You will need to know the name of the Remote Shell program on your host. For many machines, this is ‘rsh’. On some System V machines the program is ‘rcmd’. On some HP systems it is ‘remsh’. Edit the ‘hdsrunwm.c’ source file and set the REMOTE_SHELL_PROGRAM definition appropriately.

To compile, execute the command:

```
cc -o hdsrunwm hdsrunwm.c
```

If you are using an alternate compiler, such as GNU/C, you may need to use a command such as ‘gcc’. This will produce a ‘hdsrunwm’ executable file.

3. Copy the ‘hdsrunwm’ file into a directory which will be included in your execute search path when you want to start the window manager. Possibilities include /bin, /usr/bin/X11, /usr/local/bin, etc.

4. The ‘hdsrunwm’ program uses the DISPLAY environment variable to determine the hostname of the ViewStation on which to start the window manager.

The ‘hdsrunwm’ uses the Remote Shell program to start the command. Most Remote Shell programs require a valid hostname instead of a numeric IP address. Therefore, if your Remote Shell program requires a hostname (as do most), make sure your DISPLAY environment variable will contain a hostname and not simply an IP address.

The ‘hdsrunwm’ program will print a warning message if a number IP address is present in the DISPLAY variable.

5. Where you would normally start your window manager, substitute the ‘hdsrunwm’ command.

For example, if you start your ‘mwm’ in the .xsession file, replace the line

```
mwm -option1 -option2 -option3
```

with

```
hdsrunwm mwm -option1 -option2 -option3
```

```
Menu RootMenu
{
    "Root Menu"                 f.title
    "New Window"                f.exec "xterm &"
    "Applications"              f.menu "ApplicationsMenu"
    "Calculator"                f.exec "xcalc &"
    "Clock"                     f.exec "hdslocal hdsdclock &"
    "HDS Local Clients"         f.menu "HdsClientMenu"
    no-label                    f.separator
    "Restart..."                f.restart
    .                           .
    .                           .
```

and so on.

If you are making changes to the HDSperx clients' configuration, you can see the changes by using the "rehash" command in the Console shell window. This rehash command forces a reading of the configuration files. For MWM menus however, this rehash command has no effect, since MWM rebuilds its menus only when it is started. You must use the "Restart" command to see the new menus and configurations for MWM.

2.4.14 HDSrunwm Launcher Program

The 'hdsrunwm' program provides an alternate way of starting a local window manager on an HDS ViewStation. If the 'hdsrunwm' program is used on the host to start the ViewStation window manager, any window manager menu commands will also be started by the 'hdsrunwm' program. This allows these menu commands to inherit the same environment as they would if the window manager were being run on the host.

The 'hdsrunwm' program requires HDSware Version 3.0.3 or later.

To use the 'hdsrunwm' program:

1. Check the *hds-fx/contrib/hdsrunwm/execs* directory for an executable suitable for your host type. If you find the correct executable, use the 'uncompress' command on the file. For example,

```
uncompress hdsrunwm.SUN4.Z

mv hdsrunwm.SUN4 hdsrunwm
```

You may need to set the 'executable' permissions for the file. For this, you can use the command:

```
chmod +x hdsrunwm
```

2. If a suitable executable does not already exist for your host type, set the current directory to *hds-fx/contrib/hdsrunwm/src*. The 'hdsrunwm.c' source can be used to compile an executable for your host.

A quick and simple test for the “rsh” command is to enter:

```
rsh localhost env
```

which will show you the path variables and environments which are set in your .cshrc file. If variables are missing from this response, they will also be missing for the local window manager started by rsh.

Another simple test is to do an “rcp” between two of the hosts on the network to test its operation independent of the FX and its file transfer requirements.

The ViewStation’s requirements are simple: the ability to start a remote shell and do a remote copy. Added features, like nameservices, and configurations, like special user environments, can make this simple requirement into a complicated set of tasks.

There are many situations and problems of this type. Almost without exception, the problems can be traced to incompatibilities within the system configuration. HDS Technical Support people will be glad to help you examine the problem, but they are not experts on your system. Use your system administrator and your network utilities to try to trace the problem.

2.4.13 Using HDSperx Clients with MWM

The HDSperx are set up to run from a root menu submenu automatically. Their operation from the menu is simple, since they start with just a mouse click on their entry.

Control of the HDSperx submenu and its name is handled by two Mwm resources:

```
Mwm*autoClientMenu:  True
Mwm*autoClientMenuName:  HDSperx
```

These are both Mwm resources and should be used with other Mwm resources in .Xdefaults or .mwmrc. (Resources for the HDSperx clients themselves are HDS resources and should be used only in the HDS client configuration file.)

If you want to load the HDSperx submenu automatically, use "Mwm*autoClientMenu: True"; this is the default setting. If you want to move the menu from its top position, you should set the "autoClientMenu" to False and locate the tools menu where you want it. You can also use the "autoClientMenuName" to enter a new menu name, which might be "HDS Local Clients". By default, the ViewStation’s local MWM has an internally defined menu called "HdsClientMenu" which can be explicitly referenced in your .mwmrc file menu table.

```
# Menu RootMenu entries
    "HDS Local Clients"                 f.menu "HdsClientMenu"
```

The MWM window manager can also start HDSperx clients explicitly, without using the HDSperx menu. For example, you can start a local HDSdclock with an .mwmrc file containing menu entries:

.mwmrc, .openwin-menu, etc. You must insure that these files are read, which means both that the locations are correctly specified and also that the "rcp" command succeeds. The ViewStation attempts to read these files in a number of locations; use the Console window for messages to see the file names and locations it looks for. These may be different than the locations your host used for host-based operation of the window managers.

Note that the shell startup files may not produce any output messages or the "rcp" command will fail and an "RCP protocol screwup" message will be sent. This is a restriction inherent in the "rcp" protocol.

5) Resource file contents - the contents of the resource file may cause problems. Comment lines, variable names, etc. must conform to the window manager's format and syntax. For example, resources written as C code cannot be imported without some syntax adjustments. These are often tricky problems to find and may depend on syntax differences between window manager versions. Check the **man** pages of the window managers; the ViewStation uses full, licensed copies of the window managers, so all resources and syntax are fully supported.

