CA-ScreenIO Users:

As you know, NORCOM (the original developer of ScreenIO) has agreed to "adopt" CA-ScreenIO users.

Under our agreement with Computer Associates, we will provide upgrades and support at no additional cost to you through the end of your maintenance contract with CA, whereupon we will bill you directly for maintenance and support.

Your upgrade to ScreenIO 2.4 is enclosed. ScreenIO 2.4 looks much like earlier DOS versions of ScreenIO, but it is completely Windows under the hood. You will find it very easy to convert your earlier ScreenIO applications to version 2.4.

Technical notes on ScreenIO 2.4:

Internet Support:

Updated versions, technical support, beta versions, and additional documentation on calling a few useful Windows APIs using ScreenIO facilities (as well as a fully-functional product for evaluation purposes) are available from our Internet site:

http://www.screenio.com

You can obtain timely technical support via email from:

Support@screenio.com

Which Panel Editor Should I Install?

If your development workstation is running a 32-bit version of Windows, use the 32-bit editor, which was compiled using CA-Realia's 32-bit COBOL compiler (version 6). If your development station is running Windows 3.1, you'll have to use the 16-bit editor, which was compiled using CA-Realia's 16-bit COBOL compiler (version 5). *Both panel editors produce identical panel copybooks.* The only difference is that they use different file systems.

The 32-bit panel editor will automatically convert a 16-bit panel library to a 32-bit panel library when you attempt to open it*. We don't modify the original library, we make a copy with the extension .PNL.* If a file with the .PNL extension exists, the converted library will be named with a .TMP extension.

You cannot convert a 32-bit library back to a 16-bit library, so if you make changes to a library using the 32-bit panel editor, you won't be able to use the DOS panel editor to access that library. This should not be a problem, however.

The 16-bit panel editor uses the Realia 16-bit file system to maintain the source library; it is backward compatible with the DOS panel editor.

Registration Key and Initial Setup:

The first time that you run the panel editor, it will ask you for a key. Enter the 13-character key printed on your product diskette. You only have to do this once. Write the key on your manual so that, if you have to reinstall ScreenIO, you'll know what it is. If you lose the key, just call us and we'll give you one.

The first time that you run the panel editor, it will automatically go to a Session Options panel. This replaces the DOS method of specifying the locations and names of libraries using environmental variables (which is not convenient in Windows). ScreenIO 2.4 saves your options in a .INI file located by default in the Windows subdirectory of your computer (though you may

specify another location if you wish.

The default .INI file is PEF.INI. You may use any name here; we suggest you use a different .INI file for each application that you maintain. If you do so, then you can create shortcuts on your desktop for each application and specify the .INI file as a command tail in the Target field of the shortcut's properties; e.g.,

<div align="center">C:\Program Files\ScreenIO\Pef32\Pef32.exe   Pef.ini</div>

Specify the name of the panel library in the Name field, the path in the Location field, the path to your mask library EDITMASK.SEQ and the subdirectory where you want ScreenIO to place the panel copybooks that it creates. Press Enter to save these in your .INI file. Note: The target subdirectories must exist; the panel editor will not create them for you!

If you're running the 32-bit panel editor, it will automatically convert your old 16-bit panel library to the 32-bit file system (relax, it won't touch the old library). Now you're ready to go.

Window Size and Font Selection:

Because ScreenIO 2.4 is, practically speaking, a character-mode application closely related to earlier versions of ScreenIO, it handles the size of its window from the "inside out" by basing it on the size of the font that you select. Use View/Fonts on the Window menu to experiment with this.

We have found that many of the fixed fonts supplied with Windows contain bugs in how they report the size of characters. Therefore, you may have to experiment a little to select the font that works best for you. Terminal and Fixedsys fonts always work, but Courier and Lucida may report the height and width of characters incorrectly, which causes ScreenIO to position the caret wrong, or causes the line spacing to be too tall for the characters (which results in vertical lines being broken at each screen row). In some cases, Regular style fonts work, but Bold fonts report the width incorrectly.

This is not much of a problem, just something that you should be aware of when you distribute your application. Generally speaking, if you specify the name of a .INI file to ScreenIO in your initialization call (causing ScreenIO to save and restore your settings between sessions of YOUR application), your users will only have to run through font selection once; it will be remembered automatically henceforth.

File Locations:

Our installation creates a subdirectory structure that will allow you to install all of the components without interfering with one another. You can't install everything to a single subdirectory because the names of some modules will conflict.

The shortcut we install will point at the panel editor executable, but you will have to configure your compiler to pick up the copybooks and libraries, and you will have to configure the panel editor to pick up your panel library and EDITMASK.SEQ file (and to direct it where to store your panel copybooks).

