NEURON DATA

INSTALL NOTES

Elements Environments
Solaris / SPARC Users

RUNNING UNDER OPENWINDOWS WINDOWING SYSTEM

Starting An X Application

You must be running under the OpenWindows Windowing System to use the Elements Environment or your Elements Environment-based application. If you are not familiar with the OpenWindows Windowing System, see your system administrator.

You can start the Elements Environment executable by typing:

```
% ee
```

(Your Unix PATH environment variable should include the bin subdirectory of the Elements Environment home directory. If it does not, you will have to go to the `bin` subdirectory to start EE, or use its complete path: `$ND_HOME/bin/ee`, or a relative path, such as `../bin/ee`.)

If you get the message:

```
Cannot open display
```

it means that Elements Environment or your Elements Environment application cannot make a connection to the OpenWindows Windowing System. Perhaps the X/terminal server process has not been started, or your machine is not authorized to the server, or perhaps the network is down. Other explanations are that you did not start the application from a bitmapped terminal, or have not correctly set the environment variable DISPLAY.

The DISPLAY environment variable should be set to (in order of preference):

```
`hostname`:0
unix:0
<internet address>:0
```

For example:

```
% setenv DISPLAY `hostname`:0                C shell
```

or

```
$ DISPLAY=`hostname`:0; export DISPLAY       Bourne shell
```

If you are still having difficulties, see your system administrator.

Reading the .Xdefaults File

In order for the Elements Environment to read the resources of a **.Xdefaults** file not located in your home directory, you will need to set the X11 environment variable, XENVIRONMENT, to the location of the **.Xdefaults** file.

For example, if the **.Xdefaults** file that you want the Elements Environment to read is located in the `/usr2/users` directory, you would set the XENVIRONMENT to the following:

```
% setenv XENVIRONMENT /usr2/users/.Xdefaults   C shell
```

or

```
$ XENVIRONMENT=/usr2/users/.Xdefaults          Bourne shell
$ export XENVIRONMENT
```

Reading the .Xdefaults File Remotely

You will also need to set the XENVIRONMENT environment variable if you are running the Elements Environment or your application on a remote machine, but are displaying the application on an X windows display on your local machine, and you want the Elements Environment to read the resources of the **.Xdefaults** file from the local machine (i.e., the machine which the Xwindow server is running.) For example, lets say the local (X server) machine is called “dolphin” and the remote machine (on which the application is running) is called “shark”.

If you start an Elements Environment application on the “shark” machine but tell the application to appear on the Xwindows server running on the “dolphin” machine, the Elements Environment application will read the resources from the **.Xdefaults** file from the remote machine, in this case, the machine “shark”. If you want to override this behavior (for example, you wish that the Elements Environment application read the resources from the local machine, “dolphin”, on which the Xwindow server is running), you must mount the “dolphin” file system on to the system “shark”, and set the XENVIRONMENT environment variable to point to the correct location before starting the Elements Environment application.

```
% mount dolphin:/usr /dolphin                  C shell
% setenv XENVIRONMENT /dolphin/users/me/.Xdefaults
```

or

```
$ XENVIRONMENT=/dolphin/users/me/.Xdefaults    Bourne shell
$ export XENVIRONMENT
```

Reading in X Motif Resources

You can specify MOTIF Elements Environment resources for all users on a system by creating an Elements Environment file in the `/usr/lib/X11/app-defaults` directory or by adding Elements Environment resources into the **/usr/lib/X11/Mwm** file. You can also specify these resources locally by creating or adding resources to your local **.mwmrc** or **.Xdefaults** files in your home directory. You can also specify MOTIF Elements Environment resources for all users on a system by adding Elements Environment resources to the **/usr/lib/X11/sys.Xdefaults** file. See Chapter 10 of the Xwindow System User’s Guide for more detail information on resource definition and setting resources. For the MOTIF window manager, Elements Environment recognizes the following resources:

Background - This resource specifies the background color of the Elements

Environment windows. The following example shows how this resource can be set only for Elements Environment applications:

```
Mwm.OpenInterface*background:   blue
```

The following example shows this resource can be set for all MOTIF application including Elements Environment applications:

```
Mwm*background:                 blue
```

Foreground - This resource specifies the foreground color of the Elements Environment windows.

ActiveBackground - This resource specifies the background color of the Elements Environment window's mwm decoration when the window is active (has the keyboard focus.)

ActiveForeground - This resource specifies the foreground color of the of the Elements Environment window's **mwm** decoration when the window is active (has the keyboard focus.)

FontList - This resource specifies the font used in the window manager decoration.

