Human Designed Systems ViewStation Series: System Administrator's Guide

DN 13fx3.2.1 - 9601 - 14

c 1996 by HDS Network Systems, Inc.
400 Feheley Drive
King of Prussia, PA 19406
(610) 277-8300

copyright 1996 by HDS Network Systems, Inc. This manual is copyrighted by HDS Network Systems, Inc. All rights are reserved. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from HDS Network Systems, Inc.

HDS, Human Designed Systems, ViewStation, ViewStation FX, ViewStation Ultra are trademarks of HDS Network Systems, Inc.

Other trademarks used in this manual are the properties of their owners.

The X Window System: ia a trademark of the X Consortium, Inc..

Printed in the U.S.A.

Preface

This *HDS ViewStation System Administrator's Guide* is intended for advanced users, generally system administrators or individuals who have need to configure or customize ViewStations on a network.

This manual is intended as a reference point for ViewStation operations. There are many levels of functions in a distributed network environment, including the ViewStation and its server software or clients, the network itself, the host's operating system and utilities, the X protocol, and the application software you're running. Problems or configurations can occur at any level. This manual only addresses ViewStation issues and briefly describes some network and host system issues that are appropriate. More detailed information is available in other reference materials or manual pages for specific functions.

Additional material is available on the HDS Internet Server at:

http://www.hds.com

A hypertext version of this manual is available on the HDS Internet site.

This manual is shipped with the HDSware distribution. It is also available to users as a set of postscript files (readable with the HDSview local client) on the HDSware distribution media. These can be viewed with the ViewStation's HDSperx client, HDS view.

The *HDS ViewStation Owner's Manual* is for general users and focuses on setting up the ViewStation for the first time. It is shipped with each ViewStation.

This manual is arranged in general sections as follows:

- Section 1.0 - ViewStation features and operations
- Section 2.0 - ViewStation local client programs

- Section 3.0 - Network functions and operations
- Section 4.0 - Host system operations and installations
- Section 5.0 - ViewStation Developer's Kit
- Section 6.0 - HDS Warranty and Services
- Index

Section numbers are used throughout for organization and ease of reference.

Table of Contents

Front Matter and Copyrights iii
Table of Contents v
Preface xxiii

Section 1.0 - HDS ViewStation 1

1.1 HDS ViewStation Specifications 1

1.1.1 HDS ViewStation Overview 1
1.1.2 Hardware Specifications 1
1.1.2.1 ViewStation Base Unit 1
1.1.2.2 ViewStation Circuit Board 1
1.1.2.3 ViewStation Memory 1
1.1.2.4 ViewStation Bootprom 2
1.1.2.5 ViewStation Video Memory 2
1.1.2.6 ViewStation External Connections 2
1.1.2.7 ViewStation Optional Communications Board 2
1.1.2.8 ViewStation FlashPROM Board 2
1.1.2.9 ViewStation PC Card Board 3
1.1.2.10 ViewStation PC Card Devices 3
1.1.2.11 ViewStation Floppy Disk Drive 3
1.1.3 ViewStation Power Supply 3
1.1.4 ViewStation Keyboards 3
1.1.5 ViewStation Mouse and Pointing Devices 3
1.1.6 ViewStation Display Monitors 4

1.1.6.1 Selecting the Monitor Type 5
1.1.6.2 Changing ViewStation Monitors 6
1.1.7 ViewStation Physical Characteristics 7
1.1.7.1 ViewStation Base Unit 7
1.1.7.2 ViewStation Keyboards 7
1.1.7.3 ViewStation Pointing Devices 8
1.1.7.4 ViewStation Monitors 8
1.1.8 ViewStation Environmental Requirements: 8
1.1.8.1 Operation 8
1.1.8.2 Storage 8
1.1.9 ViewStation Controls and Indicators 8
1.1.10 ViewStation Connections 8
1.1.10.1 Mouse Connector 8
1.1.10.2 Keyboard Connector 9
1.1.10.3 Serial Port - RJ11 Connector 9
1.1.10.4 Parallel Port - DB25 Connector 10
1.1.10.5 Video Connector - DB15 high-density (VGA-type) 11
1.1.10.6 Twisted-Pair Ethernet - RJ45 Connector 11
1.1.10.7 ThinWire Ethernet - BNC Connector 12
1.1.10.8 ThickWire Ethernet - DB15 Connector 12
1.1.10.9 Serial Port - DB-25 Connector 13
1.1.10.10 Sun Keyboard Connector 13
1.1.10.11 Audio Ports - Input/Output Connections 13
1.1.10.12 PC CardPort 13
1.1.11 ViewStation Software 14
1.1.11.1 ViewStation Server Components 14
1.1.11.2 Setup Mode 14
1.1.11.3 Setup Mode Command Buttons 16
1.1.11.4 Network Services 16
1.1.11.5 Terminal Emulator 17
1.1.11.6 Font Loading 17
1.1.11.7 Server-Resident Fonts 17
1.1.12 HDS ViewStation Number Encoding 19
1.1.12.1 HDS ViewStation Serial Number 19
1.1.12.2 HDS ViewStation Part Number 19
1.2 HDS ViewStation Internal Architecture 27
1.2.1 ViewStation Console Shell Window 27
1.2.1.1 Access to the Console Window 28
1.2.1.2 Console Window as a Local Display 29
1.2.1.3 Console Window from a Remote Host 29
1.2.1.4 Diagnostic Messages from a Remote Terminal 30

1.2.1.5 Console Window Security 30
1.2.2 Console Window Commands 31
1.2.2.1 Summary of Console commands: 32
1.2.2.2 Summary of privileged commands 34
1.2.2.3 Privileged Commands in a Shell Script 35
1.2.3 Console Window Messages 35
1.2.3.1 ViewStation Error Message Reporting 35
1.2.4 Printing from the Console Window 37
1.2.5 ViewStation Internal File System Structure 38
1.2.5.1 FXUtilities Abbreviation 38
1.2.5.2 ViewStation Mount Points 38
1.2.5.3 Local Mount Directory 39
1.2.5.4 Remote Mount Directory 39
1.2.5.5 Mount Directory Example 40
1.2.6 Access to ViewStation Ports 41
1.2.6.1 The opendev Command 41
1.2.6.2 Remote Access to ViewStation Ports 41
1.2.6.3 Usepty Script 42
1.2.6.4 Implementing "opendev" in a C Program 42
1.2.7 ViewStation Flash File System 43
1.2.7.1 Operation with the flashPROM Board 43
1.2.7.2 Operation with PCMCIA Devices 43
1.2.8 ViewStation Startup Files 43
1.2.8.1 Server-reset Command File 44
1.2.8.2 Startup Command File 44
1.3 HDSware 47
1.3.1 HDSware Software Formats 47
1.3.1.1 HDSware on an Internal Hard Disk Drive 48
1.3.1.2 Premium Software Service and Software Upgrades 48
1.3.2 HDSware CD-ROM Format 48
1.3.2.1 HDSware CD-ROM File Naming 48
1.3.2.2 Mounting the CD-ROM 48
1.3.2.3 Loading files from the CD-ROM 49
1.3.2.4 HDSware Installation Script 49
1.3.2.5 HDSware CD-ROM Contents 50
1.3.3 HDSware Tape Contents 52
1.3.4 Automatic ViewStation Configuration 55
1.3.5 HDSware Upgrades 55
1.3.6 Loading the HDSware Tape Manually 56
1.3.6.1 Cartridge tape handling 56

