NAME

olwm – OPEN LOOK window manager for OpenWindows

SYNOPSIS

olwm [*options*]

DESCRIPTION

Olwm is a window manager for the X Window System that implements parts of the OPEN LOOK graphical user interface. It is the standard window manager for Sun's OpenWindows product, but it will work properly with any X11 system. The only requirements for running **olwm** are that the server have the OPEN LOOK glyph and cursor fonts available.

OPTIONS

Most command-line options have counterparts in the resource database. A command-line option will override any setting from the resource database.

–2d Use two-dimensional look. This is the default for monochrome systems.

–3d Use three-dimensional look. This is the default for color systems. This option is ignored for monochrome systems.

–bd *color*, **–bordercolor** *color*
Specifies the border color. See the description of the **BorderColor** resource.

–bg *color*, **–background** *color*
Specifies the background color. See the description of the **Background** resource.

–c, **–click**
Use click-to-focus mode. This is the default focus mode.

–depth *depth*
Specifies the depth of the visual in which **olwm** is to run. See the discussion in the Screen Resources section for further information about depths.

–display *display-string*
Specify the name of the display to manage. Overrides the DISPLAY environment variable, if any. In addition, the display string is exported to **olwm**'s environment, so processes forked from **olwm** will inherit this value.

–dsdm Specify that **olwm** should provide the Drop Site Database Management (DSDM) service. This is the default.

–f, **–follow**
Use focus-follows-mouse mode. Default mode is click-to-focus.

–fn *font-name*, **–font** *font-name*
Set the font for window titles.

–fg *color*, **–foreground** *color*
Specifies the foreground color. See the description of the **ForegroundColor** resource.

–multi Manage windows on all screens that a display supports. This is the default.

–name *resource-name*
Use *resource-name* to look up resources in the resource database.

–nodsdm

Specify that **olwm** should not provide the Drop Site Database Management service. The default is to provide the service.

–single
Manage windows for a single screen only, using the default screen for the specified display. Overrides the **–multi** option.

–syncpid *process-id*
When **olwm** has completed its initialization, it will send a signal (SIGALRM by default) to *process-id.* The signal will be sent only if this option is present. This is useful for running **olwm** from shell scripts (such as **.xinitrc**) in such a way that the script waits for **olwm** to finish its initialization, while leaving **olwm** as a child process of the shell script. This can be done using the following **sh**(1) construct:

```
sleep 15 & pid=$!
olwm –syncpid $pid &
wait $pid
```

–syncsignal *signal*
Specifies the signal to send instead of SIGALRM. The signal is specified as a number, not symbolically.

–visual *visual-class*
Specifies the class of the visual in which **olwm** is to run. See the discussion in the Screen Resources section for further information about visuals.

–xrm *resource-string*
Specify resources on the command-line. Resources specified here will override resources found in resource files.

DEBUGGING OPTIONS

The following options are strictly for debugging. They are not recommended for general use. Don't use them unless you know what you are doing.

–all Print a message for every event received.

–debug
Equivalent to turning on all debugging options.

–orphans
Print orphaned events. Orphaned events are events that are associated with a window or frame that has no entry in the frame hash table, or events that are not handled by the various event handlers.

–synchronize
Run the window manager in synchronous mode.

INTERNATIONALIZATION OPTIONS

–basiclocale *locale-name*
Specifies the basic OPEN LOOK locale category setting. This category will be the base for other locale categories.

–displaylang *locale-name*
Specifies the display language OPEN LOOK locale category. This category

affects the contents of workspace menu, window menu and notice messages.

–**numeric** *locale-name*
Specifies the numeric format OPEN LOOK locale category. This category affects the numeric format displayed in any message that contains numerics.

LOCALE HANDLING

The *locale* is the set of language and cultural conventions used by a program. The locale controls the language-dependent part of **olwm**'s behavior. The OPEN LOOK international extensions have defined several locale categories as follows:

Basic Locale
This is the basic setting for the entire locale mechanism. This category specifies internal character handling behavior.

Display Language
This category specifies the language used for displaying menus, notice messages, and error messages.

Input Language
This category specifies the language used for text input. This category has no effect on **olwm**, because it does not accept text input from the keyboard.

Date Format
This category specifies the format of date and time. This category has no effect on **olwm**, because it does not display any date and time information.

Numeric Format
This category specifies the format of displayed numeric data.

The Basic Locale setting determines the character set used by **olwm**. The other locale categories can differ from the basic setting, but they cannot require a different character set from the Basic Locale. The following restrictions thus apply:

1. If basic locale setting is the "C" locale, then all other locale categories must be in the "C" locale.
2. If the Basic Locale is set to a locale other than the "C" locale, then all other locale categories must be set either to a locale that uses the same character set as the basic setting, or to the "C" locale.

The following methods are available to inform **olwm** of the locale settings, listed in order of priority:

1. Command line options (such as –**basiclocale**);
2. by resource database; and
3. **setlocale**(3C) function defaults (e.g. LANG environment variable).

INPUT FOCUS

The *input focus* is the window that will receive keystrokes. **olwm** has two different input focus modes, which are different ways of transferring the input focus from one window to another. By default, **olwm** uses "click-to-focus" (also known as "click-to-type") mode. This means that you must click on the window in order to get the focus to it. While a window has the input focus, the pointer can be anywhere on the screen; the keyboard events will still go to that window. You can set the input focus to a window and simultaneously raise it to the top by clicking the left mouse button in the window's title bar or border.

olwm has another focus mode called "focus-follows-mouse." In this mode, whatever window the mouse is pointing to will receive the input focus. To switch the input focus from one window to another, you simply move the pointer to the other window; you don't have to click at all. Note, however, that to transfer the focus amongst subwindows of a single top-level window, you must click in the subwindow, or you must use focus transfer function keys (if available from the application).

The input focus mode can be controlled with command-line options or by entries in the resource database. Neither focus mode has inherent advantages. Which one you choose is a matter of personal preference.

MOUSE BUTTONS

OPEN LOOK defines three mouse button functions: SELECT, ADJUST, and MENU. On systems with three mouse buttons, these functions are mapped to buttons 1, 2, and 3 (left, middle, and right) respectively. On systems with two mouse buttons, SELECT is on button 1 (left) and MENU is on button 2 (right). ADJUST can be performed by holding down the Shift key while pressing button 1. On systems with a single mouse button, that button is SELECT. Holding Shift while pressing the button gives ADJUST, and holding Control gives MENU.

There is an alternate style of button handling for two-button mice: SELECT is button 1, ADJUST is button 2, and MENU is performed by holding down buttons 1 and 2 simultaneously. This technique is referred to as mouse button *chording.* Turns on the mouse chording mechanism. This allows two-button mice to have a different button binding than the OPEN LOOK defaults for mice with fewer than three buttons. When **OpenWindows.MouseChordMenu** is set to True, mouse button 1 is SELECT , mouse button 2 is ADJUST , and pressing the SELECT and the ADJUST buttons at the same time will act as the MENU button. For the default behavior when this resource is set to False, see the section "MOUSE BUTTONS" in the **olwm**(1) manual page.

MANIPULATING WINDOWS AND ICONS

Window Title Bar and Borders.

Clicking SELECT selects the window, raises it above other windows, and deselects any other objects. In click-focus mode, the focus is also transferred to this window. Pressing and holding SELECT and then dragging the mouse will move windows without raising them or setting the focus. If this window is selected, it and all other selected windows are moved simultaneously. Otherwise, just this window is moved, and it is not selected. If you hold down the Control key while you are moving a window, motion is constrained either vertically or horizontally, depending on which direction you move first.

Double-clicking SELECT on the window is the same as selecting the Full Size (or Restore Size) menu item. Clicking ADJUST will toggle the selected state of this window. If other windows or icons are already selected, they remain selected. ADJUST is useful for selecting several windows and icons. Pressing MENU will bring up the window menu. See the Window Menu section for further details. If the **Alt** key is held down, the mouse button functions become accessible anywhere over the window, not just over the title bar and borders. The modifier used can be changed; see the description of the **WMGrab** resource in the section on Modifier Customization.