■ Common Error Messages

The ViewStation Diagnostic messages (or Console Window) are a valuable tool in tracing problems and errors. The Diagnostic messages report rcp attempts to read files with both names and locations, and display error messages when they occur. These are some common errors:

1) "RCP failed" - this message typically points to problems with permissions. Check /etc/hosts.equiv and path variables, or perhaps nameservice failures.

2) "Error: Remote shell 'env' command to user@host contains an invalid line." - this message typically points to path or environment variable problems. It may be that the .cshrc file (or its equivalent) has a command that is not understood or has incorrect syntax.

3) "Error executing remote shell env command to user@host." - this message usually indicates that there is a problem with permissions or getting to the correct files.

4) "RCP protocol screwup" - this message can refer to any of the permission, path, or environment variable problems.

5) "Login incorrect" or "Permission denied" - these messages typically indicate a problem with the /etc/hosts or /etc/hosts.equiv files.

■ Suggestions for fixes

The basic task is to isolate the problem. Simplify the configuration as much as possible. Use a single host, single host name, and a single user. Rename the user's .cshrc and use a minimal file in its place. Rename the resource files and use a minimal file in their places. Use the Console window messages to track the loading process and file locations. Reduce your system configuration to the minimal configuration and try to run the ViewStation's window manager before you add the more complicated elements.

2) /etc/hosts.equiv - The ViewStation must have specific equivalency with its primary host (the source of the “rsh” command). Often this /etc/hosts.equiv file is empty or has only a “+” entry; it should have a specific entry for the ViewStation using the rsh command. Check your **man** pages for the correct form and syntax for this file. For a Sun host, the /etc/hosts.equiv file looks like this:

```
# hosts.equiv file
# See /etc/hosts for a list of valid names
hdssun1
hds486
davidfx
+
```

If you are using an .rhosts file, exchanging permissions is a little more complicated and sensitive since "Trusted Access" is implemented in different ways by different systems. The system administrator should be familiar with how this works on his system. Each user may have an .rhosts file in their home directory which contains the names of hosts equivalent to the user. The file itself contains a host name and a user name for each user that is to be granted equivalent status with the home user.

3) Domain Name Service (and/or NIS or Yellow Pages) - The ViewStation must have a name (not just an IP address) on file and the nameservice lookup functions must work correctly from all devices and in all directions.

- Resource File Requirements

The “rsh” command (or its equivalent on other systems, such as “rcmd” on SCO systems and “remsh” on HP systems, etc.) starts a shell. This shell must have correct environment variables, paths, and so on, in place or it cannot locate the files it needs for operation. Different systems and different users may have this information in different places. You must insure that this information is available to the local window manager.

1) For the C shell - the .cshrc file should contain all the pertinent path and environment variables. The .cshrc file is read each time the “rsh” command is given, so this file is the desired location for them. A .login file is read only on startup, and so is not a good location for these variables.

2) For the Bourne shell - the .profile file should contain all the pertinent path and environment variables. The .profile file is read when the “rsh” command is given, so this file is a good location for them.

3) For the Korn shell - the .profile file should contain all the pertinent path and environment variables. You must also put the path and environment variables in some system-wide location where they will be read whenever the “rsh” command is given, perhaps invoking the command from a script which explicitly finds and reads them.

4) Resource files and locations - resource variables for the window managers can have many locations, such as .Xdefaults, /usr/lib/X11/app-defaults, .Xresources,

```
}
Buttons PointerButtonBindings
{
<Btn1Down>          frame|icon                f.raise
<Btn3Down>          frame|icon                f.post_wmenu
<Btn1Down>          root                      f.menuRootMenu
<Btn1Down>          window                    f.raise
Meta<Btn1Down>      window|icon               f.lower
!Meta<Btn2Down>     window|icon               f.resize
!Meta<Btn3Down>     window|icon               f.move
}
#
#  END OF mwm RESOURCE DESCRIPTION FILE
```

2.4.12 Window Manager Remote Starting Checklist

Starting the ViewStation's window managers from a remote host with the "rsh" command allows the local FX window manager access to its resource files on the host and permits "rcp" and "rcmd" access to the network.

These unique and powerful features all depend on the success of the initial "rsh" command and the successful operation of nameservice and other network services. If these processes fail, the local window managers cannot gain the access they require.

This checklist shows a few things you can check if you have problems with this process. It is suggestive of places to look for problems. It is not specific about many items because the names and syntax of files can vary from system to system.

■ Necessary Network Services

1) /etc/hosts file - The ViewStation must have a name entry here on its "rsh" host.

For a Sun host, the /etc/hosts file looks like this:

```
# Sun host database
# If the NIS is running, this file is consulted only when booting
#
128.91.3.12     hdssun
#
128.91.6.5      mikegfx
128.91.6.10     davidfx
```