1.3.6.2 Diskette handling (5.25" or 3.5"): 57
1.3.7 HDSware Utility Programs 58
1.3.7.1 Compiling the Utility Programs from Source Code 58
1.3.7.2 bdfto960 Font Utility Program 58
1.3.7.3 hdsmkdir Font Utility Program 59
1.3.7.4 hdsfc960 Font Conversion Utility 59
1.3.7.5 getbdf Font Conversion Utility 61
1.3.7.6 hdscrypt Encryption Utility Program 61
1.3.7.7 flashfilesys Utility Program 61
1.3.8 HDSware Contributed Programs 61
1.3.8.1 bootp 62
1.3.8.2 hdsrunwm Launcher Program 62
1.3.8.3 xscope Diagnostic Utility 62
1.3.8.4 xmkmf Utility Program 62
1.3.8.5 snftobdf Font Conversion Program 62
1.3.8.6 usepty Utility Program 62
1.3.8.7 xrsh Utility Program 62
1.4 Network Connections 63
1.4.1 Communications Menu 64
1.4.2 Ethernet Connections 65
1.4.2.1 Ethernet Communications Menu 66
1.4.2.2 Physical Connection to the Ethernet Network 67
1.4.2.3 Twisted-pair Ethernet with Link Integrity and SQE Settings 68
1.4.2.4 Selecting the Ethernet Physical Type 68
1.4.2.5 Logical Connection to the Ethernet 69
1.4.2.6 Network Addressing 70
1.4.2.7 IEEE 802.3 and "Ethernet" Networks 71
1.4.2.8 Use of Network Trailers 71
1.4.3 Token Ring Interface 71
1.4.3.1 Token Ring Communications Menu 72
1.4.3.2 Token Ring Specifications 73
1.4.3.3 ViewStation Token Ring Controls 73
1.4.4 ViewStation Serial Connection 74
1.4.4.1 ViewStation Serial Port Physical Connection 74
1.4.4.2 Serial Port Troubleshooting 74
1.4.4.3 ViewStation Serial Port Menu 74
1.4.4.4 PPP/SLIP Options Menu 76
1.4.4.5 PPP Protocol for Serial Communications 77
1.4.4.6 Direct Serial Connections 79
1.4.4.7 Serial Connection Example 80
1.4.4.8 PPP Connection Example 81

1.4.4.9 SLIP Protocol or CSLIP Protocol for Serial Communications 81
1.4.5 ViewStation IP Addresses and Route Table 82
1.4.5.1 PPP Configuration and Network Routes 82
1.4.5.2 Typical PPP Configurations 82
1.4.5.3 ViewStation Local Route Command 84
1.4.5.4 ViewStation Local Ping Command 84
1.5 Setup Mode 85
1.5.1 Entering Setup Mode 85
1.5.1.1 Setup Mode with Hidden Icon 85
1.5.1.2 Remote Setup Mode Operation 86
1.5.2 Setup Mode Operation 86
1.5.2.1 Quick Starting with Setup Mode 87
1.5.2.2 Setup Mode Editing 88
1.5.2.3 Setup Mode Editing Keys 88
1.5.2.4 File Names and Paths in Setup Mode 88
1.5.2.5 Setup Mode Password Protection 89
1.5.3 Remote Configuration of the ViewStation 89
1.5.4 Setup Mode Main Menu 90
1.5.4.1 Setup Mode Heading Bar 90
1.5.4.2 Setup Mode Selection Buttons 90
1.5.4.3 Setup Mode Function Buttons 91
1.5.5 Starting ViewStation Clients 93
1.5.5.1 Telnet Terminal Window Selection 93
1.5.5.2 Serial Terminal Window Selection 93
1.5.5.3 Console Window Selection 95
1.5.5.4 Available Memory Selection 96
1.5.5.5 Low Memory Warning Client 97
1.5.5.6 Motif Window Manager 97
1.5.5.7 OPEN LOOK Virtual Window Manager 97
1.5.6 Setup Options Selection 97
1.5.6.1 Communications Menu 98
1.5.6.2 Ethernet Communications Menu 99
1.5.6.3 Configure Serial 1 (2) Port Menu 99
1.5.6.4 Token Ring Communications Menu 99
1.5.6.5 XDMCP Options Selection 100
1.5.6.6 Remote Hosts Menu 102
1.5.6.7 Domain Name Servers Selection 105
1.5.6.8 User Preferences Selection 106
1.5.6.9 Startup Options Selection 114
1.5.6.10 Restart and Reset Menu Selection 116
1.5.6.11 Reset NVM to Factory Defaults Selection 117

1.5.6.12 Restore Settings from NVM 118
1.6 Starting the ViewStation 119
1.6.1 Starting from Local Storage 119
1.6.1.1 Starting from a ViewStation Hard Disk 119
1.6.2 Starting from Monitor Mode 120
1.6.2.1 Monitor Mode Key Mappings 120
1.6.2.2 Monitor Mode Screen 120
1.6.2.3 Monitor Mode Commands 121
1.6.2.4 Monitor Mode Sample Entries 122
1.6.2.5 Booting ViewStations with 4 MB Memory 123
1.6.2.6 Booting the ViewStation from a VMS Host 124
1.6.3 Starting with BOOTP 125
1.6.3.1 BOOTP Command Fields 125
1.6.3.2 Problems with BOOTP Starting 126
1.6.3.3 Sample bootptab File 127
1.6.4 Startup Considerations 128
1.6.4.1 Local storage Benefits 128
1.6.4.2 BOOTP Benefits 128
1.6.4.3 Monitor Mode Benefits 129
1.6.5 Determining the IP Address 129
1.6.5.1 Setup Mode IP Address Selection 129
1.6.5.2 IP Addressing with BOOTP 129
1.6.5.3 IP Addressing with RARP 129
1.6.5.4 Sun NIS IP Addressing 130
1.6.5.5 RARP Protocol Example 130
1.6.6 Network-based ViewStation Operation 131
1.7 Remote Configuration 133
1.7.1 Determining the IP Address 133
1.7.2 Loading the ViewStation's Server File 133
1.7.3 Loading the ViewStation's Configuration File 134
1.7.3.1 FXUtilities and HDSware Abbreviation 134
1.7.4 Configuration File Syntax 134
1.7.4.1 Generating an Individual Configuration File 143
1.7.5 Using Include Statements in a Configuration File 144
1.7.6 ViewStation Hard Disk Configuration File 146
1.7.6.1 Configuration File config.fx.dsk 146
1.7.6.2 Configuration File clients.config.dsk 147
1.7.7 Configuration File for 4 MB ViewStations 148
1.7.8 Remote Configuration Security Considerations 148