Resize Corners.

You can resize a window by pressing and holding SELECT over any of the resize corners and then dragging the mouse to the new location. Releasing the mouse button will set the new size of the window. If you hold down the Control key while you are dragging, the resize operation is constrained to resize vertically or horizontally, depending on which direction you move first.

Window Button.

The Window Button is the small box with a downward-pointing triangle near the left end of the title bar. Pressing MENU over the window button will bring up the Window Menu. Clicking SELECT over the left mouse button on the Window Button will execute the window menu's default action. This will usually close the window into an icon. You can change the window menu's default action by holding down the Control key while manipulating the window menu.

Pushpin.

OPEN LOOK pop-up windows have a pushpin instead of a window button. The pin is either in or out, and you can click SELECT on the pin to move it to the other state. If the pin is out, pressing a command button inside the window will execute the command and then dismiss (take down) the window. If the pin is in, the window is "pinned" to the workspace, and it will remain on the screen even after you have pressed a command button in the window. This allows you to press several command buttons in the same window. Pulling the pin out (by clicking SELECT over it) will dismiss the window immediately.

Icons.

An icon represents a closed window. You can still do most of the same operations as with an open window. Moving and selecting icons with SELECT and ADJUST is exactly the same as for open windows. A similar version of the Window Menu is available on an icon by pressing MENU. Double-clicking SELECT will open the icon. Icons cannot be resized.

NON-RECTANGULAR WINDOWS

The X11 Non-Rectangular Window Shape Extension (commonly referred to simply as the SHAPE extension) allows windows to have arbitrary shapes. **Olwm** will handle these windows by giving them no decoration whatsoever. Shaped windows can be manipulated by using the WMGrab modifier (Alt by default) with the mouse buttons. (See the section on Modifier Customization for further details.) Shaped windows can be moved, resized, closed, opened, etc. like ordinary windows. The selection feedback for shaped windows is the presence of resize corners floating at the corners of the bounding rectangle of the window's shape.

SELECTIONS ON THE WORKSPACE

You can select a group of windows and icons by using the left or middle mouse buttons over the Workspace (the area of the screen outside of all windows and icons, commonly known as the "root window"). Pressing either SELECT or ADJUST and dragging the mouse will define a rubber-band rectangle. When you release the mouse button, the set of windows and icons enclosed by this rectangle will be operated on. If you created the rectangle using SELECT, the windows and icons within will be selected, and all other objects will be deselected. If you used ADJUST, the objects within will have their selected state toggled, and any other windows and icons already selected will remain selected.

MENU OPERATION

In general, pop-up menus are operated using the MENU mouse button. There are two methods of operating with an OPEN LOOK menu: the "click-move-click" method and the "press-drag-release" method. You choose the method either by clicking the MENU button (pressing and releasing it quickly) or by pressing it down and holding it. If you click the MENU button, the menu will pop up and will stay up indefinitely. To continue operating the menu, click the MENU button over a menu item. To dismiss the menu, click the MENU button on an area of the screen outside the menu. To operate menus in press-drag-release mode, press the MENU button and hold it down while you move the mouse. The menu will remain on the screen as long as you hold down the MENU button. To execute an action, move the pointer over a menu item and release the mouse button. To dismiss the menu, move the pointer outside the menu and release the MENU button.

Some menu items have a sub-menu. This is indicated by a right-pointing triangle at the right edge of the item. To activate a submenu, click on the item (in click-move-click mode) or move the pointer to the item and then move toward the right edge of the menu (in press-drag-release mode).

Some menus have pushpins. If a menu has a pushpin, it will initially be in the "out" state. If you click on the pin (in click-move-click mode) or move over it and release (in press-drag-release mode) you will pin the menu to the workspace. The menu will remain on the screen indefinitely and you can execute commands from it by clicking on its items. To remove the menu, move over the pin and click SELECT on it.

The behavior of menus can be customized using **olwm**'s resources. In the Global Resources section, see the entries for **ClickMoveThreshold**, **DragRightDistance**, **MultiClickTimeout**, and **SelectDisplaysMenu** for further information.

Some menus may have "accelerators" defined for them. See the section on Menu Accelerators for further details.

WORKSPACE MENU

Pressing MENU over the workspace brings up the Workspace Menu. This menu is customizable, but it typically contains at least the following items. (The items may appear in a different language depending on the current locale setting.)

Programs
This item has a sub-menu that allows you to invoke applications. The default Programs sub-menu contains all of the programs in the OpenWindows DeskSet. However, users typically customize this menu to contain many more programs and to contain nested sub-menus. See the section on Menu Customization for further information.

Utilities
This item has a sub-menu that contains several utility functions for the workspace, including Refresh (redisplay all windows on the screen), Lock Screen, and Save Workspace.

Properties...
This item brings up the Workspace Properties window, which allows you to view and customize settings of the OpenWindows environment.

Help... Brings up the table of contents of the Help Handbooks.

Desktop Intro...
Brings up a tutorial introduction to the Sun Desktop.

Exit Shuts down all applications and exits the window system. A confirmation notice will appear first to give you a chance to cancel the operation.

WINDOW MENU

The window menu of most windows has the following items. (The items may appear in a different language depending on the current locale setting.)

Close Closes the window to an icon. Any OPEN LOOK pop-up windows are closed into this icon as well. They will reappear when the icon is opened. This item is "Open" if you bring up this menu over an icon.

Full Size
Expands the window to the full height of the screen. If this has already done, the button is Normal Size instead of Full Size. Normal Size restores the window to the size it was before you did the Full Size operation. If the application has specified a maximum size for the window, this size is used for Full Size instead of the full screen height.

Move Starts the keyboard-based form of moving the window. Appears only if OPEN LOOK Mouseless Mode is enabled.

Resize Starts the keyboard-based form of resizing the window. Appears only if OPEN LOOK Mouseless Mode is enabled.

Back Moves the window behind all other windows.

Refresh
Clears and redisplays the window.

Quit Kills the program running in the window and removes the window. If the application has elected to participate in the WM_DELETE_WINDOW protocol, **olwm** sends a WM_DELETE_WINDOW ClientMessage instead of killing that window.

OPEN LOOK pop-up windows (as opposed to base windows) have a smaller window menu. It lacks the Close, Full Size, and Quit items, but it has two new items:

Dismiss
Causes the window to be dismissed. This button has a sub-menu with two items: This Window, which dismisses just this window, and All Pop-ups, which dismisses all pop-up windows owned by this application.

Owner?
Raises and flashes the title bar of the base window that "owns" this pop-up window.

MENU CUSTOMIZATION FILES

You can customize **olwm**'s Workspace Menu by putting a menu description into a file that **olwm** will read. When it starts up, **olwm** will first look for a file named by the OLWMMENU environment variable. If this variable does not exist, or if the file is not readable, **olwm** will then look in the file named **.openwin–menu** in your home directory. If this file is not present or is unreadable, **olwm** will fall back on the system default menu file. If, for some reason, the system default menu file cannot be found, **olwm** will use a minimal, built-in menu. The menu file that is read can also be modified by the display language locale setting. The locale name is used as a suffix for the filename. If a localized menu file is found, it is used in preference to the non-localized menu file. For example, if the display language locale is "japanese", the file **.openwin–menu.japanese** will take precedence over the file **.openwin–menu**.

Olwm will automatically re-read its menu file whenever the menu file changes. This lets you make many small changes to a menu file, trying out the modified menu after each change. The automatic re-reading can be controlled with the **AutoReReadMenuFile** resource.

If **olwm** encounters a syntax error during the reading of any menu file, a message is printed to the standard error file and the reading of this menu file is considered to have failed. **Olwm** will then attempt to read the next file in the sequence as described above.