If you installed the product to the C:\Program Files\ScreenIO subdirectory, the files are placed in this structure:

C:\Program Files\ScreenIO\Copybook	Copybooks needed for compiles
C:\Program Files\ScreenIO\Copypan	Target location for panel copybooks
C:\Program Files\ScreenIO\Fujitsu	Fujitsu COBOL (32-bit)
C:\Program Files\ScreenIO\Mfocus40	Micro Focus Object COBOL support
C:\Program Files\ScreenIO\Pef32	32-bit Panel Editor executable
C:\Program Files\ScreenIO\Realia50	CA-Realia COBOL 5.0 support (16-bit)

C:\Program Files\ScreenIO\Realia60	CA-Realia COBOL 6.0 support (32-bit)
C:\Program Files\ScreenIO\Run32	32-bit runtime files for distribution
C:\Program Files\ScreenIO\Samples	Sample programs and panel library
C:\Program Files\ScreenIO\Source	Source code for stubs (other compilers)

Note: Even though we provide support for a number of compilers, all of our development is done using CA-Realia COBOL. In our opinion, it's technically the best PC COBOL out there.

Compiler Notes:

Each support subdirectory contains a LINKING.TXT file with specific instructions for using ScreenIO with that compiler, including linking and debugging information. Please review this information before calling us about technical support on these issues.

Place the copybook subdirectory and the panel copybook subdirectory in the path used by your compiler for copybooks. For CA-Realia, this is the SYSLIB environmental or the Build/Options/Compile/Copybook Path option in the CA-Realia Workbench. For Micro Focus COBOL, use the COBCPY environmental variable. See your compiler documentation, please.

Place the support subdirectory (and the ScreenIO.LIB object library contained therein) in the path used by your linker. Do this using the LIB environmental (batch compiles) or the Build/Options/Link/Library Files Path in the CA-Realia Workbench.

Mailing List:

If you send your email address to support@screenio.com we will place it in our distribution list for notices of interest to ScreenIO users. We promise that we won't drown you in junk mail or sell our mailing list to anyone.

Windows API Support:

It is sometimes handy to utilize Windows APIs for, say, maintaining a .INI file. We have exported a couple of Windows APIs from our runtime for this purpose. The available APIs are found in the SCR_UTIL.COB stub, located in the \Source subdirectory.

Documentation for these APIs is available at our Internet site.

Common Problems:

Please carefully read the compiler specific notes in the LINKING.TXT file in the appropriate support subdirectory for that compiler. If that doesn't help, see your compiler's documentation before calling us for technical support on compiler issues.

Most people have a bit of difficulty figuring out how to LINK their ScreenIO application properly. This has nothing to do with ScreenIO; each compiler has its own peculiarities, and some are just plain obtuse as to how you do certain things.

If you're still stuck, well, give us a call or email support@screenio.com and we'll help you out. We've always had a good reputation for technical support.

A little background on ScreenIO 2.4:

We implemented Windows features in ScreenIO 2.4 that made sense; scrollbars, the ability to easily select fonts, INI file support that remembers the look of your window between executions, and a few other things. But, we stayed with the old DOS-style OEM fonts for your convenience. You could characterize ScreenIO 2.4 as a "hybrid" application.

Why didn't we implement ScreenIO 2.4 as a GUI product?

Early on, we had to decide whether we were going to make your conversion to ScreenIO 2.4 an easy conversion (character mode) or a difficult one (GUI mode).

We chose easy.

As a result, all that you need to do is to insert a few lines of preamble code, recompile and link your application, and you're running as a Windows application. Your applications will run seamlessly under Windows. You get a big, flat address space, so you no longer require DOS memory extenders and their associated compatibility problems. You also won't encounter many hardware compatibility problems. All that stuff is handled by Windows, which is really nice.

If we had chosen to implement ScreenIO 2.4 as a fully GUI product, we could have preserved most of the programming interface, but you would have been forced to redesign your panels to make them look right with Windows ANSI fonts, plus you would have had to redefine all of your menus as Windows style structures. It would have been an extremely painful upgrade for programmers with large applications.

ScreenIO 2.4 represents a very large first step in our Windows product development efforts. It provides a simple and straightforward path to bring your legacy application into the Windows environment. We're not finished, however.

Will there be a GUI ScreenIO?

Well, yes, there will. We've been working for the last two years on a new, fully GUI "Son of ScreenIO" for COBOL that will be as easy to use as ScreenIO, but far richer in powerful features. You'll be able to easily support familiar Windows features; File/Open, File/Save, Print (and other) common dialogs, Windows cut and paste, scrollbars, buttons, graphical objects such as pictures, multiple child windows, and so on. We're pretty excited about it.

This will be a completely new product, but ScreenIO users will find it easy to convert using the conversion software we have developed. Programming is conceptually similar to ScreenIO.

We have not set a date for release yet, but test versions will be available from our internet site as soon as we feel it's ready for general testing; probably in a couple of months.

Comments/questions? Drop us a line at: support@screenio.com