1.7.9 Remote Management of ViewStations 148
1.7.9.1 Changing ViewStation Settings 148
1.7.9.2 Changing the ViewStation Password 148
1.7.9.3 Changing Server Code Versions 148
1.7.9.4 Problem Diagnosis from a Remote Machine 148
1.8 Keyboards and their Operations 149
1.8.1 HDS ViewStation Keyboards 149
1.8.1.1 Keyboard Connections 150
1.8.2 Keystroke Functions on the ViewStation 150
1.8.2.1 Keyboard Cursor Mode 150
1.8.2.2 Setup Mode Keystrokes 150
1.8.2.3 Monitor Mode Keystrokes 151
1.8.2.4 HDSterm Environment Variable Keystrokes 151
1.8.3 Keyboard Operations 151
1.8.3.1 Keycode Modification Tables 152
1.8.3.2 Key Mappings 152
1.8.3.3 Modifier Keymappings 153
1.8.3.4 ViewStation Pointer Mappings 153
1.8.4 Xev Key Monitoring 154
1.8.5 Key Handling in a Client/Server Environment 154
1.8.5.1 Keymap Changes with Xmodmap 155
1.8.5.2 Keymap Changes with Event Translations 157
1.8.5.3 HDSterm Event Translations 159
1.8.5.4 HDSterm Keymap Groups 159
1.8.5.5 Customizing HDSterm Keymaps 160
1.8.5.6 Customizing Example 160
1.8.5.7 Special Motif Bindings for HDSterm Keys 162
1.9 Font Handling 163
1.9.1 Fonts in the X Window System 163
1.9.2 Automatic Fontpath Configuration 164
1.9.3 Font Server Operations 164
1.9.3.1 Putting the Font Server in the Font Path 164
1.9.3.2 Font Server Benefits 164
1.9.3.3 Using Alternate Font Servers 165
1.9.4 HDS ViewStation Font File Formats 165
1.9.4.1 ViewStation Compressed Font Files 165
1.9.4.2 Making Special .snf Font Files 165
1.9.4.3 Font File Formats - .bdf and .snf 166
1.9.5 Using HDS Font Utilities 166
1.9.5.1 HDS Font Utilities 166

1.9.5.2 bdfto960 Font Utility Program 167
1.9.5.3 hdsmkdir Font Utility Program 168
1.9.5.4 getbdf Font Utility Program 168
1.9.5.5 hdsfc960 Font Utility Program 168
1.9.5.6 Handling .bdf Fonts and Fonts in other Formats 168
1.9.6 Loading Fonts to the ViewStation 170
1.9.6.1 Setting the Fontpath 170
1.9.6.2 Verifying the Fontpath Setting 171
1.9.6.3 Making a fonts.dir File 172
1.9.6.4 Making a fonts.alias File 173
1.10 PC CardDevices 175
1.10.1 ViewStation PC Card Adapter Card 175
1.10.2 ViewStation FlashPROM 176
1.10.2.1 Loading New Server Code into flashPROM 176
1.10.3 ViewStation ROM cards 177
1.10.4 Hard Disk Drive Storage Devices 177
1.10.4.1 Adding a PC Card Hard Disk Drive 177
1.10.4.2 Using the Hard Disk Drive 178
1.10.4.3 Using the fdisk Command 178
1.10.4.4 Using the flash Command 179
1.10.4.5 The flash boot Command 179
1.10.4.6 The flash server Command 179
1.10.4.7 The flash files Command 179
1.10.5 Token Ring Interface Card 181
1.10.5.1 Token Ring Specifications 181
1.10.5.2 ViewStation Token Ring Controls 182
1.10.6 Multimedia Devices 183
1.10.6.1 Configuring the ViewStation for Multimedia 183
1.10.6.2 ViewStation Multimedia Audio Specifications 183
1.10.6.3 ViewStation Multimedia Audio Capabilities 184
1.10.7 Other PC Card Devices 184
1.11 HDS ViewStation Security Considerations 185
1.11.1 ViewStation Passwords 185
1.11.1.1 Changing the ViewStation Password 185
1.11.1.2 Setup Mode Password Protection 186
1.11.1.3 Console Window Password Protection 186
1.11.1.4 HDSperx Password Protection 186
1.11.1.5 Password Security 187
1.11.1.6 Passwords in Bootp 188
1.11.1.7 Password Encryption with hdscrypt 188

1.11.2 ViewStation rhosts Security 189
1.11.2.1 Setting Up an rhosts File on the ViewStation 189
1.11.2.2 ViewStation Remote Access Reporting 189
1.11.3 Permission for Remote Shell Commands 190
1.11.3.1 Using the /etc/hosts.equiv File 190
1.11.3.2 Using the .rhosts File on the Host 190
1.11.4 Host Access Control Security 191
1.11.4.1 Host Access Control from Setup Mode 191
1.11.4.2 Using Magic Cookie Security 191
1.11.4.3 Starting Magic Cookie Security in Xdm 191
1.11.4.4 Magic Cookie Security on the ViewStation 192
1.11.4.5 Problems with Magic Cookie Authentication 192
1.11.4.6 Using the ViewStation's xhost Client 193
1.11.5 CMW Security 193
1.11.5.1 CMW Security Features 193
1.12 ViewStation Floppy Drive Feature 195
1.12.1 Hardware configuration 195
1.12.2 Using the Floppy Drive with a Printer 196
1.12.3 Software configuration 196
1.12.4 Changing Floppy Disks 196
1.12.5 Floppy Disk Drive Access 196
1.12.6 Local Floppy Filesystem Access 197
1.12.7 Floppy Drive RCP (Remote Copy) Access 197
1.12.8 Floppy Drive NFS (Network File Service) Access 197
1.12.9 Floppy Drive File Ownership 198
1.12.10 Floppy Drive Formatting disks 198
1.12.11 Using the ViewStation's Floppy Disk with SoftWindows from Insignia Solutions 199
1.12.12 Using the ViewStation's Floppy Disk from WABI 2.0 200
Section 2.0 - HDS ViewStation Local Clients 203
2.1 HDS ViewStation Client Operations 203
2.1.1 Using the HDSperx Clients 203
2.1.2 HDSperx Clients 203
2.1.2.1 HDSperx Clients Menu 204
2.1.2.2 HDSperx Client List 205
2.1.3 HDSware Installation of HDSperx 205
2.1.3.1 Using the HDSperx Clients 205
2.1.4 Customizing HDSperx Clients 206

2.1.4.1 Format of Client Configuration Files 206
2.1.4.2 A clients.config Sample File 206
2.1.4.3 Pre-loading Motif Libraries 207
2.1.5 HDSperx Client Attributes 207
2.1.6 X11 Server Extensions 211
2.1.6.1 Downloadable X11 Server Extensions 211
2.1.6.2 Loading X11 Server Extensions 212
2.1.6.3 XTrap Extension 212
2.1.6.4 Multi-Buffering Extension 212
2.1.6.5 Shared Memory Extension 212
2.1.6.6 PEX-3D Extension 212
2.2 HDSterm Terminal Emulation Client 213
2.2.1 Using the HDSterm Client 213
2.2.1.1 HDSterm Window 214
2.2.1.2 Using the tty Telnet Client 214
2.2.2 HDSterm Functions 214
2.2.2.1 Commands Pull-down Menu 215
2.2.2.2 Customize Pull-down Menu 216
2.2.2.3 Serial Options Pull-down Menu 219
2.2.2.4 Help Pull-down Menu 219
2.2.3 HDSterm Operation 219
2.2.3.1 HDSterm Manual Starting 220
2.2.3.2 HDSterm rlogin Capability 220
2.2.3.3 Auto-Starting an Application in HDSterm 221
2.2.4 HDSterm Manual Page 226
2.2.5 HDSTerm Resource Files 237
2.2.5.1 HDSTerm Resources in HDS$$ROM 237
2.2.5.2 HDSTerm Resources in $HOME/.Xdefaults 239
2.2.6 HDSterm Command Sequences 239
2.2.6.1 HDSterm Supported Command Set: 239
2.2.7 HDSterm Print Commands 244
2.2.8 VT320 Keyboard Mapping 245
2.2.8.1 Using IBM 101+ or 102 keyboards: 245
2.2.8.2 Using DEC VT320 keyboards 246
2.2.9 HDSterm Key Translations 246
2.2.9.1 Sample Key Translations 247
2.3 HDS Local Terminal Emulation Clients 255
2.3.1 Using Local Clients 255
2.3.2 HDS3270 Terminal Emulation 255

