Contents Index

B

Altering the Menu

ENlighten/DSM has menus and icons that can be altered to suit your needs, including:

- The order of menu icons
- The order of menu items
- The actions of menu items
- The number icons and menu items

☞ You may *not* remove the CONFIG menu from the main menu list. This will cause improper placement of the remaining menu icons in the main menu bar.

The ENlighten/DSM GUI is built through a small text database (file) containing objects and attribute lists. You can create alternative interfaces by editing this file. For example, you may wish to add a menu item that runs your own script as its action. Or you can replace the archive configuration menu item and substitute your own backup tool.

☞ During the ENlighten/DSM start-up, the interface file will be scanned for both syntax and for attribute matching. If an error is detected, ENlighten/DSM reports on `stderr` the line number and the reason for its rejection, and terminates the start-up process.

Overriding the Default Configuration File

The ENlighten/DSM GUI is defined in a configuration file. This file contains information describing the layout of the windows. You can edit the configuration file to make desired additions or modifications to the GUI. Then you can override the default configuration file `$ENLIGHTEN/config/winn.out` by setting the shell variable `WINOUT_NET` to the name of YOUR configuration file.

If ENlighten finds the shell variable `WINOUT_NET` set and it contains the name of a readable file, this file will be used instead of the default configuration file. If the shell variable is set, but points to a file that does not exist or is unreadable, a warning will be issued. ENlighten will then use the default configuration file.

ENlighten is shipped with a backup copy of the configuration file called `$ENLIGHTEN/config/winn.org`. You can copy this over its corresponding `.out` file if you need to return the interface to its original state.

Warning! The format of the interface file is crucial to the successful start-up of ENlighten and the consistency of the product. *Be careful when making any modifications to this file*. You should also test your modified interface before replacing the current version.

Where to Edit

While there is no order of declaration of menus within the interface file, the current group of Menu definitions occurs after the set of MODEL initializations. If you wish to alter the existing menu definitions, or add your own, you should do so at this location (between the MODELs and the Window definitions). You can quickly locate the first menu definition by searching for the keyword Define.

The only required order in the file is that menus must be defined prior to any reference to their use. Thus, a Menu must be defined before it is attached to an icon in a window, and a submenu must be defined before it is attached to a menu item.

☞ A submenu is not a unique object; it is just the term for a Menu that's attached to a menu item as a pull-right style menu.

Defining Menus

The interface file uses a simple, yet concrete, syntax. Any line with a # as its first character is a comment. Any line with no characters (zero-length) is a break line. Otherwise, every line has three fields separated by tabs.

To create a menu, use the syntax:

DEFINE`<tab>`**12**`<tab><menu-name>`

where `<menu-name>` is a single-word name that is unique among the defined menus and used internally for reference purposes only. For example, to add a "UserMenu1" menu, add the definition:

```
DEFINE     12     USERMENU1
```

Subsequent lines following the DEFINE statement set the menu attributes and menu items (and their attributes) for the current DEFINE menu. This occurs until the next DEFINE statement is reached.

Menu Attributes

The syntax for an attribute is:

```
<tab><attribute-type><tab><value>
```

The valid attributes for menus are (order is not important):

- **categ** `<number>` = general ENlighten sphere of monitoring
- **context** `<string>` = mini-help text that appears when the mouse moves over the menu icon
- **pinned** `<value>` = whether window is pinnable
- **label** `<string>` = menu title appearing at the top of the menu window

categ `<number>`

Each category is in charge of windows and actions associated with it. The default interface shows the full range of windows accessible from a given category; it is not possible to change a window's accessibility from one menu category to another. Thus, it is not possible to call the TTY Enabling window from the Printer category. This restriction occurs since the window generation is determined by the Menu Category and the Return Code (as generated by menu items).

The valid (reserved) menu categories for ENlighten/DSM are:

2 =	Configuration
3 =	System
4 =	Disk
5 =	User
6 =	RESERVED
7 =	Printer
8 =	Archive
9 =	Events
10 =	Security
11 =	User-Defined Section 1
12 =	User-Defined Section 2
15 =	Network
17 =	Status Map
18 =	Close/Exit

The valid (reserved) submenu categories for ENlighten are:

51 =	Spool
53 =	Activities
55 =	Statistics
57 =	Logins
58 =	Hard disks
59 =	cron
60 =	NFS
61 =	DNS
62 =	Aliases
63 =	Clocks

context `<string>`

You may give a string, enclosed in quotation marks, that ENlighten uses to display at the bottom of the Main Menu Window as the mouse is dragged into the menu's associated icon. This is a form of context-sensitive help. If the text is longer than the window, it is "lopped off" at the window edge.

pinned `<value>`

This attribute determines whether a menu is pinnable; it will remain visible by moving the mouse over the window push-pin, so its menu items are always accessible without going back to the menu icon. The value is either 0 or 1 (False or True). By default, a menu is not pinned, so you only need to specify this attribute if the menu will be pinnable.

label `<string>`

Whenever a menu is displayed, the title of the menu appears at the top. A title is required for all menus and it must have quotation marks around it.

Sample UserMenu1 Menu

For our UserMenu1 menu example, we might set the following attributes:

```
DEFINE       12            USERMENU1
             categ         11
             pinned        0
             label         "User Scripts 1"
             context       ""Help subsystem"
```

This example defines a Menu we can refer to later in the interface file as `USERMENU1`, within the User-Defined section, not pinned (so it will disappear after a selection is made), with the title "User Scripts 1", and some help text showing that this menu comprises menu items that call up internally created executables.

Defining Menu Items

Menus also contain Menu Items, which comprise a list of entries from which the user may select. A menu item may be a pull-right item, meaning that a small arrow appears to the right of the item label. If the user “follows” the arrow, a submenu appears.