and so on.

```
#
# key binding descriptions
#
Keys DefaultKeyBindings
{
Shift<Key>Escape    window|icon                f.post_wmenu
Meta<Key>space      window|icon                f.post_wmenu
Meta<Key>Tab        root|icon|window           f.next_key
Meta Shift<Key>Tab  root|icon|window           f.prev_key
Meta<Key>Escape     root|icon|window           f.next_key
Meta Shift<Key>Escaperoot|icon|window          f.prev_key
Meta Shift Ctrl<Key>exclam root|icon|window    f.set_behavior
Meta<Key>F6         window                     f.next_key transient
Meta Shift<Key>F6   window                     f.prev_key transient
<Key>F4             icon                       f.post_wmenu
}
#
# button binding descriptions
#
Buttons DefaultButtonBindings
{
<Btn1Down>          icon|frame                 f.raise
<Btn3Down>          icon                       f.post_wmenu
<Btn1Down>          root                       f.menuRootMenu
}
Buttons ExplicitButtonBindings
{
<Btn1Down>          frame|icon                 f.raise
<Btn3Down>          frame|icon                 f.post_wmenu
<Btn1Down>          root                       f.menu RootMenu
Meta<Btn1Down>      window|icon                f.lower
!Meta<Btn2Down>     window|icon                f.resize
!Meta<Btn3Down>     window|icon                f.move
```

```
#  DEFAULT mwm RESOURCE DESCRIPTION FILE (system.mwmrc and .mwmrc)
#
# menu pane descriptions
#
# Root Menu Description
#
# HDS Users Note: The "New Window" function is not present and the
# "Quit..." mwm termination option has been added.
#
Menu RootMenu
{
"Root Menu"         f.title
"Shuffle Up"        f.circle_up
"Shuffle Down"      f.circle_down
"Refresh"           f.refresh
no-label            f.separator
"Restart..."        f.restart
no-label            f.separator
"Quit..."           f.quit_mwm
}

# Default Window Menu Description
Menu DefaultWindowMenu
{
Restore             _R        Alt<Key>F5       f.normalize
Move                _M        Alt<Key>F7       f.move
Size                _S        Alt<Key>F8       f.resize
Minimize            _n        Alt<Key>F9       f.minimize
Maximize            _x        Alt<Key>F10      f.maximize
Lower               _L        Alt<Key>F3       f.lower
no-label                                       f.separator
Close               _C        Alt<Key>F4       f.kill
}
```

```
osfPaste                        :1004FF04

osfAddMode                      :1004FF31
osfPrimaryPaste                 :1004FF32
osfQuickPaste                   :1004FF33

osfPageUp                       :1004FF41
osfPageDown                     :1004FF42

osfEndLine                      :1004FF57
osfBeginLine                    :1004FF58

osfActivate                     :1004FF44

osfMenuBar                      :1004FF45

osfClear                        :1004FF0B
osfCancel                       :1004FF69
osfHelp                         :1004FF6A
osfMenu                         :1004FF67
osfSelect                       :1004FF60
osfUndo                         :1004FF65

osfLeft                         :1004FF51
osfUp                           :1004FF52
osfRight                        :1004FF53
osfDown                         :1004FF54

!       DEC specific keysyms
!
DRemove                         :1000FF00
```

2.4.11.2 MWM .mwmrc File

This file is the Default mwm Resource Description file, labeled system.mwmrc.

```
#
```

2.4.11 Local MWM Default Files

The HDS ViewStation has some simple default files for running MWM locally. These are listed below. Note that these files must be loaded into the ViewStation memory (with xrdb loading your .Xdefault or .xresource file) before the MWM client starts.

You will probably want to make some additions or changes to these default files. Note that they are for the local MWM. Your own .Xdefaults file will supplement or replace entries in these default files.

2.4.11.1 MWM App-defaults File

This file is the app-defaults file for the Motif Window Manager.

```
!
! app-defaults RESOURCE SPECIFICATIONS FOR Mwm (mwm has precedence
over Mwm)
!
! FONT stuff
Mwm*fontList:                 variable
! Use smaller fixed font for icons
Mwm*icon*fontList:            fixed
!
!  END OF RESOURCE SPECIFICATIONS
!
```

This file is the default XKeysymDB file.

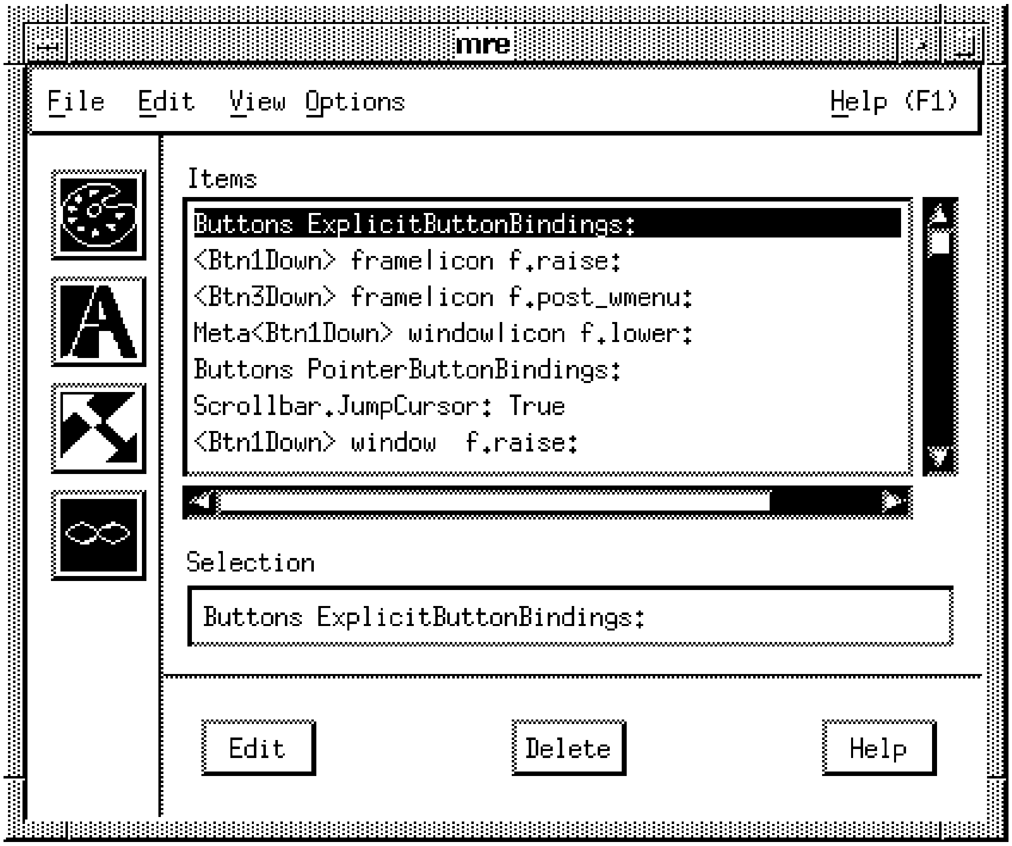
```
!
! HDS default XKeysymDB file.
!
! The 'osf' Open Software Foundation keysyms have been added to
! support the Motif Window Manager.
!
!
!    OSF Keysyms
!
osfBackSpace                  :1004FF08
osfInsert                     :1004FF63
osfDelete                     :1004FFFF
osfCopy                       :1004FF02
osfCut                        :1004FF03
```

for instance, the *Shift-Esc* key sequence brings up the MWM Default Window Menu in the active window regardless of the location of the mouse pointer. Other windows, such as xterms, might have their own key and mouse bindings for particular functions. These might conflict with one another if you are not careful of your definitions.