2.3.3 Open Connect tn3270 Terminal Emulation 260
2.3.4 Tandem 6530 Terminal Emulation 260
2.4 Motif Window Manager 261
2.4.1 Starting Motif Window Manager 261
2.4.2 Starting Local MWM Remotely 262
2.4.3 MWM Operations 263
2.4.3.1 Pointer Shapes 263
2.4.3.2 Mouse Buttons 263
2.4.3.3 Mouse Buttons and Modifier Keys 263
2.4.3.4 Selecting a Window with MWM 264
2.4.3.5 Window Decoration 264
2.4.3.6 Moving a Window 265
2.4.3.7 Resizing a Window 265
2.4.3.8 Iconifying and Deiconifying a Window 265
2.4.3.9 Maximizing a Window 265
2.4.3.10 MWM Window Menu 265
2.4.4 Using the MWM Root Menu 266
2.4.5 Exiting MWM 267
2.4.6 Customizing the Motif Window Manager 267
2.4.6.1 MWM Resource Files 268
2.4.7 Starting Local Processes from the Local MWM 268
2.4.8 Starting Remote Processes from the Local MWM 269
2.4.8.1 MWM Menu with HDSperx 269
2.4.9 Editing the Root Menu Description 270
2.4.9.1 Sample Customized Mwm Root Menu 271
2.4.9.2 Starting Clients with Mwm 274
2.4.10 Motif Resource Editor 275
2.4.11 Local MWM Default Files 276
2.4.11.1 MWM App-defaults File 276
2.4.11.2 MWM .mwmrc File 277
2.4.12 Window Manager Remote Starting Checklist 280
2.4.13 Using HDSperx Clients with MWM 283
2.4.14 HDSrunwm Launcher Program 284
2.5 OPEN LOOK Virtual Window Manager 287
2.5.1 Starting OPEN LOOK Virtual Window Manager 288
2.5.2 Starting Local OLVWM Remotely 288
2.5.3 The OPEN LOOK Environment 289
2.5.4 OPEN LOOK Virtual Window Manager Operations 290
2.5.5 Using the OLVWM Window Manager 291

2.5.5.1 Pointer Shapes 291
2.5.5.2 Mouse Buttons 291
2.5.5.3 Mouse Buttons and Modifier Keys 291
2.5.5.4 Selecting a Window with OLVWM 292
2.5.5.5 Window Decoration 292
2.5.5.6 Moving a Window 293
2.5.5.7 Resizing a Window 293
2.5.5.8 Iconifying and Deiconifying a Window 293
2.5.6 OLVWM Menu 293
2.5.7 Using HDSperx with OLVWM 294
2.5.8 Exiting OLVWM 295
2.5.9 Starting Local Processes from the Local OLVWM 295
2.5.10 Starting Remote Processes with Local OLVWM 295
2.5.11 Customizing OLVWM 295
2.5.12 OPEN LOOK Window Manager Menu 296
2.5.12.1 Starting Clients with OLWM 297
2.5.13 Using HDSperx Clients with OLWM 298
2.5.13.1 OLWM .openwin-menu Sample File 298
2.5.13.2 HDSperx Menu Contents 300
2.5.13.3 HDSperx Menu Handling 300
2.5.14 Window Manager Remote Starting Checklist 300
2.5.15 OLWM Version 1.0 Syntax Differences 304
2.5.16 HDS OLVWM Extensions 305
2.5.17 HDSrunwm Launcher Program 306
2.6 HDSperx Clients 309
2.6.1 HDSperx Manual Pages 309
2.6.2 HDScalc Manual Pages 311
2.6.3 HDS Chooser Client 319
2.6.4 HDSdclock Manual Page 321
2.6.5 HDSoclock Manual Page 326
2.6.6 HDSxclock Manual Page 328
2.6.7 HDSscreensaver Manual Page 331
2.6.8 HDStetris Manual Page 336
2.6.9 HDSview Manual Page 337
2.6.10 Waitforserver Manual Page 355
2.6.11 Xhost Manual Page 356
2.6.12 Xmodmap Manual Page 358
2.6.13 Xset Manual Page 363

2.6.14 FDFORMAT Disk Format Utility 366
2.6.15 PEX 3D Server Extension 367
2.6.15.1 Installation of the HDS PEX-3D Extension 367
2.6.15.2 HDS PEX-3D Limitations 367
2.6.15.3 HDS PEX-3D Functional Description 368
2.6.15.4 HDS PEX-3D Implementation Dependent Constants 371
2.6.15.5 HDS PEX-3D Lookup Tables 372
2.6.15.6 HDS PEX-3D Lookup Table default predefined values 372
2.6.16 Xscope Manual Page 377
2.6.17 PING Manual Page 379
2.6.18 HDSshare Manual Page 381
2.6.18.1 Using the HDSshare Client 381
2.6.19 Xpmroot Manual Page 382
2.7 HDS ViewStation Multimedia Operations 385
2.7.1 HDS ViewStation Audio Operation 385
2.7.1.1 Audio Operation Hardware 385
2.7.1.2 Audio Device Setting Menu in Setup Mode 385
2.7.1.3 ViewStation Audio Client Overview 387
2.7.1.4 HDSperx Audio Control client - hdsaudio 387
2.7.1.5 Starting the HDS Audio Client 389
2.7.1.6 HDSperx NetAudio compatible server - auhds 389
2.7.1.7 HDSperx Audio File Playback program - hdsplay 390
2.7.1.8 HDSperx Audio File Recording program - hdsrecord 390
2.7.1.9 Using ViewStation Audio with Application Programs 390
2.7.1.10 ViewStation Audio Specifications 391
2.7.2 HDS ViewStation Video Operations 392
2.7.2.1 HDS Video Hardware Configuration 392
2.7.2.2 ViewStation with HDS netVideo Hardware 392
2.7.2.3 HDS netVideo (analog and digital video) Specifications 393
2.7.2.4 HDS netVideo Playback (digital video) Specifications 393
2.7.3 HDSware netVideo Software 394
2.7.3.1 Installing HDSware netVideo Software 394
2.7.4 HDS netVideo Clients 395
2.7.4.1 Using the HDS Video Client 396
2.7.4.2 Using the HDS Movie Client 401
2.7.4.3 Using the HDS Conference Client 403
2.7.4.4 Using the HDS Tuner Client 408
2.7.4.5 HDS netTV Simulcast Client 410
2.7.4.6 HDS netTV Viewer Client 413
2.7.5 Frequently Asked Questions: 417

2.7.5.1 How does Video affect my network? 417
2.7.5.2 How large are digital video files in different formats? 418
2.7.5.3 What do I need for my in-house video training? 418
2.7.5.4 What are some other applications for netVideo ViewStations? 418
2.7.6 HDS Multimedia Client Manual Pages 419
2.7.6.1 HDS Audio Local Client Manual Page 419
2.7.6.2 HDS Play Audio Client Manual Page 423
2.7.6.3 HDS Au Audio Client Manual Page 425
2.7.6.4 NetAudio Audio Client Manual Page 427
2.7.6.5 HDS Conference Manual Page 430
2.7.6.6 HDS Movie Manual Page 436
2.7.6.7 HDS NetTV Simulcast Manual Page 439
2.7.6.8 HDS NetTV Viewer Manual Page 445
2.7.6.9 HDS Tuner Manual Page 448
2.7.6.10 HDS Video Manual Page 451
2.7.6.11 HDS Record Client Man Page 456
2.8 FVWM Virtual Window Manager 459
2.8.1 FVWM Manual Page 459
2.8.2 FvwmAudio Manual Page 490
2.8.3 FvwmAuto Manual Page 492
2.8.4 FvwmBacker Manual Page 493
2.8.5 FvwmBanner Manual Page 494
2.8.6 FvwmClean Manual Page 495
2.8.7 FvwmDebug Manual Page 496
2.8.8 Fvwm Goodstuff Manual Page 497
2.8.9 FvwmIconBox Manual Page 500
2.8.10 FvwmIdent Manual Page 505
2.8.11 FvwmPager Manual Page 506
2.8.12 FvwmScroll Manual Page 509
2.8.13 FvwmWinlist Manual Page 510
Section 3.0 - Network and Host System Operations 513
3.1 Network File Transfer Operations 513
3.1.1 ViewStation File Transfer Protocols 513
3.1.1.1 Selecting a File Transfer Protocol 513
3.1.2 NFS 514
3.1.2.1 For Sun OS 4.1.x and BSD type UNIX: 514
3.1.2.2 For Solaris 2.x and Sys V R4.x type UNIX: 514
3.1.2.3 Setting Up the ViewStation for NFS 515
3.1.2.4 Using NFS for ViewStation File Access 515