Menu.Background - This resource specifies the background color of the Elements Environment menus and menu bars.

Menu.Foreground - This resource specifies the foreground color of the Elements Environment menus and menu bars.

ResizeBorderWidth - This resource specifies the width (in pixels) of a Elements Environment's client window frame border with resize handles. The specified border width includes the 3-D shadows.

KeyboardFocusPolicy - This resource defines the keyboard focus policy.

explicit - If the KeyboardFocusPolicy is set to "explicit", you must click on the window with the left mouse button to change the input focus to the current window.

pointer - If the KeyboardFocusPolicy is set to "pointer", you simply have to move the pointer over the window to change the input focus to the current window.

The keyboard focus policy can be set for not just Elements Environment applications but for all MOTIF applications:

```
Mwm*KeyboardFocusPolicy:        pointer
```

Reading in Open Windows Resources

You can either specify Open Windows Elements Environment resources for all users on a system by creating an OpenWindows file in the `/usr/openwin/lib/app-defaults` directory, or you can also specify these resources locally by creating or adding resources to your local **.Xdefaults** file in your home directory. If you have an **.OWdefaults** file in your home directory, you may also specify resources there; resource definitions in the **.OWdefaults** file will override definitions in the **.Xdefaults** file. For the OPEN WINDOWS window manager Open Interface Elements recognizes the following resources:

SelectDisplaysMenu - This resource determines if the SELECT button is clicked on a button menu whether or not it will automatically execute the default choice on the menu (the option shown with the parentheses around it.)

False - If the SelectDisplaysMenu resource is set to "False" when you click on a button menu with the SELECT button, the default choice on the menu automatically gets executed (the option shown with the parentheses around it.)

True - If the SelectDisplaysMenu resource is set to "True" when you click on a button menu with the SELECT button, the default choice on the menu does not get executed (the option shown with the parentheses around it.)

If this resource is not set, it defaults to the "True" option

SetInput - This resource defines the keyboard focus policy.

select - If the SetInput policy is set to "select", you must click on the window title bar to change the input focus to the current window.

followmouse - If the SetInput policy is set to "followmouse", you simply have move the pointer into the window to change the input focus to the current window.

The key board focus policy can be set for not just the Elements Environment applications but for all Open Windows applications:

```
OpenWindows*SetInput:       select
```

WindowColor - This resource specifies the background color of the window.

Matte Decoration Element

For the Elements Environment to work correctly with the MOTIF Window Manager, the decoration element called matte must be set to 0. This is how it should appear in the **.Xdefaults** file located in user's home directory:

```
Mwm*OpenInterface.matteWidth: 0
```

On most platforms this is the default, so unless this element is explicitly set to something besides 0, you will not have to set this element.

IMPORTANT ENVIRONMENT VARIABLES

The following environment variables are available for you to customize the appearance or functionality of Elements Environment applications on Solaris Sun/Sparc.

ND_X11RELSE

If you are running Open Interface Element on an X Windows Server that is based on MIT Release3 (X11R3) or less of the X Windows System and you are using the MOTIF Window manager then you should set the ND_X11RELSE environment variable to PREX11R4 for Solaris Sun4/Sparc:

```
22% setenv ND_X11RELSE  PREX11R4       C shell
    or
$ ND_X11RELSE=PREX11R4                 Bourne shell
$ export ND_X11RELSE
```

ND_WINMGR

Open Interface Element tries to determine which window manager is running, so in most cases, you will not need to set the ND_WINMGR variable. Open Interface Element informs you through error messages if it cannot determine which window manager is running or if the ND_WINMGR environment variable is not set correctly. If Open Interface Element cannot determine what window manager is running, then ND_WINMGR must be explicitly set. Here are the possible error messages that you can receive from Open Interface Element regarding the window manager:

Error 1

```
Cannot determine which window manager is running (none found), please set the
ND_WINMGR variable (see installation guide for details.)
```

In this case, Open Interface Element cannot determine which window manager is running. Either there is no window manager running and you must start the window manager or the window manager running is not MOTIF or OPENLOOK. If the window manager running is not MOTIF or OPENLOOK, then the ND_WINMGR environment variable should be set to DECWINDOWS or OTHER depending on your actual window manager. (See below for directions on setting the ND_WINMGR environment variable.)

Error 2:

```
Cannot determine which window manager is running (ambiguity), please set the
ND_WINMGR variable (see installation guide for details.)
```

In this case, Open Interface Element cannot determine what window manager is running. Some of the characteristic atoms of the window manager are reporting the window manager is of one type and some characteristic atoms are reporting that the window manager is of another type. Open Interface Element cannot resolve this ambiguity, so you must explicitly set ND_WINMGR to the appropriate value. (See the directions below for setting the ND_WINMGR environment variable.)

