Modifying Environment Variables

You must modify the environment variables of your system to be able to compile and run Elements Environment applications directly from the Neuron Data Elements program group.

You must modify the environment variables of your system to be able to compile and run Elements Environment applications directly from the Neuron Data Elements program group.

ON WINDOWS NT

Under Windows NT, environment variables can be set in either of the following ways.

1) Using Control Panel

Changes made to control panel are global and affect all programs. Note that an already opened DOS command window will not pick up the changes. You must close it and re-open a new one.

The steps involved are:

- Double click on the Control Panel icon, visible from the My Computer icon.
- Double click on the System icon within the Control Panel window.
- Type the environment variable name in the "Variable:" edit box and its value in the "Value:" edit box.
- Click the "Set" button and repeat the last two steps for any other new variables. Note that you can click on existing variables visible in the "User Environment Variables for ..." box to edit already existing variables. You can also remove existing variables by assigning no value to it.

Assuming that you have installed the Elements Environment under the `C:\<home_dir>` directory, the environment variables should be set as follows:

Variable	Value
`PATH`	for the Neuron Data debug program libraries, add `C:\<home_dir>\c\bin;C:\<home_dir>\c\dlldbg` or for the Neuron Data non-debug program libraries, add `C:\<home_dir>\c\bin;C:\<home_dir>\c\dll`
`INCLUDE`	add `C:\<home_dir>\c\include.`
`LIB`	add `C:\<home_dir>\c\lib.`

Also add the following lines to set Elements Environment-specific environment variables:

```
ND_PATH     C:\<home_dir>\c\dat;C:\<home_dir>\c\dat\enusasc
ND_HOME     C:\<home_dir>.
```

Note that for INCLUDE and LIB, you must append `%INCLUDE%` and `%LIB%` respectively to the value. This will make sure that the System Environment Variables are included along with your User Environment Variables.

Example: our value for LIB is `%LIB%;f:\<home_dir>\c\lib` where `%LIB%` is preset by the System to `d:\msvcnt\c\lib` (Visual C++ libraries).

The above prefix is not needed for the PATH variable since it picks up the System Environment Variables automatically.

2) Using a batch file

You can also run a batch file to change the path and environment variables from a Command Prompt window. The changes will then only affect the current Command Prompt window. A template batch file, setenv.bat, is included in the misc directory to set the environments for compiling and running Elements Environment applications.

SPECIAL ENVIRONMENT VARIABLES

There are several environment variables available to customize your Elements Environment applications. A typical default configuration should not require the usage of any special environment variables. You may want to review the list, however, to familiarize yourself with the options available.

ND_CACHECLIPS

When this environment variable is set to TRUE, Open Interface delays the actual low-level clipping operations until a drawing call is made, and discards all clipping operations for which no drawing call is made. This allows for a speed increase in drawing, especially under Microsoft Windows and Windows NT. By default, ND_CACHECLIPS is set to TRUE.

ND_COLOROPT

If set to MONOCHROME (on all platforms except Mac and Windows) everything is in monochrome.

ND_DEBUGFILE

Set to a file name (string) to specify an output file for debug messages. If no name is supplied (NULL), then no debug file is created and error or warning messages are not written to file.

ND_DRAWOPT

This environment variable can take the following values under Windows:

DITHERNONE	No dithering. All colors are solid.
DITHERALL	Dithered colors are used to fill shapes and draw lines.
DITHERRGBONLY	Dithered colors are used except to draw lines which will be solid. (Default value.)

ND_ENFORCE64KLIMIT

This environment variable is provided to test compatibility with 16 bit Microsoft Windows applications for which allocation of buffers larger than 64K require using special “Huge” buffers and API calls. 16 bit Windows applications use Windows or OS/2 libraries linked with the Microsoft compiler.

When this flag is set to TRUE, buffers larger than 64K cannot be allocated with the call PTR_New and must use the call PTR_HugeNew instead. An assertion failure will be generated if the wrong call is used. By default ND_ENFORCE64KLIMIT is set to TRUE when running 16 bit Windows applications and set to FALSE for all other environments.

This environment variable can be set to TRUE on a non-Windows platform to ease portability to Windows.

ND_ICONDITHERING

Controls the rendering of color images. 4 values:

- if set to NONE, it tries to allocate the exact color

- if set to "1x1", it tries to map to the closest color in the colors pre-selection

- if set to "2x2", it tries to map to the closest color in the colors pre-selection with dithering on colors following a 2by2 Bayer matrix

- if set to "4x4", it tries to map to the closest color in the colors pre-selection with dithering on colors following a 4by4 Bayer matrix

ND_LOGDPI and ND_SCALEFACTORS

A screen is characterized by two resolutions:

- its physical resolution in dpi (dots per inch)
- its logical resolution in logical dpi (dots per logical inch)

The physical resolution is the number of pixels per inch on the screen. It is a characteristic of the screen hardware.

The logical resolution is the resolution for which the fonts have been designed and may be quite different from the physical resolution. The logical resolution is actually a characteristic of the screen driver, not a characteristic of the screen hardware itself.

Open Interface can perform scaling of window and widget coordinates to accomodate different screen resolutions.