You should always be sure that these resources, from whatever source, are loaded before the client starts.

2.4.10 Motif Resource Editor

There is an editing utility program provided with the Motif Window Manager which lets you edit your .Xdefaults file. The Motif Resource Editor may be installed on your system, or you may need to have your system administrator set it up and put it in your path.



```
Buttons ExplicitButtonBindings:
<Btn1Down> frame|icon f.raise:
<Btn3Down> frame|icon f.post_wmenu:
Meta<Btn1Down> window|icon f.lower:
Buttons PointerButtonBindings:
Scrollbar.JumpCursor: True
<Btn1Down> window  f.raise:
```

```
Buttons ExplicitButtonBindings:
```

You can start the resource editor by typing:

```
mre &
```

This starts the Motif Resource Editor and reads the .Xdefaults file in your home directory. From that point, you can use the editor to change your Motif resources. The Motif Resource Editor screen is shown below:

- The f.exec entries for ViewStation local clients use the "hdslocal" as an indicator that the process is a local ViewStation process, and also as a ViewStation command to select the local client. Optionally, you could have a script on your host computer called "hdslocal" that would kick off ViewStation clients (using the "rsh" command line) if you were using the host's MWM. This would insure that the menu entries were executed correctly regardless of the MWM's location, host or ViewStation.
- The "Lock" entry on the Root Menu gives a quick and easy way to lock your screen with ViewStation's local screen lock. The password (TV342 in this case) entered on this line is the password to unlock the screen. You could protect the *.mwmrc* file to maintain security for this password.
- Note the use of "rlogin" for the RemoteWindowsMenu. This opens a new window without the need to login (based on correct passwords and permissions on your network). Different color backgrounds are given to distinguish the HDS-term windows (you could also use Title bar names).
- Note the "WindowMenu" entry. This is the menu for the Motif button on the upper left of the title bar. The letters in the second column are accelerators. Do not modify this menu.

When you make changes to this menu, do them one at a time and make sure they work. Check your command lines for f.exec entries manually to make sure they do what you want before you add them to the menu.

2.4.9.2 Starting Clients with Mwm

There are many places to start clients and control their resources. Consider these other locations for configuration selections.

- Setup Mode - The ViewStation has many configuration settings built into Setup Mode, such as the ability to load individual colormaps, or a configuration file of NVM settings, fontpaths, etc.
- Xdm - Xdm is the display manager and is usually the best place to open clients, remote shells, clients you open when you start (like xclock or xload), and xrdb.
- Xrdb - this is a utility to read your .Xdefaults file from some location or from your home directory. Any Mwm* resources in this file would be given to MWM when it started up.
- Individual .rc files - MWM has its own .rc file, called .mwmrc, usually found in your home directory. Comparable files for other clients are app-defaults files (such as /usr/lib/X11/app-defaults/xterm) which are usually located in the */usr/lib/X11/app-defaults* directory.

An example of the kind of problem you might encounter with conflicting settings or definitions of this type is with key assignments (bindings) for different clients. In the X world, the xmodmap utility is used to remap the keyboard, but this is a global setting and will apply to all clients and all windows. This is the correct place for some key preferences, such as multinational keyboards with swapped letters, or setting the mouse buttons for left-handed users. The window manager (like MWM) also sets some key bindings which have an almost global application;

```
Buttons DefaultButtonBindings
{
 <Btn1Down>        frame             f.raise
 <Btn1Down>        icon              f.normalize
 <Btn3Down>        icon              f.post_wmenu

f.menu RootMenu
}

Buttons ExplicitButtonBindings
{
 <Btn1Down>        frame             f.raise
 <Btn1Down>        icon              f.normalize
 <Btn3Down>        frame|icon        f.post_wmenu
 <Btn3Down>        root              f.menu RootMenu
 Shift<Btn1Down>   window|icon       f.lower
 Shift<Btn2Down>   window|icon       f.resize
 Shift<Btn3Down>   window|icon       f.move
 Meta<Btn1Down>    window|icon       f.lower
 Meta<Btn2Down>    window|icon       f.resize
 Meta<Btn3Down>    window|icon       f.move
}

Buttons PointerButtonBindings
{
 <Btn1Down>        frame             f.raise
 <Btn1Down>        icon              f.normalize
 <Btn1Down>        root              f.menu RootMenu
 <Btn1Down>        window            f.raise
 <Btn3Down>        frame|icon        f.post_wmenu
 Shift<Btn1Down>   window|icon       f.lower
 Shift<Btn2Down>   window|icon       f.resize
 Shift<Btn3Down>   window|icon       f.move
 Meta<Btn1Down>    window|icon       f.lower
}
#
# END OF mwm RESOURCE DESCRIPTION FILE
#
```

There are a number of things to notice in this file.