MENU SPECIFICATION SYNTAX

The menu specification language has a number of keywords, all of which are in all upper case letters. The keywords are *not* translated into the language specified by the the locale category settings. Keywords are always in English.

Each line typically specifies one menu button. There are three fields on each line: a label, the optional keyword DEFAULT, and a command. The label is either a single word or a string enclosed in double quotes. This is the label that appears in the menu button. If the optional keyword DEFAULT appears next, this menu item becomes the default item for this menu. The rest of the line (excluding leading whitespace) is considered to be a command. It is executed by sending it to **sh**(1). Any shell metacharacters will be passed through to the shell unchanged. The command field can be extended onto the next line by placing a backslash '\' at the end of the line. The newline will not be embedded in the command.

A sub-menu is specified using the special keyword MENU in place of a command. A button is added to the current menu, and clicking or pulling right on this button will bring up the sub-menu. Subsequent lines in the menu file define buttons for the sub-menu, until a line that has the special keyword END in the command field is encountered. The label of the MENU line must match the label on the END line, otherwise an error is signaled. Sub-menus can be nested arbitrarily, bracketed by MENU and END lines with matching labels.

Sub-menus can be defined in a different file using either the MENU or the INCLUDE keyword. To include a sub-menu from another file, use a line with a label, either the MENU or the INCLUDE keyword, and then the filename. The file so named is assumed to contain lines that specify menu buttons. The sub-menu file need not have any MENU or END lines (unless it has sub-menus itself). The current file need not have a matching END line if the sub-menu is read from another file. Sub-menu files included with the

MENU keyword are considered to be an integral part of the menu tree, and any error encountered during reading of the file will cause the entire menu to be considered invalid. A sub-menu file included with the INCLUDE keyword is considered optional, and any error encountered during reading of the file is not considered fatal. If an error occurs during INCLUDE processing, a disabled (grayed-out) item is inserted in place of the sub-menu and processing of the current menu file continues.

To make a sub-menu pinnable, add the special keyword "PIN" after the END keyword on the line that ends the sub-menu definition, or after the TITLE directive (see below).

By default, the label in a menu button is used as the title of the sub-menu. This can be overridden by specifying a line that has the special keyword TITLE in the command field. The label from this line will be used as the sub-menu's title. This line can appear anywhere in the sub-menu definition. It does not add an item to the menu. In addition, if the PIN keyword follows the TITLE keyword on this line, the sub-menu will be made pinnable. This construct is useful for declaring that a sub-menu defined in a separate file be pinnable.

A line containing only the keyword SEPARATOR will add extra space before the next item.

The following keywords can be used in the command field of a menu item. They specify functions that are internal to **olwm,** that are not invoked by running a shell.

BACK_SELN
: Move the selected windows and icons behind other windows.

EXIT
: Kills all applications and exits the window manager after getting confirmation from the user. This is useful for exiting the entire window system.

EXIT_NO_CONFIRM
: Like EXIT but skips the confirmation notice.

FLIPDRAG
: Toggle the state of the **DragWindow** resource.

FLIPFOCUS
: Toggle the state of the **SetInput** resource.

FULL_RESTORE_SIZE_SELN
: Toggle the full-sized/normal-sized states of the selected windows and icons.

NOP
: No operation; don't do anything.

OPEN_CLOSE_SELN
: Toggle the opened/closed states of the selected windows and icons.

QUIT_SELN
: Quit the selected windows and icons.

PROPERTIES
: Bring up Workspace Properties.

REFRESH

Refresh causes all windows on the screen to be repainted.

REREAD_MENU_FILE
: Force an immediate rereading of the workspace menu customization file. **Olwm** will start a complete search for a menu file (as described in the *Menu Customization* section) and use the first valid file it finds.

RESTART
: Restart the window manager by issuing an **exec**(2) on **argv.** This shouldn't affect any running applications, nor should it cause the server to shut down.

SAVE_WORKSPACE
: Take a snapshot of the set of currently running applications, and put the command lines so obtained into the file ".openwin–init" in the user's home directory. This runs the command specified by the **SaveWorkspaceCmd** resource.

START_DSDM
: Start providing the DSDM service. See the section on Drag and Drop for further information.

STOP_DSDM
: Stop providing the DSDM service. See the section on Drag and Drop for further information.

WMEXIT
: Exit the window manager without killing any applications.

Here is an example root menu specification.

```
"My Custom Menu" TITLE

Programs            MENU
    "Command Tool"              DEFAULT cmdtool
    "Text Editor"       textedit
    Mail                mailtool
    "File Manager"      filemgr
    Other               MENU
        "Other Tools"           TITLE
        "Shell Tool"            shelltool
        "Icon Editor"           iconedit
        Clock                   clock
        "Perf Meter"            DEFAULT perfmeter
    Other               END
Programs            END PIN

"Repaint Screen"    REFRESH

"Properties ..."  PROPERTIES

Exit            EXIT
```

COLORMAP INSTALLATION

Olwm will handle colormap installation for windows that have colormaps other than the default colormap. There are two colormap focus modes: "color-follows-mouse" and "color-locked". They are roughly analogous to the corresponding modes for input focus. However, the colormap focus mode can be completely independent of the input focus mode. The mode in which the system starts up is determined by the **ColorFocusLocked** resource.

Olwm keeps track of a set of windows that are eligible to have their colormaps installed. This set includes all top-level windows of clients. If any clients have specified other windows in a WM_COLORMAP_WINDOWS property, these windows are included in the set as well. The windows listed in this property need not be top-level windows; they can be nested subwindows as well.

In color-follows-mouse mode, **olwm** keeps track of the location of the pointer and always keeps installed the colormap of the eligible window underneath the pointer. Thus, you can install the colormap of a particular window simply by sliding the pointer into it. The default colormap will be restored if you move the pointer back out into a window frame or into the workspace. In this mode, the WM_COLORMAP_WINDOWS properties are tracked for changes, but only to change the set of eligible windows. Changes to these properties only cause colormaps to be installed if the eligible window under the pointer has changed as a result of the set of eligible windows changing. In this mode, no window is considered to have the colormap focus; colormap installation entirely is under control of the user.

In color-locked mode, colormaps are not installed based on pointer motion. Instead, colormaps are installed explicitly by the user using function keys or by a program changing the contents of the WM_COLORMAP_WINDOWS property on its top-level window.

The user can install the colormap of a window (or subwindow listed in the WM_COLORMAP_WINDOWS property) by moving the pointer over the window or subwindow and pressing the Color-Lock key (which is bound to Control-L2 by default). This will install the colormap of the window or subwindow under the pointer, and it will also grant the colormap focus to the top-level window. When a window has the colormap focus, **olwm** will honor changes to this window's WM_COLORMAP_WINDOWS property by installing the colormap of the first window named in this property. In this way, the application whose window has the colormap focus can control colormap installation by altering the contents of the WM_COLORMAP_WINDOWS property.

Note that, according to the ICCCM, if WM_COLORMAP_WINDOWS does not include the top-level window, it is assumed to occur first in the list. If you want your program to request colormap installation via changes to WM_COLORMAP_WINDOWS, you must make sure that the top-level window appears somewhere in this property. Otherwise, **olwm** will always install the colormap of the top-level window.

The colormap focus may be given to a window in several other ways. If you press the Color-Lock key over a window's title bar or border, that window will be given the colormap focus and the first window in the WM_COLORMAP_WINDOWS property will be installed. If the **AutoColorFocus** resource is set, new windows will be given the colormap focus automatically. If the **ColorTracksInputFocus** resource is set, the colormap focus will always be given to the window that has the input focus.

If you press the Color-Lock key over the workspace, the default colormap will be installed, and any window with the colormap focus will lose it. The root window is then considered to have the colormap focus. At any time, you can revert to color-follows-mouse mode by pressing the Color-Unlock key. Any window with the colormap focus will lose it.