Error 3

```
The ND_WINMGR environment variable is currently defined as "???" and that window
manager is not running on your display
```

In this case, Open Interface Element has determined that the window manager is of one type and the ND_WINMGR environment variable reports that the window manager is of a different type. In this case, you should correctly set the ND_WINMGR environmental variable to the correct value. (See the directions below for setting the ND_WINMGR environment variable.) If ND_WINMGR is set to OTHER, Open Interface Element doesn't execute this check.

The environment variable, ND_WINMGR can be set to four possible values, MOTIF, OPENLOOK, DECWINDOWS, or OTHER. If you are displaying Open Interface Element on the local X Windows server then the window manager running is probably OPENLOOK.

If you are prompted by Open Editor to set the ND_WINMGR environment variable, this is how it should be defined for the OPENLOOK window manager:

```
22% setenv ND_WINMGR OPENLOOK                C shell
    or
$ ND_WINMGR=OPENLOOK; export ND_WINMGR       Bourne shell
```

ND_CACHECLIPS

By default, if ND_CACHECLIPS is not set, ND_CACHECLIPS is TRUE but is only used under Microsoft Windows and the X11 Windowing System. When this environment variable is set to TRUE, Open Interface applications delay the actual low-level clipping operation until a drawing call is made, and discard all clipping operations for which no drawing call is made. This allows for a speed increase, especially under Microsoft Windows.

```
22% setenv ND_CACHECLIPS FALSE               C shell
    or
$ ND_CACHECLIPS=FALSE                        Bourne shell
$ export ND_CACHECLIPS
```

ND_COLOROPT

If you want to test how applications will look on a monochrome display, you can set the ND_COLOROPT environment variable to MONOCHROME.

```
22% setenv ND_COLOROPT MONOCHROME            C shell
    or
$ ND_COLOROPT=MONOCHROME                     Bourne shell
$ export ND_COLOROPT
```

ND_DELETEISBACKSPACE

ND_DELETEISBACKSPACE does not need to be set on most platforms. It is automatically enabled on RISC Ultrix and is automatically disabled on non-RISC Ultrix UNIX platforms. ND_DELETEISBACKSPACE only needs to be set if you are running Open Editor or a Open Interface Element application on a RISC Ultrix machine but are displaying it over the network to an Xwindow Server running on a non-RISC Ultrix machine, or if you are running Open Editor or a Open Interface Element application on a non-RISC Ultrix machine but are displaying it over the network to an Xwindow Server running on a RISC Ultrix machine. It does not need to be set when running on a RISC Ultrix machine and displaying on RISC Ultrix machine, or running on non-RISC Ultrix machine and displaying on non-RISC Ultrix machine.

This environment variable is necessary because many platforms have two similar keys: backspace and delete. Backspace is mapped to a destructive backward delete; delete is mapped to a destructive forward delete. The "standard" DEC keyboard (LK201) has an unlabeled key that is used as "backspace", but appears (under X) to send "delete". So the default on the DEC keyboard is that there is only one erase-type key, but it deletes "forward". This environment variable tells Open Interface Element to do the expected thing with this key (the user expects it to function like "backspace").

ND_DELETEISBACKSPACE needs to be set to FALSE if you are running Open Editor or a Open Interface Element application on a RISC Ultrix machine but are displaying it over the networking to a Xwindow Server running on a non RISC Ultrix machine.

```
22% setenv ND_DELETEISBACKSPACE FALSE         C shell
    or
$ ND_DELETEISBACKSPACE=FALSE                  Bourne shell
$ export ND_DELETEISBACKSPACE
```

ND_DELETEISBACKSPACE needs to be set to TRUE if you are running Open Editor or a Open Interface Element application on a non RISC Ultrix machine but are displaying it over the networking to a Xwindow Server running on a RISC Ultrix machine.

```
22% setenv ND_DELETEISBACKSPACE TRUE          C shell
    or
$ ND_DELETEISBACKSPACE=TRUE                   Bourne shell
$ export ND_DELETEISBACKSPACE
```

ND_DRAWOPT

If your X Window's server is slow at drawing dashed lines (many X servers are very slow on such operations), you should define the following option:

```
22% setenv ND_DRAWOPT SOLIDLINES              C shell
    or
$ ND_DRAWOPT=SOLIDLINES                       Bourne shell
$ export ND_DRAWOPT
```

ND_ENFORCE64KLIMIT

By default if ND_ENFORCE64KLIMIT is not set on non-PC platforms it is assumed to be FALSE. It is set to FALSE because on non-PC platforms memory is not limited to 64K segments. For a PC system regardless of what this environment variable is set to, ND_ENFORCE64KLIMIT will set to TRUE because a PC system is limited to 64K memory segments. This environment variable can be set to TRUE on a non-PC platform for portability testing with a PC system.