- Look at the file syntax. Note the { and } to separate sections of the file. Note the use of white space and tabs. It’s easiest just to copy an example like this, but be careful as you do it. It is good practice to copy the original working file under another name while you experiment with your changes.
- Look at the way the menus and submenus are named and designated within each other.
- Note that the HDSperx submenu is not listed here. It is added automatically as the first item on this root menu by the merging mechanism when the ViewStation’s MWM starts, if it is started with the remote shell process. This is discussed in more detail below.
- Look at the way “no-label”/”f.separator” entries are used to provide visual organization to the menus.
- Look at the way the Motif functions are labeled, with f.menu designating a menu, f.exec designating command line execution, as well as special Motif functions like f.restart, and so on.

```
“HDS486 Window”           f.exec “rcmd hds486 scoterm”
“HDSsun6 Window”          f.exec “hdslocal hdsterm -bg cyan -e rlogin
hdssun6 &”
“HDSsun5 Window”          f.exec “hdslocal hdsterm -bg lemon -e rlogin
hdssun5 &”
“HDSsun9 Window”          f.exec “hdslocal hdsterm -bg thistle -e rlogin
hdssun9 &”
}

Menu ApplicationsMenu
{
“Applications”            f.title
“HexCalc”                 f.exec “hexcalc &”
“Mail”                    f.exec “xmail -iconic &”
“News”                    f.exec “xrn -iconic &”
“Clock”                   f.exec “xclock -bg thistle &”
“Performance Meter”       f.exec “perfmeter &”
“File Manager”            f.exec “filemgr &”
“Local Xclock”            f.exec “hdslocal hdsxclock -bg cyan &”
“HDS PS View”             f.exec “hdslocal hdsview &”
}

# Window Menu Description

Menu WindowMenu
{
Restore           _R                Alt<Key>F5         f.normalize
Move              _M                Alt<Key>F7         f.move
Size              _S                Alt<Key>F8         f.resize
Minimize          _n                Alt<Key>F9         f.minimize
Maximize          _x                Alt<Key>F10        f.maximize
Lower             _L                Alt<Key>F3         f.lower
no-label                                               f.separator
Close             _C                Alt<Key>F4         f.kill
}
#
# key binding descriptions
#

Keys DefaultKeyBindings
{
 Shift<Key>Escape          window|icon            f.post_wmenu
 Meta<Key>space            window|icon            f.post_wmenu
 Meta<Key>Tab              root|icon|window       f.next_key
 Meta Shift<Key>Tab        root|icon|window       f.prev_key
 Meta<Key>Escape           root|icon|window       f.next_key
 Meta Shift<Key>Escape     root|icon|window       f.prev_key
 Meta<Key>Pause            root|icon|window       f.exec “hdslocal rehash”
 Meta<Key>PgUp             root|icon|window       f.exec “hdslocal xset fp
rehash”
 Meta Shift Ctrl<Key>exclam root|icon|window      f.set_behavior
 Meta<Key>F6               window                 f.next_key transient
 Meta Shift<Key>F6         window                 f.prev_key transient
 <Key>F4                   icon                   f.post_wmenu
}

#
# button binding descriptions
#
```

```
"Restart..."      f.restart

no-label          f.separator

"Quit..."         f.quit_mwm

}
```

Note that the HDSperx entry is added automatically to the top of this menu. It is not included as part of this menu since it is added by the MWM "autoClientMenu" resource .

If you want to customize this menu, you can simply add a few lines with your text editor. This is a convenient place for commands you use frequently, but you should think carefully about how you will use it. Note that there are system considerations involved with your choosing commands that start other processes. Circulating windows up and down is a local function handled by the window manager. Starting a new client on a remote host involves many more considerations, such as authentication privileges, identifying the display, and so on.

2.4.9.1 Sample Customized Mwm Root Menu

This example is an .mwmrc file with a fairly complicated menu, including a number of submenus and button bindings, etc. It is intended to give you samples of possible entries for your own files.

```
#
# Root Menu Description

Menu RootMenu
{
 “Root Menu”              f.title
 “Local XTerms”           f.menu “LocalWindowsMenu”
 “Remote Windows”         f.menu “RemoteWindowsMenu”
 “Applications”           f.menu “ApplicationsMenu”
 no-label                 f.separator
 “Lock”                   f.exec “hdslocal hdssaver -lock -password TV342 -
nographics&”
 no-label                 f.separator
 “Shuffle Up”             f.circle_up
 “Shuffle Down”           f.circle_down
 “Refresh”                f.refresh
 no-label                 f.separator
 “Restart...”             f.restart
 no-label                 f.separator
 “Quit Mwm...”            f.quit_mwm
}
Menu LocalWindowsMenu
{
 “Local XTerms”           f.title
 “HDS Memory”             f.exec “hdslocal hdsmemory &”
 “SlateBlue”              f.exec “xterm -fg white -bg slateblue &”
 “IndianRed”              f.exec “xterm -fg white -bg indianred &”
}

Menu RemoteWindowsMenu
{
“Remote Windows”          f.title
```

1) You must start the ViewStation local window manager from a remote host in order to use these resource specifications and menu entries for the local window manager. This remote host and this user will be the host and the user for starting remote processes, so the appropriate permissions and accounts must exist. If you start the ViewStation's local window manager from the Setup Mode menu, none of these commands will work because only the default resources are used. An error message "No remote host running" will appear in this case. The usage of these files is the same as when a host-based window manager is used. Refer to Section at the end of this chapter for a checklist and a more complete discussion of starting from a remote host.

Further, the permissions for the ViewStation must be the same as for the remote host that started the local window manager. This means that a process valid for the host will also be permitted for the ViewStation. This permission is set in the */etc/hosts.equiv* file, where you must enter the ViewStation's name as an equivalent to the host.

2) The local window manager reads (for purposes of its internal **rcmd** function) the environment files for the user specified, so the environment settings appropriate for the process must be present in the user's .cshrc file. For Bourne and Korn shells, the corresponding files are .profile, though you should double-check that these files are always read when the new shell is spawned. All commands to the remote host are prepended with the ViewStation's display setting so the output is returned to the ViewStation.

If you have problems starting remote processes, look in the Console Window for error messages and use the checklist in Section 2.4.12 at the end of this chapter. You may see messages that report the rhosts command failed because of incorrect permissions, or a file was not found, or that a host is not reachable. Your host machine must be configured with permission for the ViewStation to start remote processes; refer to your */etc/hosts.equiv* file for a ViewStation entry.

2.4.9 Editing the Root Menu Description

You can customize the MWM menus to include things that you do frequently. This is usually done in the .mwmrc file, which is located in your home directory and is read when MWM starts up. A sample .mwmrc file for the HDS local MWM is included at the end of this chapter. The default menu for the RootMenu is shown below:

```
Menu RootMenu

{

"Root Menu"        f.title

"Shuffle Up"       f.circle_up

"Shuffle Down"     f.circle_down

"Refresh"          f.refresh

no-label           f.separator
```

submenu lists all the HDSperx clients you have loaded and you can start them simply by clicking on them.