MAGNIFY HELP

Olwm provides on-line help through the Magnify Help utility for frames, icons, the Workspace and Window menus, window buttons, resize corners, pushpins, and the Workspace itself. This is done via a separate slave program, **olwmslave**(1)**.** The slave program is forked automatically when **olwm** starts up. The forking of the slave program can be controlled by the **RunSlaveProcess** resource.

MULTIPLE SCREENS

By default, **olwm** will manage windows on all screens of the display server. Most operations are unchanged from single screen operation. A window exists on a particular screen for its entire lifetime. The window cannot be moved from one screen to another, nor can it be resized to cross a screen boundary. Windows invoked from the Workspace menu will appear on the same screen as the menu. Magnify Help will appear on the same screen as the pointer when the Help key is pressed (F1 on x86 keyboards).

Previous releases required modifications to the user's **.xinitrc** script to start multiple instances of **olwm,** one for each screen. These modifications are no longer necessary. The default **Xinitrc** (which contains a single invocation of **olwm**) works for both single and multiple screen situations.

DRAG AND DROP

The OpenWindows drag and drop system relies on a third-party client (i.e. a client other than the source or destination clients of a drag and drop operation) to maintain a database of all possible locations on the screen where an object may be dropped. These locations are referred to as ''drop sites.'' This third party client is thus called the Drop Site Database Manager or DSDM. By default, **olwm** is configured to provide the DSDM service to clients. This can be controlled using the **StartDSDM** resource or the −**dsdm** and −**nodsdm** command-line options.

If you have customized your Workspace Menu (see the section on Menu Customization) you can add items that use the START_DSDM and STOP_DSDM menu keywords. Invoking a menu item bound to one of these keywords will enable or disable **olwm**'s providing of the DSDM service.

A standalone client **dsdm**(1) exists in order to provide the DSDM service in the case where **olwm** is not running or if it has been directed not to provide the DSDM service. Note that the START_DSDM and STOP_DSDM functions do not run an actual **dsdm** process; rather, they control whether **olwm** provides the DSDM service itself. It is not necessary to run **dsdm** if **olwm** is providing the DSDM service.

GLOBAL RESOURCES

Global resources in **olwm** consist of two resource components. The first component in the resource name is taken from the trailing pathname component of **argv[0]**. This value is typically 'olwm'. This name can be altered by using the −**name** command-line argument. The second resource component names the global attribute being set. It should be one of the names from the following list. Thus, to set the **AutoColorFocus** attribute, one would use "olwm.AutoColorFocus" as the resource specification.

Olwm will automatically pick up changes to many of these resources if the resource database changes at run-time. One can thus modify **olwm**'s behavior by changing the resource database with **xrdb**(1) or with Workspace Properties. If a resource value is specified on **olwm**'s command line, it will override the value in the resource database, and thus changing the resource's value in the database will have no effect on this resource setting.

Some resources are also interpreted by XView (see **xview**(7)) and are set by the Workspace Properties program (see **props**(1)). For these resources, **olwm** will also accept the string 'OpenWindows' as the first resource component. These resources are marked with an asterisk '*'.

Colors can be specified using the formats parsed by the Xlib XParseColor() function. Common formats are color names (see **showrgb**(1)) and explicit red, green, and blue values in hexadecimal, preceded by a '#'. For example, a cyan (full green and blue) would be specified with "#00ffff".

Boolean values can be specified with the words "true", "false", "on", "off", "yes", "no", "1", "0", "t", and "nil".

AutoColorFocus *(boolean)*
: Indicates whether newly appearing windows are to be given the colormap focus automatically. See the section on Colormap Installation for further details. *Default value: false.*

AutoInputFocus *(boolean)*
: Indicates whether newly appearing windows are to be given the input focus automatically. *Default value: false.*

AutoRaise *(boolean)*
: Raise windows automatically when they receive the focus. This is useful in click-to-focus if you always like to type into the topmost window. This is useful in focus-follow-mouse when the **AutoRaiseDelay** resource is set to a reasonable value. *Default value: false.*

AutoRaiseDelay *(integer)*
: Amount of time to delay, in microseconds, between a window receiving the focus and raising it above other windows. Effective only when the value of the **AutoRaise** resource is true. *Default value: 0.*

AutoReReadMenuFile *(boolean)*
: Specifies whether the menu file is to be re-read whenever it changes. If the value is True, this will causes olwm to perform several **stat**(2) filesystem operations every time the Workspace Menu is raised. This may cause a noticeable delay. If this delay is objectionable, it may be eliminated by setting AutoReReadMenuFile to False. In such cases, one can use the REREAD_MENU_FILE to re-read the Workspace Menu definition file. See the section on Menu Customization for further information. *Default value: true.*

Background *(color)*
Specifies the background color. This is used for the background of masked icons. Note: it is not used for the backgrounds of icon windows such as those used by XView (see **xview**(7)). This resource is also distinct from the **WindowColor** resource. *Default value: white.*

BasicLocale *(locale name)*
Specifies the basic OPEN LOOK locale category setting. See the section on Locale Handling for more details.

Beep *(enumeration)* *
Specifies the circumstances under which **olwm** should beep. Permissible values are the strings "always", "never", and "notices". The string "never" means that **olwm** should never beep, "notices" means that **olwm** should beep only when a notice appears, and "always" means that **olwm** will beep whenever it is appropriate. *Default value: always.*

BorderColor *(color)*
Specifies the color used for window and icon borders. *Default value: black.*

ButtonFont *(font name)*
Font to be used for buttons in menus and notices. *Default value: lucidasans-12.*

ClickMoveThreshold *(integer)*
This value is used when bringing up a menu. If the pointer moves more than this amount while the menu button is down, the menu is considered to be in press-drag-release mode. Otherwise, the menu is in click-move-click mode. *Default value: 5.*

ColorFocusLocked *(boolean)*
Specifies the initial state of the colormap focus policy. If true, the default colormap is locked into the hardware. If false, the colormap of the window under the pointer is kept installed. See the section on Colormap Installation for further details. *Default value: false.*

ColorTracksInputFocus *(boolean)*
If true, indicates that the colormap focus is to be set automatically to any window that receives the input focus. See the section on Colormap Installation for further details. *Default value: false.*

CursorFont *(font name)*
Specifies the font to be used for cursors. It is probably not useful to change this unless you have an alternate cursor font with the same encoding as the OPEN LOOK cursor font. *Default value:*
–sun–open look cursor––*–*–*–*–120–*–*–*–*–*–*.*

DefaultIconImage *(filename)*
Specifies a file containing a bitmap to be used as the default icon image.

DefaultIconMask *(filename)*
Specifies a file containing a bitmap to be used as the default icon mask.

DefaultTitle *(string)*

Specifies the string to be used in the title bar of windows that have not provided a string in the WM_NAME property. *Default value: No Name.*

DisplayLang *(locale name)*
Specifies the display language OPEN LOOK locale category. See the section on Locale Handling for more details.