3.1.3 TFTP 517
3.1.3.1 Using TFTP 517
3.1.3.2 Starting TFTP on your Host 517
3.1.3.3 TFTP in Secure Mode 517
3.1.3.4 TFTP in Secure Mode on Sun Systems 518
3.1.3.5 TFTP on Hewlett Packard Systems 519
3.1.3.6 TFTP File Transfer Protocol on IBM RS6000 Systems 521
3.1.4 FTP 521
3.1.4.1 Anonymous FTP File Transfer 521
3.2 BOOTP Operations 523
3.2.1 Starting with BOOTP 523
3.2.1.1 Ethernet Hardware Address 523
3.2.1.2 Bootptab Command Fields 524
3.2.1.3 Problems with BOOTP Starting 525
3.2.2 Setting Up BOOTP on your Host 525
3.2.2.1 Installing BOOTP on your Host 526
3.2.2.2 BOOTP Debugging 527
3.2.3 Sample bootptab File 527
3.2.3.1 Password Security with BOOTP 528
3.3 XDMCP Operations 529
3.3.1 Starting Xdm on your Host 529
3.3.2 Xdm Configuration Files 530
3.3.2.1 Xdm Authorize 530
3.3.2.2 Xresources File 530
3.3.2.3 Xservers File 530
3.3.2.4 Xsetup File 531
3.3.2.5 Xstartup File 531
3.3.2.6 Xreset File 531
3.3.2.7 xdm-errors File 531
3.3.2.8 xdm-pid 531
3.3.2.9 Xsession file 531
3.3.3 Xdm Versions 531
3.3.4 Using Xdm with your ViewStation 532
3.3.4.1 Xdm Direct Access 532
3.3.4.2 Xdm Indirect Access 532
3.3.4.3 Xdm Broadcast Access 532
3.3.4.4 Xdm Chooser 532
3.3.4.5 ViewStation's Chooser Client 532
3.3.5 Using XDM on the ViewStation 533
3.3.5.1 XDMCP Options Selection in Setup Mode 533

3.3.6 Using the .xsession File 535
3.3.6.1 Vendor Display Class String 537
3.3.6.2 ViewStation .xsession File 537
3.4 X Resource Operations 539
3.4.1 X Resource Format 539
3.4.1.1 X Resource Syntax 540
3.4.1.2 Resource Classes and Resource Instances 540
3.4.1.3 Class and Instance Hierarchy 540
3.4.1.4 Resource Bindings 541
3.4.1.5 Resource Precedence 541
3.4.1.6 X Client - editres 541
3.4.1.7 X Client - mre 541
3.4.2 X Resource Database 542
3.4.2.1 The X Resource File 542
3.4.2.2 Application Resource Files 544
3.4.2.3 Run Command Files for Resources 544
3.4.2.4 Server-related Resource Files 545
3.4.2.5 Host-related Resource Files 545
3.4.2.6 Resource Specification from the Command Line 545
3.4.2.7 ViewStation Local Client Resources 545
3.4.3 Loading Resource Files with Xrdb 545
3.5 Network Printing Operations 547
3.5.1 The LPD Print Spooler for BSD Systems 547
3.5.1.1 LPD Daemon 547
3.5.1.2 Sample Input Filter File 548
3.5.1.3 Spooler Shell Script 549
3.5.1.4 Printcap File 551
3.5.1.5 ViewStation Print Spooler Example 552
3.5.2 The LP Print Service for UNIX System V Printing 553
3.5.2.1 Using LP Print Service with the ViewStation 553
3.5.2.2 Sample Printer Model File 553
3.5.2.3 Adding a ViewStation Printer to the LP Print Service 554
3.5.2.4 HDS-Model File 555
Section 4.0 - Using the ViewStation on Selected Host Systems 559
4.1 Sun Workstation Configuration 559
4.1.1 Sun Network Services 559
4.1.2 TFTP File Transfer Protocol 560
4.1.2.1 TFTP in Secure Mode 560
4.1.2.2 Paths with Secure TFTP 561

4.1.3 FTP File Transfer Protocol 561
4.1.3.1 Anonymous FTP File Transfer 561
4.1.4 NFS File Transfer Protocol 562
4.1.5 Other Network Services 563
4.1.5.1 BOOTP (Boot Protocol Daemon) 563
4.1.5.2 RARP (Reverse Address Resolution Protocol) 563
4.1.6 Open Windows Applications 563
4.1.6.1 Open Windows Environments 563
4.1.7 OLWM Window Manager 564
4.2 Hewlett Packard Systems 567
4.2.1 Preparing the HP700 567
4.2.1.1 Load the HDSware Tape on the HP host 567
4.2.1.2 Verify the startup method 567
4.2.1.3 Verify the file access method 568
4.2.1.4 TFTP - Trivial File Transfer Protocol 568
4.2.1.5 TFTP in "Secure Mode" on HP Systems 569
4.2.1.6 ViewStation File Path Specifications 570
4.2.1.7 FTP - File Transfer Protocol 570
4.2.1.8 Anonymous FTP File Transfer 570
4.2.1.9 NFS - Network File System 571
4.2.2 Configure the User Account 571
4.2.2.1 User's .vue Directory 571
4.2.3 vue.settings file 572
4.2.4 Optional Customization for the ViewStation 572
4.2.4.1 vue.session file 573
4.2.5 Other important files in the .vue directory 573
4.2.5.1 palette.vue 573
4.2.5.2 vue.resources 574
4.2.5.3 Creating a Common .vue Environment 574
4.2.6 Settings on the HDS ViewStation 574
4.3 DEC VMS Systems 575
4.3.1 Overview 575
4.3.2 DEC UCX Protocol 575
4.3.3 ViewStation Setup Mode Entries for VMS 576
4.3.4 NFS on VMS Systems 577
4.3.5 HDSware Installation for VMS Systems 577
4.3.6 Booting the ViewStation from a VMS Host 578
4.3.7 Automatic ViewStation Configuration 579
4.3.8 HDSware Tape Contents 579

4.3.8.1 VMS File Names and Syntax 581
4.3.9 Handling Special DEC or VMS Fonts 581
4.3.9.1 Preparing DEC Fonts 581
4.3.9.2 Preparing Other Special Fonts 582
4.3.10 DECnet Operation with Open Windows 582
4.3.11 Some typical problems on VMS 583
4.4 IBM RS6000 Systems 585
4.4.1 Overview 585
4.4.2 TFTP File Transfer Protocol 585
4.4.3 IBM RS6000 Fonts 586
4.4.4 RSH Permissions on the RS6000 586
4.4.5 Starting Xdm on the RS6000 586
Section 5.0 - HDS ViewStation Developer's Kit 587
5.1 HDS ViewStation Developer's Kit Overview 587
5.1.1 **HDS ViewStation Architecture 587**
5.1.2 Developer's Kit Contents 587
5.1.2.1 GNU Tools 588
5.1.2.2 HDS Libraries 588
Section 6.0 - HDS ViewStation Services 589
6.1 HDS ViewStation Warranty 589
6.2 HDS ViewStation Service 590
6.2.1 HDS ViewStation Upgrades 590
6.2.1.1 Memory Upgrades 590
6.2.2 HDSware Software Upgrades 591
6.2.2.1 HDSware Premium Software Service 591
INDEX 593

HDS ViewStation: Specifications

This chapter gives the hardware and software specifications for the HDS ViewStation.