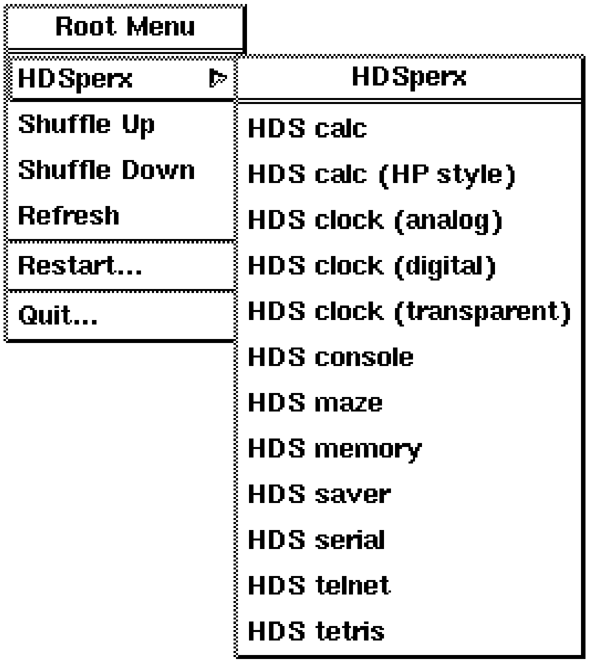
The details of making changes to this menu and setting it up manually are given later in this Section. Refer also to Section 2.1 on using the HDSperx clients.

2.4.8 Starting Remote Processes from the Local MWM

The ViewStation runs mwm locally on the server. Either window manager may be started from a remote host using the **rsh** (BSD Unix) or **rcmd** (System V Unix) command. Starting processes on other hosts is done in the same way as with a remote window manager, that is, using using remote shell commands specified in the appropriate configuration file. The sample configuration file for the window manager menus below shows the process. This file resides in the user's home directory.

2.4.8.1 MWM Menu with HDSperx

Mwm provides an f.exec function to permit the user to start an X client or other process from a menu selection as specified in a configuration file, such as .mwmrc. With the HDSperx clients loaded on your menu, the root menu looks like this:



Details of operation for MWM and this file are available with the OSF MWM specification, including matters of labeling and syntax. There are two things to note:

2.4.6.1 MWM Resource Files

Mwm behavior is controlled by resources. These are usually found in one or more of these files:These are files found in your home directory that MWM reads when it starts up. These files must be present in the ViewStation memory before the local MWM is started or they will not be used. If you start the window manager from a remote host (using the **rsh** command), the ViewStation's window manager will try to read these files with the **rcp** command.

These resources are read and loaded into the server memory with the **xrdb** utility program. You can check the current contents of your resources with the **xrdb -query** command. Refer to Section 3.4 for more details on xrdb.

- /usr/lib/X11/app-defaults/Mwm
- $HOME/Mwm
- $HOME/.mwmrc
- $HOME/.Xdefaults
- $HOME/.Xdefaults-<*hostname*>

You can place your resource entries in any of these locations, but you insure that they are read correctly when you start Mwm.

Some sample resources might be:

```
! Colors

Mwm*background: LightBlue

Mwm*foreground: Black

Mwm*activeBackground: Blue

Mwm*icon*foreground: DarkSlateBlue

! Fonts

Mwm*fontList: 8x13

Mwm*menu*fontList: 9x15

! Focus Policy

Mwm*keyboardFocusPolicy: pointer

Mwm*focusAutoRaise: true

! Icons

Mwm*useIconBox: true

Mwm*iconBoxGeometry: 2x4
```

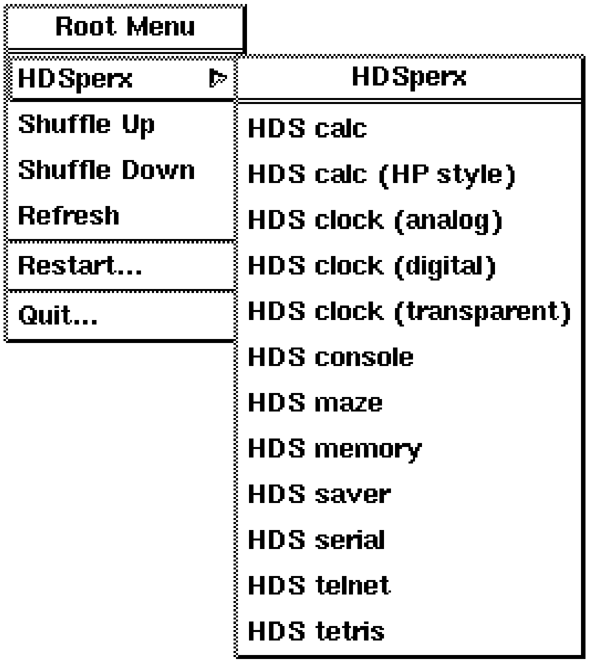
Refer to the Mwm manuals for complete listings of resources and their applicability.

2.4.7 Starting Local Processes from the Local MWM

When you use the ViewStation's local MWM window manager, a submenu for the ViewStation's HDSperx clients is automatically added to your root menu. This

Quit	Exits from MWM

This menu is an easy way to use MWM. It can also be customized to include other options; see the discussion below about changing the menu items. This menu, with the HDSperx submenu, is shown below:



2.4.5 Exiting MWM

You can exit MWM by bringing up the MWM root menu and clicking on Quit. Another window appears asking you to confirm this choice.

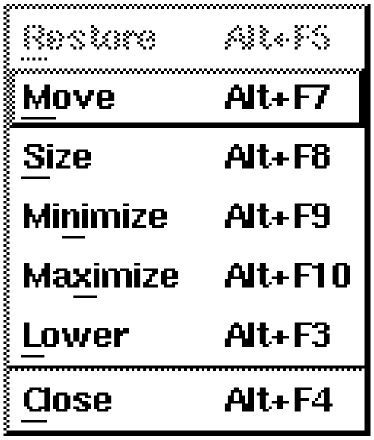
You can exit the window manager at any time, but generally it is a good idea to do some housekeeping beforehand. Close any windows not needed, especially submenus brought up by your clients or windows you have iconified. When you kill the window manager, all windows are restored from icons, so your screen may suddenly become very cluttered and the window you want may be obscured (and since you have no window manager running, you can’t move it).