DragRightDistance *(integer)* ∗
The number of pixels you must drag the mouse to the right in a menu item to bring up a sub-menu. The sub-menu always comes up when you move over the menu mark (the right-pointing triangle), regardless of the drag-right distance. *Default value: 100.*

DragThreshold *(integer)* ∗
This is the number of pixels the mouse must move while a mouse button is down in order to have the action be considered a drag. If the mouse moves fewer than this number of pixels while the button is down, it is considered to be click instead of a drag. *Default value: 5.*

DragWindow *(boolean)*
If true, drags the entire image of the window when you move it. Otherwise, just drags the window outline. *Default value: false.*

EdgeMoveThreshold *(integer)*
Specifies the amount of "hysteresis" provided when moving windows past the edge of the screen. When you move a window or an icon, it will pause when it touches the edge of the screen. This is to allow you to easily position windows right up against the edge of the screen. If you move farther, the window or icon will continue to move past the edge. You can prevent windows from ever lapping off the screen by setting an extremely large value (say, 10000) for this resource, and you can disable this feature entirely by specifying a value of zero. *Default value: 10.*

FlashCount *(integer)*
Number of times the title bar is flashed when the "Owners?" menu item is activated. *Default value: 6.*

FlashTime *(integer)*
Amount of time, in microseconds, for which the title bar is flashed when the "Owner?" menu item is activated. *Default value: 100000.*

FocusLenience *(boolean)*
If this is set to true, **olwm** will not enforce the ICCCM requirement that windows must have the input hint set in order to receive the input focus. This option is useful if you run clients that aren't ICCCM-compliant, like many X11R3-based clients. *Default value: false.*

Foreground *(color)*
Specifies the foreground color. This color is used mainly for the text of window and icon titles and in menus. *Default value: black.*

GlyphFont *(font name)*

Glyph font used for drawing OPEN LOOK graphics. Changing this font is mainly useful for changing its size. Specifying a different font, such as a text font, will result in undesirable behavior. *Default value: –sun–open look glyph–*–*–*–*–*–120–*–*–*–*–*–*.*

IconFlashCount *(integer)*
Number of times to flash the open/close "zoom" lines. *Default value: 3.*

IconFlashOffTime *(integer)*
Amount of time to pause, in microseconds, while open/close "zoom" lines are not visible. *Default value: 1.*

IconFlashOnTime *(integer)*
Amount of time to pause, in microseconds, while open/close "zoom" lines are visible. *Default value: 20000.*

IconFont *(font name)*
Font used for icon names. *Default: lucidasans-12.*

IconLocation *(enumeration)* *
One of the words "top-lr", "top-rl", "bottom-lr", "bottom-rl", "left-tb", "left-bt", "right-tb", or "right-bt". These specify that icons should be arranged along a particular edge of the screen, ordered from left to right or top to bottom as appropriate. The words "top", "bottom", "left", and "right" are synonyms for "top-lr", "bottom-lr", "left-tb", and "right-tb", respectively. *Default value: bottom.*

InvertFocusHighlighting *(boolean)*
In click-to-focus, the input focus is normally indicated by a solid rectangle in the title bar. In focus-follows-mouse, focus is normally indicated with two lines in the title bar. If this resource is true, the style of highlighting is inverted with respect to the focus style. This results in two lines for click-to-focus and a solid bar for focus-follows-mouse. *Default value: false.*

KeepTransientsAbove *(boolean)*
Specifies whether **olwm** should attempt to keep transient windows above their owner window. *Default value: false.*

KeyboardCommands *(enumeration)* *
Permissible values for this resource are **SunView1, Basic,** and **Full.** Values are case-sensitive. In **Full** mode, all OPEN LOOK Mouseless commands implemented by the window manager are active. See the section on Mouseless Navigation for further information. In **Basic** mode, the keys active are Open, Front, Help, and the colormap keys. In **SunView1** mode, the only keys active are Open and Front. *Default value: Basic.*

MenuAccelerators *(boolean)*
Determines whether menu accelerators are active. Used in conjunction with the **WindowMenuAccelerators** resource. Both must be set to true for menu accelerators to be active. *Default value: true.*

MinimalDecor *(list of strings)*
Specifies a list of windows that are to be decorated minimally. Decoration on

such windows includes only a thin border and resize corners, with no title bar or window button. The value should be a whitespace-separated list of strings. Each string should specify an application's class or instance name, as passed in the WM_CLASS property. Most applications set this property based on the name of the executable (i.e. argv[0]). For example, to specify that the clock and the calculator should be decorated minimally, you would use the following resource:

```
olwm.MinimalDecor: calctool clock
```

Many applications will allow you to override the value of the WM_CLASS property using the −**name** option on the command line. *Default value: (null).*

MouseChordMenu *(boolean)*
: If true, uses a chorded mouse button combination for MENU instead of shift keys. See the Mouse Buttons section for further details. *Default value: false.*

MouseChordTimeout *(integer)*
: Specifies the amount of time, in milliseconds, that **olwm** is to wait for subsequent events to disambiguate chorded mouse button event sequences. *Default value: 100.*

MultiClickTimeout *(integer)* ∗
: The time, in tenths of a second, that differentiates a double-click from two single clicks. This value is also used to distinguish the click-move-click and press-drag-release modes of pop-up menus. If the MENU button is held down longer than this amount of time, the menu is considered to be in press-drag-release mode, otherwise it is considered to be in click-move-click mode. *Default value: 5.*

Numeric *(locale name)*
: Specifies the numeric format OPEN LOOK locale category. See the section on Locale Handling for more details.

PaintWorkspace *(boolean)*
: If true, **olwm** will use the **WorkspaceColor** resource to set the workspace (root window) background color. If false, **olwm** will not change the root window background. This is useful If you prefer to set your own workspace color using **xsetroot**(1) or a similar program. *Default value: true.*

PointerWorkspace *(boolean)*
: If true, **olwm** will set the workspace (root window) cursor. If false, **olwm** will not change the root window cursor. This is useful if you prefer to set your own workspace cursor using **xsetroot**(1) or a similar program. *Default value: true.*

PPositionCompat *(boolean)*
: Turns on backward compatibility for older applications that have a habit of always setting the PPosition flag in the WM_NORMAL_HINTS property, even when they haven't set a position. This most often occurs with X11R3-based clients. Without backward compatibility, these windows will always appear in the upper-left corner of the screen. With backward compatibility, these windows will be positioned according to the default OPEN LOOK window placement

policy, along the diagonal of the screen. This option will not affect windows that have a geometry specified on the command line. *Default value: false.*

PopupJumpCursor *(boolean)* ∗
Specifies whether to warp the cursor to pop-up windows. *Default value: true.*

PrintWarnings *(boolean)*
Determines whether **olwm** will issue non-fatal warning messages (such as X protocol errors) to its standard error file. *Default value: false.*

RaiseOnActivate *(boolean)*
Specifies whether a window is to be raised when it is activated via a Mouseless command. *Default value: true.*

RaiseOnMove *(boolean)*
Tells **olwm** to raise a window whenever it is moved by the user. *Default value: false.*

RaiseOnResize *(boolean)*
Tells **olwm** to raise a window whenever it is resized by the user. *Default value: false.*

RefreshRecursively *(boolean)*
Determines how the Refresh menu items on the window and workspace menus operate. If the value is true, **olwm** will walk the window hierarchy and send exposure events to every window. This is useful for refreshing windows that have backing store. If the value is false, **olwm** will map a window and then unmap it, causing all windows underneath that do not have backing store get exposures. When this feature is on, the Refresh operation generates a large amount of client-server traffic. It may be useful to turn this feature off if the connection transport has low bandwidth or high latency. *Default value: true.*

ReverseVideo *(boolean)*
If true, reverses the sense of black and white on monochrome screens. Ignored for color screens. *Default value: false.*

RubberBandThickness *(integer)*
Specifies the thickness of the "rubber-band" line that is drawn when a window is resized, when a group of windows is selected by dragging a rectangle on the root, and when a window is moved and the value of the **DragWindow** resource is false. *Default value: 2.*

RunSlaveProcess *(boolean)*
If false, disables the running of **olwmslave**(1) at startup time. If the slave process is not running, Magnify Help will not be available on objects owned by **olwm** such as pushpins and resize corners. *Default value: true.*

SaveWorkspaceCmd *(string)*
The command to execute to perform the Save Workspace functionality. This command defaults to running owplaces(1) which saves the currently running clients into the OpenWindows startup script $HOME/.openwin–init. *Default value:*

owplaces –silent –multi –local –script –tw –output $HOME/.openwin–init

SaveWorkspaceTimeout *(integer)*
Number of seconds to wait while the Save Workspace operation is in progress. If the Save Workspace command has not completed within this amount of time, the operation is considered to have failed. *Default value: 30.*