1.0 Section 1.0 - HDS ViewStation

1.1 HDS ViewStation Specifications

1.1.1 HDS ViewStation Overview

The HDS ViewStation is a modular X Window System terminal composed of a base unit, a keyboard, a mouse, and a display monitor.

The ViewStation uses standard components for its keyboard, mouse, and monitor. Other devices using the same specifications can be substituted. See the sections below for some supported devices.

1.1.2 Hardware Specifications

1.1.2.1 ViewStation Base Unit

The ViewStation's base unit contains a highly integrated main circuit board and a power supply. These are described individually below.

1.1.2.2 ViewStation Circuit Board

The main circuit board contains the main processor, an Intel 80960CA RISC processor. Various processor speeds are supported, including 16, 25, and 33 MHz. The main circuit board also contains a high performance Ethernet controller, the Intel 82596CA processor.

1.1.2.3 ViewStation Memory

The ViewStation main circuit board contains memory of up to 132 MB of DRAM memory, depending on the density of the memory modules. The memory consists

of DRAM sockets which accept 0, 2, 4, 8, 16, 32, or 64 MB SIMM modules (some boards also have 4 MB of memory on the main board). Various combinations of these values are available.

Optional flashPROM memory and other storage devices for the server code is available. Booting is done with a PLCC boot prom, either 2 or 4 Mb; or from memory allocated for the bootprom on the flashPROM board. The board also contains an 8 KB Non Volatile Memory chip.

The ViewStation with the FXA / B / C / D circuit boards can accept SIMM modules in any mixture. The ViewStation Ultra must use SIMM modules as identical pairs of SIMMs.

ViewStation memory uses standard 32-bit wide SIMM modules with speeds 70 ns or faster. The 72-pin SIMMs should have tin-lead contacts. These can be ordered from HDS or obtained from other market sources.

1.1.2.4 ViewStation Bootprom

The ViewStation uses a PLCC boot prom or a dedicated memory partition on the FlashPROM board (jumper selectable) for easy upgrades. The bootprom has 2 Mb capacity. There is an optional bootprom (Vnn-E) available with 4 Mb capacity for Token Ring network operation.

1.1.2.5 ViewStation Video Memory

Video memory supports resolutions up to 1280x1024 pixels, with 1 or 8 bit planes of depth. Memory is installed as either 1 MB or 2 MB, depending on the monitor requirements. Color handling is done with a TI TLC34075 palette chip and supports 8 bitplanes of color, 256 colors from a palette of 16.7 million.

1.1.2.6 ViewStation External Connections

The main circuit board has ports for the video connector, the mouse and keyboard, as well as a Centronics parallel port (DB25 connector) for local printing, an RS232 serial port (RJ11 connector), and a Twisted-pair Ethernet port (RJ45 connector). Some boards also have an optional separate connector for a Sun keyboard.

1.1.2.7 ViewStation Optional Communications Board

An optional communications board may be added. This board contains a ThinWire Ethernet port (BNC connector), a ThickWire Ethernet port (DB15 connector), and a second RS232 serial port (DB25 connector).

1.1.2.8 ViewStation FlashPROM Board

An optional FlashPROM board is available with 2.0 MB or 4.0 MB of flash ROM memory. This board stores the ViewStation's server code, and optionally, the bootprom software for local booting. The ROM memory can be easily reprogrammed for server code updates (refer to Section 1.4.7).

1.1.2.9 ViewStation PC Card Board

An optional PC Card interface board is available. This connects to the main circuit board on the connector for the FlashPROM, so you would use the connector for either FlashPROM or PC CardPC Card cards. The PC Card board has slots for two PC Card devices and one for a FlashPROM board. Typically, you could use one for a FlashPROM storage device for booting, one for an audio board, and the third for an optional communications connection.

1.1.2.10 ViewStation PC Card Devices

The PC Card interface (also called the PCMCIA interface) is a specification for both hardware and software interfaces. There are a number of PC Card devices currently supported and many more are being added.

Currently supported PC Card devices are:

ROM storage - 2.5 MB and 5.0 MB ROM cards

Hard disks - 20 MB and 40 MB hard disk drives

Audio card - HDSaudio, with input and output connectors

Token Ring communications interface - Token Ring network connectivity

1.1.2.11 ViewStation Floppy Disk Drive

The ViewStation supports a 3.5" floppy disk drive using its parallel port. The installation and operation of the floppy disk drive are described in Section 1.12 of this manual.

1.1.3 ViewStation Power Supply

The internal power supply is switching-type and is rated at 40 W. It automatically detects and accepts either 120 VAC or 240 VAC line voltage. The convenience outlet on the back panel provides the same voltage output as the voltage input.

1.1.4 ViewStation Keyboards

The ViewStation supports a number of different keyboards, shown below:

- IBM 101+ style keyboard (AT type) - this is the standard keyboard
- DEC style (VT420 type)
- Sun Type 4 style or Type 5 style (using either Serial Port 1 with special modifications for older ViewStations) or with a Sun keyboard connector on FXD and Ultra ViewStation models
- IBM 102 key multinational keyboard
- Lexmark 122 key IBM 3270 keyboard

All keyboards (except Sun) use a five-pin DIN male connector with an IBM interface. Complete key remappings and configurations are selectable from software.

1.1.5 ViewStation Mouse and Pointing Devices

The ViewStation supports a number of pointing devices:

- Three-button opto-mechanical PS/2 style mouse which uses a six-pin mini DIN connector.
- A serial mouse and other serial devices can be used on the ViewStation. Serial devices are automatically sensed on the serial ports and the appropriate drivers used.
- Trackballs in a number of configurations
- Light pen, FTG Data Systems, IBM F10 and MAC F11 Model PXL-780
- Other devices, such as bar code readers, digitizing tablets, etc. can be used with the serial ports and the HDSware software "opendev" and "usepty" commands.

1.1.6 ViewStation Display Monitors

The ViewStation supports a number of different monitors, in sizes from 14" to 35". The resolution and color depth of the monitors are selectable in Setup Mode (up to the limits of the installed VRAM). This list is subject to change without notice.