In general, you should always exit your clients in a orderly way, that is, you should specifically exit a process running in a window rather than just killling the window. An orderly exit insures that files are correctly saved and processes correctly terminated on your host. Just killing the window with the window manager may be too sudden and uncontrolled to protect your work.

2.4.6 Customizing the Motif Window Manager

You can choose the way MWM appears on your display by selecting different resources in your .Xdefaults and .mwmrc files.

from the keyboard. All of the menu items have keyboard key sequences, which are shown on the menu. The menu contains the selections shown below



These functions are the same as the other window manager operations just described. You can select them by dragging the mouse pointer to them and releasing the depressed mouse button. If a menu item is not available, it appears with a gray display; for instance a normal window shows the Restore command as grayed because it is already in its normal condition. The menu items are:

Restore - Restores a window from an icon or from a maximized window.

Move - Change the location of the window.

Size - Change the size of the window.

Minimize - Iconify the window.

Maximize - Enlarge the window to the maximum screen size.

Lower - Rotate the window down to the bottom of the window stack.

Close - Kill the process in the window and make it disappear.

Some windows on your screen may not have all these options on their menu. For instance, a client's submenus may not be able to be iconified or resized, so the Minimize and Maximize functions are absent.

2.4.4 Using the MWM Root Menu

There is another MWM menu which is brought up by pressing the mouse button when the pointer is in the root window. The root menu can be a customized menu you specify in your .mwmrc file, or it can be the ViewStation's default menu, shown below. This default menu has six commands:

HDSperx	Brings up the HDSperx clients menu
Shuffle Up	Moves the windows up the stack
Shuffle Down	Moves the windows down the stack
Refresh	Refreshes the screen
Restart	Restarts MWM

Resize side/Resize corner - These are the handles to resize the window.

Maximize button - This button enlarges the window to fill the entire screen.

Minimize button - This button reduces the window to an icon.

Menu button - This button brings up the MWM menu.

2.4.3.6 Moving a Window

You can move windows around on the screen to make your work more efficient. This is a primary function of the window manager. To move a window, place the mouse pointer in the title bar and press the left button down, then drag the mouse pointer to the new location of the window. An outline of the window will follow your movements. You will notice a little box in the center of the screen which gives the location of the window's upper left corner in pixels. When the position of the window is correct, simply release the mouse button and the window appears at its new location.

2.4.3.7 Resizing a Window

You can change a window's size in several ways. Imagine that the window has elastic sides. By grabbing an edge or a corner (point the mouse on an edge or a corner and press the left button and hold it down), you can stretch the window to a larger size or let it contract to a smaller size. Dragging the left side stretches the window to the left (with the right side remaining fixed); dragging the lower left corner stretches the left and bottom sides (with the upper right corner remaining fixed). Try resizing a window - it's easier to learn by doing than by reading.

2.4.3.8 Iconifying and Deiconifying a Window

You can gain space on your screen by changing a window into an icon. The window will continue to function if it has a process running, but it is temporarily hidden from view. To iconify a window, simply move the pointer to the Minimize button and click on it. The window changes to an icon and moves to a row of icons, usually at the bottom of the root window. You can move the icons (they are windows), or define a different place for the icon row.

The icon itself has an image and a label so you can distinguish them from each other. To deiconify an icon, move the mouse pointer to it and double click. The window will be restored to its original size and location on the screen.

2.4.3.9 Maximizing a Window

If you want to work in one window for a while, you can enlarge it to take up the whole screen by maximizing it. Move the mouse pointer to the Maximize button and click on it. To restore the window to its original size, click the Maximize button again, or use the Window Menu, which is described below.

2.4.3.10 MWM Window Menu

Pressing and holding the Menu button brings up the MWM menu. It disappears when you release the button. You can also bring up the menu by typing *Shift-Esc*

2.4.3.4 Selecting a Window with MWM

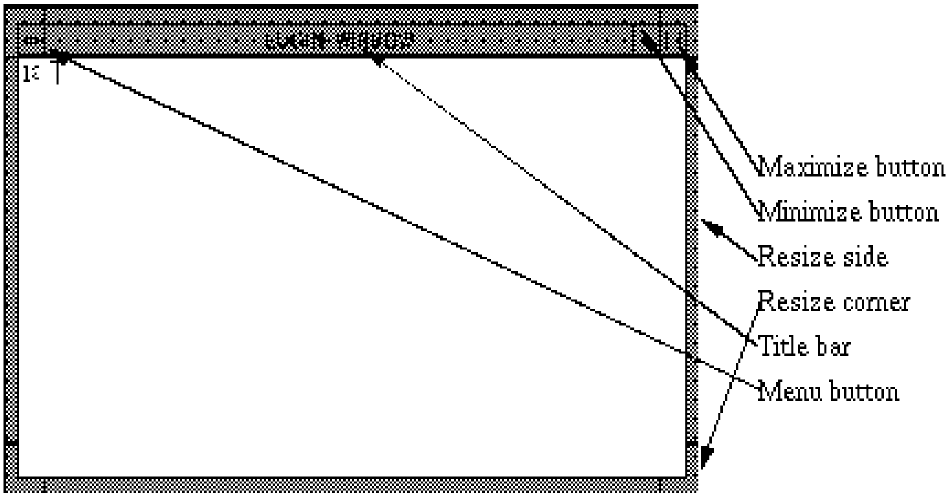
When you have many windows on the screen, you move from one to another by moving the mouse pointer and clicking (with the left button) on the window to select it. You will notice that the border of the window changes color or shade as an indication that it is the currently selected window. This is called getting window focus; it means that data from the keyboard or mouse enters that window. Giving a window focus also means that it moves to the top of the stack, so if any other windows were covering it, they are moved down.

The Motif Window Manager can give focus to a window in two ways - either by placing the pointer in the window and clicking on it (this is the default choice), or by simply moving the pointer into the window. There is a configuration selection (called a resource) that selects this mode of operation. Setting this resource will be discussed later in this chapter. If you point and click to give a window focus, that window retains focus regardless of where the mouse pointer is located. Many people prefer this type of operation because they can work in one window and ignore the mouse position. Others prefer to be able to get window focus by moving the mouse only, without the need to click when they are in the window.