SelectDisplaysMenu *(boolean)* *
If true, pressing the SELECT mouse button will bring up a menu item's sub-menu (if any) instead of executing the sub-menu's default action. *Default value: false.*

SelectionFuzz *(integer)*
Number of pixels of "fuzz" to be applied when selecting windows and icons by dragging a rectangle on the workspace. Consider an object that lies almost entirely within the selection rectangle, but that laps outside the rectangle by a few pixels. The object will be considered to be within the selection rectangle if it laps outside by fewer than or equal to "fuzz" pixels. *Default value: 1.*

SelectToggleStacking *(boolean)*
If true, double-clicking on a window will push it to the back instead of zooming it to and from its full size. *Default value: false.*

SelectWindows *(boolean)*
If false, the SELECT mouse button will not select windows and icons. Its other functions are unaffected. The ADJUST mouse button can still be used to select windows and icons. *Default value: true.*

ServerGrabs *(boolean)*
Controls whether **olwm** grabs the server while menus and notices are up. *Default value: true.*

SetInput *(enumeration)* *
Controls the input focus mode. If the value is "select", it means click-to-focus. If the value is "followmouse", it means focus-follows-mouse. *Default value: select.*

ShowMoveGeometry *(boolean)*
Indicates whether the geometry box should be shown while moving windows and icons. *Default value: false.*

ShowResizeGeometry *(boolean)*
Indicates whether the geometry box should be shown while resizing windows. *Default value: false.*

SnapToGrid *(boolean)*
Determines whether icons will snap to a grid when they are moved. *Default value: false.*

StartDSDM *(boolean)*
Determines whether **olwm** will provide the DSDM service. See the section on Drag and Drop for further details. *Default value: true.*

TextFont *(font name)*
Font used in the text of notices. *Default: lucidasans-12.*

TitleFont *(font name)*
Font used in title bars atop windows and menus. *Default: lucidasans-12 Bold.*

TransientsSaveUnder *(boolean)*
Specifies whether the save-under attribute of frames of transient windows is to be forced on. *Default value: false.*

TransientsTitled *(boolean)*
Specifies whether transient windows should have title bars. Normally, transient windows have a title bar and resize corners, but no window button or pushpin. Setting this resource to false will remove the title bar from transient windows. *Default value: true.*

Use3D *(boolean)*
Specifies whether to use 3D OPEN LOOK when possible. If false, 3D look is never used. If true, 3D is used unless the display hardware cannot support it. *Default value: true.*

Use3DFrames *(boolean)*
Specifies whether to use a 3D look for the frame borders. If true, the frames will be given a 3D look; otherwise, they have the same thick border as in 2D look. Some people prefer the look of 3D frames, but it is more difficult to distinguish selected from unselected windows with this option turned on. *Default value: false.*

Use3DResize *(boolean)*
Specifies whether the window resize corners are to be in the 3D look. If false, the 2D look is used for window resize corners. *Default value: true.*

WindowCacheSize *(integer)*
Olwm keeps a cache of windows in order to minimize unnecessary window creation and destruction. The value of this resource specifies the size of this cache. Setting this resource to zero disables the window cache. *Default value: 500.*

WindowColor *(color)* *
Specifies the color of windows. This is the "BG1" color for 3D OPEN LOOK. It is used for the backgrounds of windows, menus, and notices. The 3D effect is achieved by using highlight and shadow colors derived from this color. *Default value: #cccccc.* This specifies a 20% gray value.

WindowMenuAccelerators *(boolean)*
Determines whether menu accelerators are active. Used in conjunction with the **MenuAccelerators** resource. Both must be set to true for menu accelerators to be active. *Default value: true.*

WorkspaceBitmapBg *(color specification)*
Specifies the background color used for the workspace bitmap when the **WorkspaceStyle** resource is "tilebitmap". *Default value: black.*

WorkspaceBitmapFg *(color specification)*
Specifies the foreground color used for the workspace bitmap when the **WorkspaceStyle** resource is "tilebitmap". *Default value: white.*

WorkspaceBitmapFile *(filename)*

Specifies a X bitmap file that will be used for the workspace background when WorkspaceStyle is "tilebitmap". If the filename is not a full path name, the following directories are searched:

$OPENWINHOME/etc/workspace/patterns
$OPENWINHOME/include/X11/include/bitmaps
/usr/X11/include/X11/include/bitmaps

Default value: gray.

WorkspaceColor *(color)* ∗

Specifies the color for the workspace (root window). On startup, **olwm** will set the root window's background color to the color specified by this resource, and it will restore the default background on shutdown. To turn off this behavior, see the description of the **PaintWorkspace** resource. *Default value: #40a0c0.* This specifies a light blue color. *Note:* earlier versions of **olwm** would accept a bitmap file name as the value of the **WorkspaceColor** resource. This is no longer supported, and the **WorkspaceBitmapFile, WorkspaceBitmapBg,** and **WorkspaceBitmapFg** resources should be used instead.

WorkspaceStyle *(enumeration)*

This controls how the workspace is painted. If the value is "paintcolor", the solid color specified by the **WorkspaceColor** resource is used. If the value is "tilebitmap", the workspace is tiled with a bitmap using the **WorkspaceBitmapFile, WorkspaceBitmapBg,** and **WorkspaceBitmapFg** resources. If the value is "default", the server default root-weave pattern is used. If the value of the **PaintWorkspace** resource is false, then all of these resources are ignored and the workspace color or pattern is left unchanged. *Default value: paintcolor*

SCREEN RESOURCES

In addition to the global resources described above, **olwm** also uses screen-specific resources. The first component of the resource specification is the trailing pathname component of **argv[0]**. The second component is the screen number appended to the string 'screen'. The screens are numbered sequentially starting from zero. The third component of the resource name is the name of the resource itself. For example,

olwm.screen1.ReverseVideo: true

enables reverse video on screen number 1 for **olwm.** To affect all screens, you can use resource wildcarding. For example, 'olwm∗ReverseVideo: true' will set reverse video for all screens **olwm** manages.

Unlike many of the global resources, the screen-specific resources are only applied at olwm startup.

The following resources are available both globally and on a per-screen basis. A screen-specific resource overrides the corresponding global setting for that screen. Note that screen specific settings for WorkspaceColor and WindowColor will only affect **olwm**; this may cause clashes with XView clients which only use the global setting.

Background
BorderColor
Foreground
ReverseVideo
WindowColor
WorkspaceColor

The following resources allow the selection of visuals other than the screen's default. Available visuals may be listed with the **xdpyinfo**(1) command.

Depth *(integer)*
Specify the visual depth to be used when searching for visuals. *Default value: none.*

Visual *(enumeration)*
Specify the visual class to be used when searching for visuals. Valid visual classes are **StaticGray**, **GrayScale**, **StaticColor**, **PseudoColor**, **TrueColor**, and **DirectColor**. Names are case-sensitive. *Default value: none.*

VisualID *(id)*
Specify the visual ID to be used. Note: specifying a visual by its ID is not portable, as IDs may vary from server to server and even from one invocation of a server to the next. *Default value: none.*

MOUSELESS NAVIGATION

Olwm implements OPEN LOOK Mouseless operations. This is a set of functions bound to keys that enable one to use the window system entirely without a pointing device. Some Mouseless functions are also useful for "cross-over" users, who may want to use them as accelerators for mouse-based operations. The full benefits of Mouseless operations are realized in click-to-focus mode, although the Mouseless operations can still be used in focus-follows-mouse mode.

To use the Mouseless functions, you must make sure that the **KeyboardCommands** resource value is "Full". Other settings for this resource will leave most of the Mouseless functions disabled. For further details, see the description of the **KeyboardCommands** resource in the Global Resources section. Enabling Mouseless operation only activates keyboard-based functions. It does not affect mouse functions in any way.