A menu item is simply an attribute assigned to a menu. It is created using the attribute type ‘add’. Thereafter, any further attributes listed apply not to the menu itself, but to the menu item most recently added. The syntax for adding a menu item is:

```
<tab>add<tab><entry text>
```

where `<entry text>` is the actual text (enclosed in quotation marks) the user will see in the list of menu items. Items will appear in the menu in the order specified within the file, with the first item appearing at the top of the menu.

Menu Item Attributes

Attributes for menu items use the same syntax as those for Menus (see “Menu Attributes” on page B-5). The valid attributes for a menu item are (order is not important):

- **exec** `<string>` = Executes the given program/script
- **state** `<value>` = Shows the display style of the menu item
- **submenu** `<value>` = Attaches a pull-right menu to the current item
- **return** `<value>` = Returns the “key value” associated with the menu entry

exec `<string>`

A "system" call will be made for `<string>`, which is enclosed in quotation marks. Any program that could be called from a command-line interface (complete with arguments) is legal here. The program name must be ≤ 220 characters. If you do not wish to have the program block further execution of ENlighten/DSM, run the program in the background by using the ampersand (`&`), just as you would from the command line.

state `<value>`

The default state of an added menu item is active; the menu item is displayed clearly and is selectable by the user. Or you can have the menu item displayed but grayed-out; it is viewable but not selectable by the user. To make a menu item inactive, use the value eight (8). This option also will cancel the display of any subsequent submenus for this item.

submenu `<name>`

You attach a previously defined submenu to the current menu item by giving its internal name (from the DEFINE statement) embedded in quotation marks. This attribute overrides any exec associated with the menu item; it can only display a pull-right menu.

return `<value>`

The Return value links a menu item to a particular set of ENlighten windows. The menu Category, along with the Return value, determines which window is then initiated. In a sense, these values are "reserved" codes. The default interface file has all the combinations of the Category values and Return codes. For example:

#2 - Configuration Section
- 1 = Session Preferences
- 2 = Alter Menu
- 3 = Pool Configuration
- 4 = User Authorization
- 5 = New User Templates
- 6 = EMD Data Expiration

In addition to the reserved Return codes for Menus, there are also reserved Return codes for submenu categories. For example:

#53 - Activity Submenu
1 = Who is Logged In
2 = Process Status
3 = CPU Summary

Although a Submenu, by default, belongs in an associated Menu category, since it carries its own Category and Return codes, it can be placed anywhere. For example, you could place both the User Security and System Integrity submenus on their own User-Defined menu and call that menu "System Security."

To change the names of any existing menu items, change either the 'label' or 'add' attribute arguments. The order of the menu items is not fixed, but be sure to keep all of a menu item's attributes with its corresponding add attribute.

A Utilities Submenu

To expand on the previous example of a user-defined `UserMenu1` menu, you could add the following items:

```
#
# The following user utility menu is a predefined example.
# It can be used to add you own scripts into the ENL main menu.
#
#  The definition exists here, but is currently deactivated in the
#  DEFINE 7 MAINMENU definition further down. To activate it, the
#  comment markers (#) must be removed from the following lines
#
#   #     add MENUBUTTON1
#   #     name    "USERSCRIPT"
#   #     icon  12
#   #     menu    USERMENU2
#
DEFINE  12  USERMENU2
    categ   12
    context "This is a mini-help line as the mouse moves over the
icon"
    label   "User Scripts 2"
    pinned  1
    add "Script 1"
    return  1
    exec    “xterm -e $ENLIGHTEN/bin/script1”
    add “Script 2”
    return  2
    exec    “xterm -e $ENLIGHTEN/bin/script2”
#
# The following user utility menu is a predefined example.
# It can be used to add you own scripts into the ENL main menu.
#
#  The definition exists here, but is currently deactivated in the
#  DEFINE 7 MAINMENU definition further down. To activate it, the
#  comment markers (#) must be removed from the following lines
#
DEFINE  12  USERMENU1
    categ   11
    context “Help subsystem, by category and action”
    label   “User Scripts 1”
    pinned  1
    add “Script 1”
    return  1
    exec    “xterm -e $ENLIGHTEN/bin/script1”
    add “Script 2”
    return  2
    exec    “xterm -e $ENLIGHTEN/bin/script2”
 ........
.......
.....
..
.

#    add MENUBUTTON1
#    name    “USER MENU 1”
#    icon    11
#    menu    USERMENU1
#    add MENUBUTTON1
#    name    “USER MENU 2”
#    icon    12
#    menu    USERMENU2
```

The Main Menu Window

All menus are attached to icons in the Main Window. This definition is shown in the interface file as the first **DEFINE 7** statement. Use the attribute `add MENUBUTTON1` to attach a menu to its window (left to right, in order of declaration). An internal `name` attribute is not necessary, but is usually included.

Use the `menu menu-name` attribute to attach the previously defined menu to the menu button (the menu-name is shown in the line `DEFINE 11 name`). Use the `icon icon-type` attribute to assign the icon to be displayed as the menu button. The icons are pre-defined:

2	The configure icon
3	A picture of a terminal
4	A picture of a disk
5	A picture of users
6	A picture of a terminal
7	A picture of a laser printer
8	A picture of a 1/4-inch tape
9	The Events icon
10	A picture of a lock
11	A picture of tools (User-Defined Menu)
12	A picture of tools (User-Defined Menu)

You can also have multiple User-Defined menu buttons, as long as the return codes for the menu items are unique across all of them. For example, you can define two utility menus, “UTILITY1” and “UTILITY2,” with three menu items each, with UTILITY1 having menu items with return codes 1–3 and UTILITY2 having menu items with return codes 4–6.