ND_FORCEMOD1TOALT

If you are running on a Solaris Sun4/Sparc machine, by default ND_FORCEMOD1TOALT will be set to TRUE. If ND_FORCEMOD1TOALT is set to TRUE, then the key corresponding to MOD1 key on the keyboard maps into the ALT key. This was the default behavior with OPEN INTERFACE Version 2.0.

If ND_FORCEMOD1TOALT is set to FALSE, then the key corresponding to MOD1 does not the map into the ALT key.

ND_FORCEWINMGR

Open Interface Element will not determine which window manager is running. Open Interface Element will try to interact with the named window manager. Acceptable values are: DECWINDOWS, OPENLOOK, MOTIF, TWM, MOLWM, NCDWM, UWM, VISIONWAREWM, OTHER, NONE. For example:

```
22% setenv ND_FORCEWINMGR  TWM                C shell
    or
$ ND_FORCEWINMGR=TWM                          Bourne shell
$ export ND_FORCEWINMGR
```

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 ND_SCALE/ND_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 ND_SCALE/ND_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.

PM10	Presentation Manager, version 1.0.
PM20	Presentation Manager, version 2.0.
OPENLOOK	Sun Open Look. (default).
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.

```
22% setenv ND_MODALOPT SYSTEMMODAL           C shell
    or
$ ND_MODALOPT=SYSTEMMODAL                    Bourne shell
$ export ND_MODALOPT
```

ND_SHOWALLRES

ND_SHOWALLRES is by default set to FALSE on all platforms. The resource editor (Open Editor) resources are hidden in the editor's "List of <res class>" dialogs, when picking a Color, an Icon, etc. for a widget but they are still displayed in the 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 resource editor resources or the "<class>.Def<...><look>" resources in your application, especially icons, because these resources might be deleted or changed in a future version of Open Editor. You should only include resources from the lower level libraries or create new resources from the existing resource editor or "<class>.Def<...><look>" resources. If you want see these resource editor or "<class>.Def<...><look>" resources in the "List of <res class>" dialogs of Open Editor set ND_SHOWALLRES to TRUE.

```
22% setenv ND_SHOWALLRES TRUE                C shell
    or
$ ND_SHOWALLRES=TRUE                         Bourne shell
$ export ND_SHOWALLRES
```

ND_SUNKBTYPE

ND_SUNKBTYPE is used to set the keyboard type by any platform which accesses a Sun server. Recognized values are TYPE4, TYPE5 and OTHER. This variable will be set by the Elements Environment itself as it initializes, so you should probably never have need to change this value. The type of keyboard you are using is usually printed on the underside of the keyboard. You can set this variable for a type 5 keyboard (for example) by entering:

```
22% setenv ND_SUNKBTYPE TYPE5                C shell
    or
$ ND_SUNKBTYPE=TYPE5                         Bourne shell
$ export ND_SUNKBTYPE
```

ND_RELYONKEYRELEASE

On UNIX platforms by default ND_RELYONKEYRELEASE is set to TRUE. On all the X11 Windowing systems we tested it was found that you can rely on the fact that key release events were sent even if the X11 documentation says that you can't rely on this behavior. So in this case, the server remains grabbed while the Alt key is pressed no matter what happens in between. If you are working on a server in which you really can not rely on the key release events to be sent, set this environment variable to FALSE.

ND_TRANSLATEALTWITHMODE

The ND_TRANSLATEALTWITHMODE environment variable is only used by X11 Windowing System. By default this environment variable is set to FALSE if the X window server is X11R4 or greater. When set to FALSE, Open Interface Element does not let XLookupString translate a key event using the second keyboard group if the keyboard group modifier is the same as the Alt modifier. This is the case with HPUX systems 8.0 or higher. If you want an Open Interface application to let XLookupString use the second keyboard group no matter what the keyboard group modifier is, you'll need to set this environment variable to TRUE.

ND_USEIMPLICITCLIP

ND_USEIMPLICITCLIP is by default set to TRUE on all platforms. When it is set, an Open Interface application checks each drawing call to verify that a clipping has been done for that call. If ND_USEIMPLICITCLIP to FALSE, it does not set an implicit clipping to the corresponding widget's visible part.

```
22% setenv ND_USEIMPLICITCLIP FALSE            C shell
    or
$ ND_USEIMPLICITCLIP=FALSE                     Bourne shell
$ export ND_USEIMPLICITCLIP
```

OPENLOOK ENVIRONMENT VARIABLES

The following environment variables are available for you to customize the appearance or functionality of Open Interface applications which are running on machines that use the OpenLook window manager from SunMicrosystems.