One can navigate from window to window using the Next Application, Previous Application, Next Window, and Previous Window functions, bound by default to Alt-n, Alt-Shift-n, Alt-w, and Alt-Shift-w, respectively. (See the section on Mouseless Navigation for more detailed information.) You can bring up both the window and the workspace menu using Alt-m and Alt-Shift-m, respectively. Once a menu is up, you can navigate through it by using the arrow keys or by pressing the first letter of the menu item you want to go to. You can execute the current item by pressing Return, or you can cancel the menu using Stop or Escape.

When Mouseless navigation is turned on, Move and Resize items will appear on the window menu. These items provide an alternative technique for moving and resizing windows. They can be invoked using the mouse, using the Mouseless menu navigation functions from the keyboard, or by using Menu Accelerator keys (although they are not bound to any accelerator keys by default). After selecting either of these items, you will be put into a mode where you can move or resize the window using keyboard keys. In Move mode, you can use the arrow keys to move the window in the desired direction. You can also hold down the Control key to "jump" the window by a larger distance each time you press an arrow key. You can press Return to accept the new location, or you can press Escape or Stop to abort the move operation.

In Resize mode, the first arrow key selects the edge you are moving, and subsequent arrow keys move that edge. For example, to shrink a window from the right (that is, to move its right edge to the left) you would first enter resize mode, press the right arrow key to select the right edge, and then press the left arrow key to move this edge to the left. As in move mode, you can hold down Control to "jump" the edge by a greater increment. You can press Return to accept the new size, and you can press Escape or Stop to abort the resize operation.

MENU ACCELERATORS

Olwm supports accelerator keys for certain items on the Window Menu. By default, the items for which accelerators are enabled are Close (Meta-W) and Quit (Meta-Q). Pressing these key combinations will operate on the window or icon that has the input focus. Other Window Menu items are not bound to key combinations, but can be bound with resources. See the Key Binding Resources section (below) for further information. When a menu accelerator key is active for a particular function, an indication of this appears at the right edge of the menu item. Key combinations with modifiers are displayed in a self-evident fashion, except for the Meta modifier, which is displayed as a diamond mark. The meta keys are marked with diamonds on SPARC keyboards. On x86 keyboards the meta key is simulated by pressing the Control and Alt keys at the same time.

The default menu accelerator bindings may conflict with certain popular applications (such as Emacs or the Athena text widget). It is thus possible to disable menu accelerators on a per-application basis. To disable menu accelerators, add a resource of the form

olwm.Client.*class*.MenuAccelerators: false

to the resource database, where *class* is the application’s class or instance name as written in the WM_CLASS property. For instance, to disable menu accelerators for Emacs, one would add the following

olwm.Client.Emacs.MenuAccelerators: false

to the **.Xdefaults** file.

KEY BINDING RESOURCES

Key bindings for mouseless navigation functions and menu accelerator keys are specified using resources. There is one resource per function, and the value of the resources are the keys to which the function is bound. The resource value consists of a comma-separated list of key specifications. Each key specification consists of a keysym optionally followed by modifier keysyms; the modifier keysyms are separated by ‘+’ signs. For example, to bind a function to F2, control-F3, and alt-shift-F4, one would use the value:

F2,F3+Control,F4+Shift+Alt

Any keysym whose key is in the modifier mapping may be used as a modifier. The following can also be used as aliases for common modifier keysyms: **Shift**, **Lock**, **Control**, **Ctrl**, **Ctl**, **Meta**, **Alt**, **Super**, and **Hyper.**

Resource names are prefixed with the trailing pathname component of **argv[0]**, followed by **KeyboardCommand** for mouseless navigation functions, or **MenuAccelerator** for menu accelerator keys, followed by one of the resource names from the following list. (Note that the **KeyboardCommand** resource component is singular, and is not to be confused with the **KeyboardCommands** global resource name.) For example, the resource specification for setting the Stop function would be:

olwm.KeyboardCommand.Stop

and the resource specification for setting the Back menu accelerator function would be:

olwm.MenuAccelerator.Back

Each item in this list is followed by its default keyboard binding and a description of what the function does. Items marked with an asterisk '*' involve keyboard grabs. Items not marked with an asterisk are active only while **olwm** is in a mode, such as when a menu is up. Items marked with an exclamation point '!' are menu accelerators and are specified using the **MenuAccelerator** resource component as described above. Items not marked with an exclamation point are considered mouseless navigation functions and use the **KeyboardCommand** resource component.

Most of the mouseless navigation functions that use grabs are active only when the **KeyboardCommands** resource is set to **Full.** The menu accelerator functions all use grabs, and they are controlled by the global resources **MenuAccelerators** and **WindowMenuAccelerators.** For further information, see the description of these resources in the Global Resources section.

Stop *(L1, Escape)*
Abort the current mode or action.

DefaultAction *(Return, Meta-Return, Enter)*
Execute the default action for the current menu or notice.

Select *(space)*
Select the current button.

Adjust *(Alt-Insert)*
Toggle the selected state of the current object.

Menu *(Alt-space)*
Bring up a menu on the current object.

InputFocusHelp *(?, Control-?)*
Bring up Help on the object with the input focus.

Up *(up-arrow)*
Move up one item.

Down *(down-arrow)*
Move down one item.

Left *(left-arrow)*
Move left one item.

Right *(right-arrow)*
Move right one item.

JumpUp *(Control up-arrow)*
Move up ten items.

JumpDown *(Control down-arrow)*
Move down ten items.

JumpLeft *(Control left-arrow)*
Move left ten items.

JumpRight *(Control right-arrow)*
Move right ten items.

RowStart *(Home, R7)*
Move to the start of the current row.

RowEnd *(End, R13)*
Move to the end of the current row.

DataStart *(Control-Home)*
Move to the start of the data.

DataEnd *(Control-End)*
Move to the end of the data.

FirstControl *(Control-[)*
Move to the first item.

LastControl *(Control-])*
Move to the last item.

NextElement *(Tab, Control-Tab)*
Move to the next item.

PreviousElement *(Shift-Tab, Control-Shift-Tab)*
Move to the previous item.

Open *(Alt-L7)* ∗
Open the object with the input focus.

Help *(Help)* ∗
Bring up Magnify Help on the object under the pointer.

LockColormap *(Control-L2)* ∗
Install the colormap of the subwindow under the pointer, and give the colormap focus to the top-level window containing the pointer. See *Colormap Installation* for further details.

UnlockColormap *(Control-L4)* ∗
Revert to color-follows-mouse mode, and unset colormap focus. See *Colormap Installation* for further details.

Front *(Alt-L5)* ∗
Bring the object with the input focus to the front.

FocusToPointer *(Alt-Shift-j)* ∗
Set the focus to the window under the pointer.

NextApp *(Alt-n)* ∗
Move the focus to the next base window. Windows are ordered clockwise starting at the top. Icons come after all windows, also in a clockwise fashion. Order proceeds from the last icon on a screen to the first window of the next screen. After the last screen, the order wraps back around to the first screen.

PreviousApp *(Alt-Shift-n)* ∗
Move the focus to the previous base window. See **NextApp** for details about the window traversal order.

ToggleInput *(Alt-t)* ∗
Move the input focus to the previous window that had the input focus.

NextWindow *(Alt-w)* ∗
Move to the next window in the family of windows consisting of a base window and a set of pop-up windows. Windows are ordered clockwise, starting at the top of the screen.

PreviousWindow *(Alt-Shift-w)* ∗
Move to the previous window in the family of windows consisting of a base window and a set of pop-up windows. Windows are ordered clockwise, starting at the top of the screen.

TogglePin *(Meta-Insert)* ∗
Toggle the state of the pin of the window with the input focus.

SuspendMouseless *(Alt-z)* ∗
Temporarily suspend all key grabs associated with Mouseless operation.

ResumeMouseless *(Alt-Shift-z)* ∗
Resume grabs after temporary suspension.