- FX14 - 14" monochrome/gray-scale monitor
  gray-scale is selected for this monitor with JP12 5-6
  Resolution - 1024x768 pixels, 100x100 dpi; 1 or 256 shades
  Size - 12.6"W x 13.4"H x 12.5"D
  Weight - 16 lbs.
  Frequency - 57 KHz horiz.; 70 Hz vert.
- FX15 - 15" monochrome/gray-scale monitor
  Resolution - 1024x864 pixels; 100x100 dpi; 1 or 256 shades
  Resolution - 1024x800 pixels; 100x100 dpi; 1 or 256 shades
  Size - 14.2"W x 14.4"H x 14.8"D
  Weight - 25 lbs.
  Frequency: 64 KHz horiz.; 72 Hz vert.
- FX17 - 17" monochrome/gray-scale monitor
  Resolution - 1024x864 pixels; 85x85 dpi; 1 or 256 shades
  Resolution - 1152x900 pixels; 93x93 dpi; 1 or 256 shades
  Size - 16.2"W x 15.3"H x 17.1"D
  Weight - 44 lbs.
  Frequency: 68 KHz horiz.; 74 Hz vert.
- FX19M - 19" monochrome/gray-scale monitor
  Resolution - 1024x864 pixels; 85x85 dpi; 1 or 256 shades
  Resolution - 1152x900 pixels; 93x93 dpi; 1 or 256 shades
  Resolution - 1280x1024 pixels; 100x100 dpi; 1 or 256 shades
  Size - 18.5"W x 18.5"H x 16.0"D
  Weight - 53 lbs.
  Frequency: 77 KHz horiz.; 72 Hz vert.
- LX14C, FX14C, Ultra 14C - 14" color monitor
  1024x768 pixels; 100x100 dpi; 256 colors
  Size: 13.6"W x 13.7" H x 15"D

Weight: 27 lbs.
Frequency: 57 kHz horiz.; 70 Hz vertical

- LX15C, FX15C, Ultra 15C - 15" color monitor
  Resolution - 1024x768 pixels; 256 colors
  Size: 14.2"W x 14.9"H x 15"D
  Weight: 29 lbs.
  Frequency: 57 kHz horiz.; 70 Hz vertical
- LX17C, FX17C, Ultra 17C - 17" color monitor
  Resolution - 1024x864 pixels; 85x85 dpi; 256 colors
  Resolution - 1152x900 pixels; 93x93 dpi; 256 colors
  Size: 16.75"W x 16.75"H x 18.1"D
  Weight: 55 lbs.
  Frequency: 65 kHz horiz; 71 Hz vertical
- FX17CT, Ultra 17CT - 17" Trinitron color monitor
  Resolution - 1024x864 pixels; 85x85 dpi; 256 colors
  Resolution - 1152x900 pixels; 93x93 dpi; 256 colors
  Resolution - 1280x1024 pixels; 100x100 dpi; 256 colors
  Size: 16"W x 16"H x 17.3"D
  Weight: 47 lbs.
  Frequency:76 kHz horiz; 72 Hz vertical
- FX19CT, Ultra 19CT - 19" Trinitron color monitor
  Resolution - 1152x900 pixels; 93x93 dpi; 256 colors
  Resolution - 1280x1024 pixels; 100x100 dpi; 256 colors
  Size: 18.9"W x 19.3"H x 20.7"D
  Weight: 80 lbs.
  Frequency: 77 kHz horiz.; 79/72 Hz vertical
- LX19C, FX19Ci, Ultra 19Ci - 19" Invar-mask color monitor
  Resolution - 1152x900 pixels; 93x93 dpi; 256 colors
  Resolution - 1280x1024 pixels; 100x100 dpi; 256 colors
  Size: 19.7"W x 18.5"H x 19.9"D
  Weight: 64 lbs.
  Frequency: 77 kHz horiz.; 79/72 Hz vertical

1.1.6.1 Selecting the Monitor Type

The following table shows monitors currently supported by HDS. Monochrome and grey-scale monitors typically differ only in the amount of VRAM present. Note that the sync frequency varies with the resolution (which is user-selectable) of the monitor; the values below are the HDS factory setting. Some multi-sync monitors have a frequency shown which is the HDS factory specification appropriate for optimum display characteristics. * indicates a jumper is needed.

ViewStation Model	Monitor Number	Resolution	Sync. Freq.	Hex Setting
15" Monochrome	MO-V15M	1024x864	65 KHz/72 Hz	4
17" Monochrome	MO-V17H	1152x900	76 KHz/72 Hz	F
19" Monochrome	MO-19M	1280x1024	77 KHz/72 Hz	F

14" Color	MO-14CS	1024x768	57 KHz / 71 Hz	5
15" Color	MO-15CS	1024x768	57 KHz / 71 Hz	5
17" Color	MO-V17CT / CH	1280x1024	65 KHz / 71 Hz	3
17" Color (Trinitron)	MO-V17CTN	1280x1024	76 KHz / 72 Hz	7
19" Color (Invar)	MO-V19CS / Ci	1280x1024	77 KHz / 72 Hz	7
19" Color (Trinitron)	MO-19CT	1280x1024	77 KHz / 72 Hz	7
ViewStation Monitors no longer shipping				
19" Color (Trinitron)	MO-V19C	1280x1024	64 KHz / 61 Hz	0
19" Monochrome	MO-V19MM	1280x1024	68 KHz / 64 Hz	1
17" Monochrome	MO-V17M	1024x864	67 KHz / 74 Hz	2
16" Color	MO-V16C	1024x864	65 KHz / 71 Hz	3 *
14" Color	MO-V14C	1024x768	57 KHz / 71 Hz	5
14" Monochrome	MO-V14R	1024x768	57 KHz / 71 Hz	5 *
19" Monochrome	MO-19MS	1152x900	64 KHz / 68 Hz	6
VGA (needs factory modification)		640x480		8
VGA (needs factory modification)		800x600		9

The ViewStation produces video signals with separate sync signals or composite sync on green. Selecting JP12 3-4 on the ViewStation FX (JP 13 on the LX series, and JP13 on the Ultra series) turns off the sync on green signal (hex switch 5 video automatically shuts off sync on green).

The ViewStation uses negative polarity for its video signals. You can select positive polarity with jumpers on the main board in JP9 for the ViewStation FX (JP12 for the LX series, and JP 11 for the Ultra series), which is located near the round hex switch. To select positive polarity:

HSYNC - install jumper 1-2

VSYNC - install jumper 3-4

All monochrome monitors select grey-scale operation automatically, except the MO-V14R, which requires JP13 5-6.

1.1.6.2 Changing ViewStation Monitors

There are several considerations with changing monitors on the ViewStation.

1. Video memory:

The ViewStation can have either 1 MB or 2 MB VRAM configurations, based on its initial factory monitor type. The 1 MB configuration supports monochrome monitors at all resolutions up to 1280x1024 and color (and grey-scale) monitors at 1024x768 resolution. Higher resolution color (or grey-scale) monitors require the 2 MB VRAM configuration; this is a factory upgrade to the ViewStation base. You can see the memory available on your ViewStation by using the "hwconfig" command from the ViewStation's Console Window.

2. Cable Type:

The ViewStation base uses a DB-15 VGA-type connector for its video output, with standard VGA pin assignments, for all monitors. The ViewStation is shipped with a cable appropriate for the initial factory monitor type. Some monitors have attached cables, others use special connectors, or 1, 3, or 5 BNC connectors. It is important to check the cable type and verify the pin assignments when using a new monitor. The ViewStation's video connector pin assignments are in Section 1.1.10.5 of this manual; consult the monitor manuals for the video connector signals needed on the monitor side.

3. Signal Frequency:

In order to support the high resolution of the monitors and give sufficient refresh speed to be free of flicker, ViewStation monitors run at relatively high signal frequencies (for example, IBM-PC VGA monitors run at 31.5 KHz, while ViewStation monitors typically run in the 70 KHz range). Specific monitor frequencies are given in the table above, and a supported frequency is selected by the hex-switch setting on the circuit board.

There are other considerations about the video signal that present additional compatibility problems. Consult with HDS Technical Support for more details on these differences. There are three main areas of signal compatibility beyond sync frequency; whether the monitor uses sync on green signals, the polarity of the sync signal, and grey-scale / color mappings.