ND_SYNCMODE

In order to present a portable API, Open Interface needs to synchronize with the window manager on some window management operations (changing window states, etc...). This environmen variable can be set to turn off this synchronization.

```
22% setenv ND_SYNCMODE FALSE                   C shell
    or
$ ND_SYNCMODE=FALSE                            Bourne shell
$ export ND_SYNCMODE
```

ND_XOVERRIDESYNCONOPEN

The OpenLook window manager behaves unpredictably while a mouse button is pressed and the X server is doing an implicit grab operation. This can cause long delays before windows are displayed from a mouse button press event. This variable can be set to turn off synchronization on window opening operations:

```
22% setenv ND_XOVERRIDEONOPEN  FALSE          C shell
    or
$ ND_XOVERRIDEONOPEN=FALSE                    Bourne shell
$ export ND_XOVERRIDEONOPEN
```

ND_XINITIALSYNCDELAY

Defines the synchronization delay for a opening a window (default is 10000 miliseconds). The wait time for opening the window can be reduced by setting this variable:

```
22% setenv ND_XINITIALSYNCDELAY  10000        C shell
    or
$ ND_XINITIALSYNCDELAY=10000                  Bourne shell
$ export ND_XINITIALSYNCDELAY
```

ND_XSYNCDELAY

Defines the synchronization delay for a all window operations, except opening(default is 300 miliseconds):

```
22% setenv ND_XSYNCDELAY300                   C shell
    or
$ ND_XSYNCDELAY=300                           Bourne shell
$ export ND_XSYNCDELAY
```

ADDITIONAL UNIX SETTINGS

ND_PATH defines a list of directories that the Elements Environment will use to open resource and data files. If you do not define it, the Elements Environment will not be able to find the resource databases (*.dat files) that are needed in order to run.

ND_LIB is the environment variable that defines the directory (under the Elements Environment home directory) containing the library files with which you will link an application. You should set it to **lib** if you want to link with the non-debug libraries or **libdbg** if you want to link with the debug libraries.

Also, look in the **$ND_HOME/mkinc/c/mkdefs.inc** file for more information on setting the optional environment variables, ND_CFLAGS, ND_LDFLAGS, and ND_RCFLAGS.

LANGUAGE ENVIRONMENT SETUP

Open Interface Element supports several new languages and lets you easily change character sets to accommodate a specific language. To enable languages other than US English, do the following:

1. Activate the input method for the language as described below.
2. Select the codeset for processing keys through the ND_CHARNATIVE environment variable.
3. 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

In this version, only XIM-based input methods are available. The Japanese Canna input method is not available. In order to enable the canna-input method, you need to do the following:

1. Run `$ND_HOME/canna/canna_install` as a superuser for canna installation.
2. Edit the `$ND_HOME/dat/ee.cfg` configuration file to load the canna dynamic library instead of the default XIM-based input method. Modify the file as follows:

   change: `ND_IM_NATIVE on` **to** `ND_IM_NATIVE off`
   add: `ND_IM_JAPANESE on`
3. Run `$ND_HOME/canna/bin/cannaserver` as a superuser.
4. Select the hostname of the canna server through the `OIT_CANNAHOST` environment variable.
5. Run the Elements Environment to use the new input method.

The above example describes how to use the canna-input method in the Elements Environment. To use canna in your application, add the following two lines to your xxx.cfg file for your application’s `.cfg` configuration file:

```
ND_IM_NATIVE off
ND_IM_JAPANESE on
```

To use the canna library, the LD_LIBRARY_PATH needs to include the `/usr/lib/canna` directory. This is automatically done during the Elements Environment installation.

The ND_IM_ENABLED environment variable is by default set to TRUE in order to enable the XIM-input method. You do not need to change this environment variable when using the canna-input method.

For example:

```
% setenv ND_IM_ENABLED TRUE
```

Codeset Selection

Set the ND_CHARNATIVE variable to one of the following values:

US English:	CT_ASCII (or set to nothing)
Japanese:	CT_JEUC
Korean:	CT_KSC
Simplified Chinese:	CT_GB
Traditional Chinese:	CT_BIG5

For example:

```
% setenv ND_CHARNATIVE CT_JEUC
```

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 (default value)
Japanese EUC:	jajpeuc

For example:

```
    % setenv ND_LANG jajpeuc
```