QuoteNextKey *(Alt-q)* ∗
Pass the next key sequence to the application with the focus, ignoring any grabs.

Refresh *(no binding)* ∗!
Repaint the window with the focus.

Back *(no binding)* ∗!
Move the focus window behind other windows.

OpenClose *(Meta-W)* ∗!
Toggle the open/close state of the window with the focus.

FullRestore *(no binding)* ∗!
Toggle the full-sized/normal-sized state of the window with the focus.

Quit *(Meta-Q)* ∗!
    Quit the window with the focus.

Owner *(no binding)* ∗!
    Flash the owner window of the pop-up window with the focus.

WorkspaceMenu *(Alt-Shift-m)* ∗
    Bring up the workspace menu.

WindowMenu *(Alt-m)* ∗
    Bring up the window menu on the window with the focus.

Move *(no binding)* ∗!
    Move the window with the focus.

Resize *(no binding)* ∗!
    Resize the window with the focus.

OpenClosePointer *(L7, Meta-w)* ∗
    Toggle the open/close state of the window or icon under the pointer.

RaiseLower *(L5)* ∗
    Raise the window under the pointer if obscured by other windows. Otherwise, lower the window if it obscures other windows.

MODIFIER CUSTOMIZATION

Olwm will alter the operation of certain mouse-based functions based on the state of the modifier keys. The relationship between the alteration and the associated modifier keys is controlled by a set of resources. Resource names are prefixed with the trailing pathname component of **argv[0]**, followed by **Modifier,** followed by a resource from the list below. For example, the resource specification to bind the Reduce modifier would typically be

    olwm.Modifier.Reduce

The value of each resource is a comma-separated list of modifier keysyms. Each item in this list is followed by its default modifier and a description of what it does.

Constrain *(Control)*
    Constrain a move or resize operation to be only on a horizontal or vertical direction.

Ignore *(Lock, NumLock, mod5, Mode_switch)*
    The set of modifiers to be ignored when processing mouse button events. This resource should contain the set of locking modifiers, so that mouse actions are still interpreted properly even while locking modifiers are in effect. The *mod5* modifier is included in this set because XView places function keys into this row in the modifier mapping table for use with quick-move and quick-copy operations.

Invert *(Shift)*
    When moving windows, temporarily invert the sense of the **DragWindow** resource. When resizing a window, temporarily move the window as long as

this modifier is held down. Return to resizing when the modifier is released.

Reduce *(Meta)*
When moving windows, reduce the amount of mouse motion by a factor of ten.

SetDefault *(Control)*
Sets the default item for a menu.

WMGrab *(Alt)*
Using the WMGrab modifier allows access to the mouse button functions anywhere over the window, not just over the window's title bar and border.

ENVIRONMENT

DISPLAY
Specifies the X11 server to which to connect.

LANG, LC_CTYPE, LC_MESSAGE, LC_TIME
These variables specify which locale to use when other methods of locale announcement are not available. (See the section on Locale Handling for more details.)

OLWMMENU
Specifies a file to use for the Workspace Menu.

OPENWINHOME
Specifies the location of the OpenWindows software.

FILES

$HOME/.openwin–menu.*localename*

$HOME/.openwin–menu
Contains the user-customized Workspace Menu specification.

$OPENWINHOME/lib/openwin–menu.*locale-name*

$OPENWINHOME/lib/openwin–menu
Contains the default Workspace Menu specification.

$HOME/.openwin–init
Stores the command lines obtained during the Save Workspace operation.

$OPENWINHOME/lib/app–defaults/Olwm

$OPENWINHOME/lib/locale/*locale-name*/app–defaults/Olwm
Specifies system-wide default resource values.

TRADEMARKS

OPEN LOOK is a trademark of AT&T.
The X Window system is a trademark of the Massachusetts Institute of Technology.
OpenWindows is a trademark of Sun Microsystems, Inc.

REFERENCES

Rosenthal, David S.H. *Inter-Client Communication Conventions Manual for X11.* Copyright 1989 by the Massachusetts Institute of Technology. This document is commonly known as the ICCCM. It is an X Consortium Standard that specifies conventions to which all X11 clients must adhere.

OPEN LOOK Graphical User Interface Functional Specification. Copyright 1989 by Sun Microsystems, Inc. Addison-Wesley Publishing Company, Inc. ISBN 0–201–52365–5.

OPEN LOOK Graphical User Interface International Extensions Functional Specification. Draft 1.1 (May 10, 1990). Copyright 1990 by Unix International.

SEE ALSO

dsdm(1), **olwmslave**(1), **openwin**(1), **owplaces**(1), **props**(1), **setlocale**(3C), **xinit**(1), **Xsun**(1)

NOTES

The resource names do not follow any classing structure. There is no general way to specify resources on a per-client basis.

There is no way to reconfigure the mouse buttons.

The uses of the modifier keys described in the Modifier Customization section interferes with the button bindings for one- and two-button mice. The default value of Modifier.Invert is Shift, which interferes with using shift-button1 for ADJUST. The default value of Modifier.Constrain is Control, which interferes with using control-button1 for MENU (on one-button mice only). One can set the Modifier.Invert and Modifier.Constrain resources to null (or to other modifiers) to avoid these conflicts, allowing full access to ADJUST and MENU on systems with one- and two-button mice. There is still a further conflict, as the default value of Modifier.SetDefault is also Control. Using control-button1 on a one-button system will bring up the menu, but will set the menu's default item. One must release the Control key after the menu is up in order to get normal menu operation. The choice of Alt as the default value for Modifier.WMGrab may conflict with some applications' key bindings.

The Exit menu item on the Workspace Menu doesn't really shut down the server. It kills off all clients being managed by the window manager, and then it exits the window manager itself. This works properly if some outside agent such as **xinit**(1) or **xdm**(1) is waiting for the window manager or a client to exit. The outside agent will take care of shutting down the server or reinitializing it. If you've started up the server a different way, this option may not work. Instead, the server will be left running with no clients and no window manager running, and you will have to login from elsewhere to kill the server. A common cause of this problem is an **.xinitrc** script that inadvertently leaves a non-windowed application (such as a daemon) running in the background. If the **.xinitrc** script ends with the **wait** shell command, it will never terminate. The fix is to change the script to either wait for a particular process-id, or to run the daemon in a subshell:

```
(daemon &)
```

Olwm is fairly simplistic about how it manages its keyboard bindings. For example, if you bind a function to control-F2, **olwm** will grab F2 with the Control modifier and with all combinations of the Lock and NumLock modifiers. If another locking modifier is in effect, **olwm**'s passive grab will not be activated, and thus the function will not work.

Olwm cannot manage multiple locales at one time, therefore all clients should be running in the same locale. The "C" locale is the exception. Applications using the "C" locale (such as non-internationalized applications) can be mixed with applications using one other locale.

Olwm does not handle different sizes of the glyph fonts well. Each locale can define a different size for the default font (for example, the default glyph font size is 12 for the "C" locale and is 14 for the "japanese" locale). **Olwm** does not re-position the window decorations after switching locale, therefore the window decorations may appear to be wrong. To remedy this problem partially, **olwm** will not change the font when locale is switching from non-"C" locale to the "C" locale.

There is no input focus feedback for non-rectangular windows. The title string of non-rectangular windows cannot be displayed. Non-rectangular icon windows are not supported.

Olwm will not dynamically track screen-specific and client-specific resources. Changes to global resources, key binding resources, and modifier resources are applied dynamically.

The interaction of the **AutoColorFocus**, **ColorFocusLocked**, and **ColorTracksInputFocus** resources and the color locking and unlocking keys is overly complex.

Changing the Display Language locale setting or editing the menu specification file will cause **olwm** to unpin any menus that were pinned at the time.

Resources that specify time values use inconsistent units. Some resources are in tenths of a second, some are in milliseconds, and some are in microseconds.