This is a significant difference in the way a user controls the screen. The operation of the window manager controls this function and many other significant operations depending on its configuration.

2.4.3.5 Window Decoration

The border around the window, called its decoration, has areas marked that permit easy manipulation of the window. These are labeled below, along with brief descriptions of their operations.



Title Bar - This is the block showing the title you (or the client) gave to this window.

When the window manager is started remotely, the ViewStation tries to read the applicable resource files, such as .Xdefaults, or /usr/lib/X11/app-defaults/ Mwm, or .mwmrc with the **rcp** (remote copy) command. You can monitor this process from the ViewStation's Console Window. The remote copy command must succeed or the window manager resources can't be used. If one of those files isn't located, the ViewStation will timeout and search for the next one; using valid entries in these files, or using them with null entries, will reduce the timeout pauses. You can follow this process, and locate all the file names, by monitoring the Console shell messages.This is discussed in more detail in Section 2.4.12 at the end of this section.

2.4.3 MWM Operations

The Motif Window Manager is a graphical user interface that lets you handle your windows by pointing with the mouse and clicking to select various window operations. This simple process applies to all of your windows, regardless of the host running them or the contents of the windows themselves.

2.4.3.1 Pointer Shapes

The mouse cursor, called the Pointer, changes shape to give you an indication of your position on the screen and the type of operation currently selected. For instance, the cursor has the standard X shape when it is located in the root window. The pointer changes to an I shape when you are inside a terminal window, indicating that you are ready to enter text. When you are resizing a window the pointer indicates by its shape the current resize process, such as an upper right corner arrow or a vertical down arrow. The hourglass pointer indicates that a process is working and you should wait for its completion.

2.4.3.2 Mouse Buttons

You can use either the standard three-button mouse or an optional two-button mouse or optional trackball. The three mouse buttons behave differently; but their operation is usually just a simple click (a single press and release), a double click (two clicks in rapid succession), or click and drag (pressing a button and holding it down while you move the mouse pointer).

Different operations are selected by the left, middle, and right mouse buttons. If you have a two-button mouse, the middle button operations are selected by pressing both buttons at the same time. (The actual configuration of the mouse buttons, such as swapping left and right buttons, can be modified by the xmodmap utility, as described in Section 1.8)

2.4.3.3 Mouse Buttons and Modifier Keys

The operation of mouse buttons is affected by any modifier keys, such as Shift, Ctrl, and Alt, you have set. For instance, selecting the header bar of the window by pressing the left mouse button to move it won't work if you have the Ctrl or Alt key depressed. This is especially frustrating (and often overlooked) with other modifier keys, such as Caps_Lock and Num_Lock. If you find unexpected problems with your window manager operation, check the modifier keys.

Note that the window manager icon appears only if the HDSware host and path are correctly set and the *./hds-fx/config/clients.config* file is loaded. These processes are discussed in Section 1.5.2. You can also use this file to pre-load the Motif library, which will allow clients to start more quickly. This is discussed in Section 2.1.4.

If you have customized MWM with special resource settings, these resources must be read into the ViewStation memory before the local MWM is started. This is done with the **xrdb** utility program.

2.4.2 Starting Local MWM Remotely

Normally, you just click on the Motif Window Manager option on the Setup Menu to start it. Many times it is more convenient to start the local MWM from an .Xsession file or from a remote host login file. Starting the window manager from a remote host gives you additional power and flexibility. Starting the MWM from a remote host copies your customized window manager files, such as .mwmrc, to the local window manager's operation.

The local ViewStation Motif Window Manager can be started from a remote host with the **rsh** (remote shell command) as follows:

```
rsh -n <hostname> <mwm > [arguments] '&'
```

(your operating system may use different syntax; SCO uses **rcmd** and HP uses **remsh**, for example)

The remote shell command must use the ViewStation's hostname as it is entered in the */etc/hosts* file (the **rsh** command does not accept an IP address). You should use the -n option (null input) so the process exits properly when you close the window manager (check the syntax on your remote shell command). The next entry is the window manager you want to start, either mwm or olwm. Only one window manager can be running at one time. If you try to start a window manager when another is running, the remote shell command will fail but there will be no error message displayed except in the ViewStation's Console Window.The last argument "'&'" allows the rsh process to exit cleanly when the window manager starts.

Command line arguments for the window manager can be entered on this line. For instance, entering the -display argument would permit you to run the ViewStation's window manager on another display device. The remote shell also accepts command signals, such as Ctrl-C, which could be used to stop the window manager after it has started. If you use resource names on the command line you must take care to quote and escape them properly; for example, the operating system would misinterpret the * (asterisk) in the resource name as a wildcard character.

The **rsh** command line can be entered from any location that will start a remote shell, such as your .xsession file or .login file. Configuration options for the window managers, such as applicable resources from .Xresources, or from .mwmrc, etc. should be read before the remote shell command is given.

MWM - Motif Window Manager

This chapter describes the operation of the Motif Window Manager. There are also examples of customizing the window manager.

Section 2.0 - HDS ViewStation Local Clients

2.4 Motif Window Manager

The HDS ViewStation has licensed copies of the Motif Window Manager from the Open Software Foundation to run locally with its server code. With ViewStation server version 3.0, the Motif window manager version is 1.2.3. Since the Motif Window Manager is an authorized version, not a copy or a facsimile, you can use all the standard commands, resources, and reference materials that apply to the OSF Motif Window Manager. The manual pages for the MWM window manager are included on the HDSware tape in the */hds-fx/man* directory.

This section gives a brief overview of the Motif Window Manager. There are some simple exercises to help you learn what symbols perform which operations, and so on, but you should refer to individual Motif reference materials or man pages for full details.

This section also contains sample files for Motif resources and some simple suggestions for customizing the Motif Window Manager for your system environment and your personal preferences. Again, this material is only suggestive and you should refer to the Motif reference materials for details.

2.4.1 Starting Motif Window Manager

You can start the Motif Window Manager by selecting the Motif Window Manager button from the Main Menu of Setup Mode and clicking on it. You will notice that the window manager puts a border around all the windows currently on your screen. Any new windows you open will automatically be controlled by the Motif Window Manager.