbk	NewCenturySchlbk
Century Schoolbook	NewCenturySchlbk
Courier	Courier
Courier New	Courier
Zapf Chancery	ZapfChancery
Monotype Corsiva	ZapfChancery
Zapf Dingbats	ZapfDingbats
Monotype Sorts	ZapfDingbats
Symbol	Symbol

## Chapter III-5 — Exporting Graphics (Macintosh)

---

TrueType Name	PostScript Name
Times	Times
Times New Roman	Times

If you want to use fonts that are not in this table then you need to customize Igor's table as follows.

1. Open the experiment containing the graphic you wish to export and then create an EPS file. Igor will print in the history warnings about any fonts that are not in the table. The first time an EPS is generated from a given experiment, Igor creates a data folder containing a pair of text waves containing the names from the built-in table.
2. Using the Data Browser (Data Menu), navigate to "root:Packages:PSFontInfo:". You will see two waves named TTtoPS and TTPSFNames. Double click the icons to open the waves in a table. You will need to edit one or both of these tables.
3. Edit the TTtoPS wave to add a row at the bottom of the table that provides the screen font name on the left and the base PostScript font name on the right. If they are the same, you don't need to do this step.
4. Edit the TTPSFNames wave to add a row at the bottom of the table that provides the base PostScript font name in column 0, the normal font name in column 1, the bold name in column 2, the italic name in column 3 and the bold-italic name in column 4.

If you don't know the proper PostScript font names for the font you wish to use, you may be able to find this information in a file named 5090.fontnamelist.pdf on Adobe's web site.

This technique adds names only to the current experiment. If you want all experiments to have access to a set of names, you can create (or edit) a tab-delimited text file named UserFontNames.txt in "Igor Pro User Files/Igor Extensions" (see **Igor Pro User Files** on page II-46 for details). It must have the same structure as the TTPSFNames wave just described but should contain only the new fonts you are adding. Your screen font name should match the PostScript base name. If it doesn't, you may need to add an entry in the TTtoPS translation wave described above.

Igor gets the TrueType to PostScript translation table from the operating system if it is available. When you add a new font to your system, the installation program may update the system translation table. Because of this, the TTtoPS wave may contain more or different entries than described above. If you install a new font, you can force Igor to update the TTtoPS wave in a given experiment by deleting or renaming the wave and then writing out a dummy EPS file.

In addition to the UserFontNames.txt in the Igor Extensions folder, Igor also looks for a file named PSFontNames.txt. This file, if present, contains additional font names provided by WaveMetrics to extend Igor's built-in table. Although you can use this table rather than UserFontNames.txt you probably should not since your changes will be lost the next time a new version of Igor Pro is released.