The scaling is controlled by the ND_SCALEFACTORS environment variable:

If ND_SCALEFACTORS is not defined, Open Interface will try to preserve the "pixel" sizes of widgets and windows and will interpret the font sizes as "point" sizes for the logical resolution of the screen (for fonts defined through the new "Font Family Resource" mechanism).

If ND_SCALEFACTORS is defined, it should be defined as a list of two scaling factors expressed in percent and separated by a semi colon (i.e. 100;100 or 120;90).

The first scaling factor defines how windows and widgets will be scaled. If it is set to 100, Open Interface will try to preserve the actual sizes (in inches or millimeters) when displaying on different screens. If it is set to 120, the windows and widgets will be 20% larger than in the environment in which they were originally designed. If set to 0, Open Interface will preserve the "pixel" size of the windows instead of preserving their "actual" size.

The second scaling factor defines how fonts (and thus text) will be scaled. If it is set to 120, the fonts will be 20% larger than they would normally be, given the natural logical resolution of the screen.

To set up ND_SCALEFACTORS, you should design some sample windows on a "reference" screen, then port it to the other screens and tune ND_SCALEFACTORS to get acceptable results on all screens.

If you plan to run the Elements Environment on various screens with different resolutions, you should either define ND_SCALEFACTORS on all screens or on none of them. If you choose to define ND_SCALEFACTORS, you should configure ND_SCALEFACTORS on all your screens from a "reference" screen (as described above) first. Once the ND_SCALEFACTORS have been frozen, you can start editing resources on the various screens but you might run into troubled situations if you start modifying ND_SCALEFACTORS on one of the platform when you already have a set of resources designed on different screens.

The ND_SCALEFACTORS mechanism superseeds the OIT_SCALE/OIT_DPIS mechanism which was provided in prior versions. We hope that this new mechanism is much easier to understand than the previous mechanism and we strongly encourage you to use it rather than the OIT_SCALE/OIT_DPIS mechanism.

On X11, the windowing system does not define any "logical resolution" and usually provides several sets of fonts designed for several logical resolutions. By default, we set the logical resolution to 75 or 100 dpi, whichever is the closest to the actual screen resolution. This "guess" might not give the a very satisfactory value. So, you can set the logical resolution either by defining the ND_LOGDPI environment variable as the logical resolution in dpis or by issuing the following calls before the initialization of Open Interface.

ND_LOOK

You can control the look and feel of your application by using the environment variable ND_LOOK. The possible values of ND_LOOK are:

WIN95	Microsoft Windows 95
WIN31	Microsoft Windows. (Default value.)
PM10	Presentation Manager, version 1.0.
PM20	Presentation Manager, version 2.0.
OPENLOOK	Sun Open Look.
MOTIF	OSF Motif.
MAC	Macintosh

ND_MODALOPT

Set to SYSTEMMODAL (on X Windows-based platforms only) to force modal windows to be above all windows.

ND_SHOWALLRES

ND_SHOWALLRES is by default set to FALSE on all platforms. The Resed resources (resources defined by the Elements Environment) are hidden from Elements Environment's "List of <res class>" dialogs when picking a Color, an Icon, etc. for a widget but they are still displayed in the resource browser. Also hidden are the resources named "<class>.Def<...><look>" where <look> can be Motif, OpenLook, etc. This is done to avoid the problem of accidentally including resed resources or the "<class>.Def<...><look>" resources in your application, especially icons.

Since Resed resources are not available with the runtime version, this scheme will prevent users from inadvertently using these resources, porting to a runtime version, and discovering the absence of the resources when attempting to rescomp. Furthermore, these resources might be deleted or changed in a future version of the Elements Environment. You should only include resources from the lower level libraries or create new resources from the existing resed or "<class>.Def<...><look>" resources. If you want to see these resed or "<class>.Def<...><look>" resources in the "List of <res class>" dialogs of the Elements Environment set ND_SHOWALLRES to TRUE.

ND_USEIMPLICITCLIP

ND_USEIMPLICITCLIP is by default set to TRUE on all platforms. When it is set to TRUE, OPEN INTERFACE checks each drawing call to verify that a clipping has been done for that call. This actually restricts any drawing inside a widget to the visible region of a widget. If ND_USEIMPLICITCLIP is set to FALSE, it does not set an implicit clipping to the corresponding widget's visible part.

LANGUAGE ENVIRONMENT SETUP

Open Interface Element supports several new languages and lets you easily change char-

acter sets to accommodate a specific language. To enable languages other than US English, do the following:

1. Select the codeset for processing keys through the ND_CHARNATIVE environment variable.
2. Select the language resources through the ND_LANG environment variable.

The following paragraphs describe the selections you can use with the above procedure, for specific languages.

Input Method Activation

You do not need to do anything. The input method is automatically activated if you are using a localized Win32 platform.

Codeset Selection

Set the ND_CHARNATIVE variable to one of the following values:

For now only US English and Japanese are available:

US English:	CT_ASCII (or set to nothing)
Japanese:	CT_SJIS

For example:

```
% setenv ND_CHARNATIVE CT_SJIS
```

Language Resource Selection

Set the ND_LANG variable to one of the following values:

US English and Japanese EUC resources are available:

US English:	enusasc
Japanese Shift-JIS:	jajpsjis

For example:

```
% setenv ND_LANG jajpsjis
```