If you want to change the monitor on your ViewStation base, verify that there is sufficient VRAM to support the desired resolution, verify the correctness of the video cabling, and select the correct signal frequency in the ViewStation.

Selecting the hex-switch setting:

Disconnect the power to your ViewStation and remove the base's cover. A small round switch with numbers 0 to F on the top is located toward the left rear of the circuit board. Simply turn the white dot to the desired setting. Be sure to cycle the power to both the ViewStation and the monitor any time you change this setting.

Be careful in selecting the switch setting - you can damage the monitor if you select the wrong cable or setting. HDS cannot accept responsibility for unsupported monitors or configurations.

1.1.7 ViewStation Physical Characteristics

1.1.7.1 ViewStation Base Unit

Base Unit: 1.75"h x 13.5"w x 12.5"d (4.4cm. x 34.3cm. x 31.7cm.)

Weight: 7.5 lbs. (3.4 kg)

1.1.7.2 ViewStation Keyboards

IBM Keyboard: 1.4"h x 18.25"w x 8.25"d (3.5cm. x 46cm. x 21cm.)

Weight: 2.6 lbs. (1.2 kg)

DEC Keyboard: 1.5"h x 19"w x 8.25"d (3.5cm. x 48.5cm. x 21 cm.)

Weight: 3.75 lbs. (1.7 kg)

Sun Type 4 Keyboard: 1.64"h x 18"w x 8.1"d

Weight: 2.75 lbs.

Sun Type 5 Keyboard:

Lexmark 122 Keyboard:

1.1.7.3 ViewStation Pointing Devices

Mouse: 1.12"h x 2.75"w x 3.75"d (2.8cm. x 7cm. x 9.5cm.)

Weight: 3 oz. (0.1 kg)

1.1.7.4 ViewStation Monitors

Monitors: various models as shown above. Other monitors, such as flat-panel, rack-mounted, and energy-saving are also available.

1.1.8 ViewStation Environmental Requirements:

1.1.8.1 Operation

Temperature: 10 - 40 deg C (50 - 104 deg F)

Relative Humidity: 10 - 90%

Maximum Wet Bulb: 28 deg C (82 deg F)

Minimum Dew Point: 2 deg C (36 deg F)

Maximum Altitude: 2.4 km. (10,000 ft.)

1.1.8.2 Storage

Temperature: -15 - 70 deg C (5 - 158 deg F)

Relative Humidity: 5 - 95%

Maximum Altitude: 9.1 km. (30,000 ft.)

1.1.9 ViewStation Controls and Indicators

Base Unit: Front-mounted power switch, power LED.

Display Monitor: Brightness and Contrast controls, also other controls depending on the monitor.

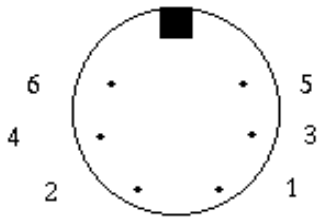
Keyboard: Three LED indicators.

1.1.10 ViewStation Connections

The ViewStation has a number of connectors on the back panel. These are shown in detail below.

1.1.10.1 Mouse Connector

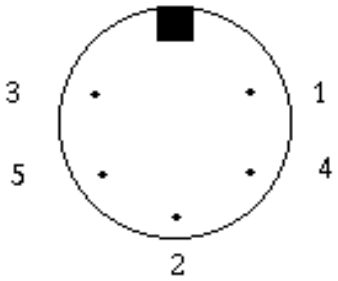
A six-pin mini DIN male connector with the following configuration; pin assignments are shown:



PIN	SIGNAL
1	Data
2	Not Used
3	Ground
4	VCC
5	Clock
6	Not Used
Shell	Protective Ground

1.1.10.2 Keyboard Connector

A five-pin DIN female connector with the following configuration; pin assignments are shown:



PIN	SIGNAL
1	Keyboard Clock
2	Keyboard Data
3	Not Used
4	Ground
5	+5 VDC
Shell	Protective Ground

1.1.10.3 Serial Port - RJ11 Connector

A six-pin RS-423 / RS-232 port using an unshielded female RJ11 connector with the following configuration: (Pin 6 is on the right edge of the connector)

PIN	SIGNAL
1	RTS
2	Transmit Data (output)
3	DTR
4	Ground
5	Receive Data (input)
6	CTS

Baud rates from 110 to 57.6 Kbaud are supported.

Note: ViewStations which use the Sun keyboard option have a special connector for the keyboard. Older ViewStation models convert the RJ11 serial port for the Sun keyboard connection.

If the RJ11 port is used for a Sun-style keyboard connection, the pin assignments are different. The connection is reworked on the main circuit board so the serial port capability is disabled. The RJ11 port has a special label "Sun-style Kbd", which indicates this rework. **WARNING:** you will damage the Sun-style keyboard if you plug it into a normal RJ11 port without the keyboard rework. The following pin assignments are present for the RJ11 connector for a Sun keyboard connection:

PIN	SIGNAL
1	RTS (not used)
2	TX - Transmit Data (output)
3	+ 5 V
4	Ground
5	RX - Receive Data (input)
6	CTS (not used)

1.1.10.4 Parallel Port - DB25 Connector

A 25-pin bidirectional Centronics parallel port with a female DB25 connector having the following configuration:

PIN	SIGNAL
1	Strobe (I/O)
2	Data 0 (I/O)
3	Data 1 (I/O)
4	Data 2 (I/O)
5	Data 3 (I/O)
6	Data 4 (I/O)
7	Data 5 (I/O)
8	Data 6 (I/O)
9	Data 7 (I/O)
10	ACK (input)
11	BUSY (input)
12	PE (input)
13	Select (input)
14	Auto Feed (output)
15	ERR (input)

16	INIT (output)
17	Select In (output)
18	Logic Ground
19	Logic Ground
20	Logic Ground
21	Logic Ground
22	Logic Ground
23	Logic Ground
24	Logic Ground
25	Logic Ground

1.1.10.5 Video Connector - DB15 high-density (VGA-type)

A 15-pin high-density female DB15 connector having the following configuration:

PIN	SIGNAL
1	Red
2	Green
3	Blue
4	Not Used
5	Ground
6	Red Return
7	Green Return
8	Blue Return
9	Not Used
10	Ground
11	Not Used
12	Not Used
13	Horiz. Sync.
14	Vert. Sync.
15	Not Used

1.1.10.6 Twisted-Pair Ethernet - RJ45 Connector

An 8-pin connector conforming to IEEE 802.3 10BaseT standards with the following configuration (Pin 1 is on the left side of the opening):

PIN	SIGNAL
1	Transmit Data +
2	Transmit Data -
3	Receive Data +

4	Not Used
5	Not Used
6	Receive Data -
7	Not Used
8	Not Used

Note that the ViewStation uses the standard Twisted-Pair interface of an unshielded 100 Ohm cable termination as its default setting. If you are using a shielded 150 Ohm cable termination, you should select jumpers 1-2 on JP15.

If your Twisted-Pair network does not use Link Integrity, you have two methods of disabling this on the ViewStation. Jumpers 1-2 on JP16 selects software specification of the Link Integrity setting, with the "Thin" setting enabling Link Integrity and the "Thick" setting disabling it. If you select jumpers 3-4 on JP17, this disables Link Integrity on the circuit board and is independent of the software setting.

1.1.10.7 ThinWire Ethernet - BNC Connector

A BNC female connector conforming to IEEE 802.3 10Base2 standards.

1.1.10.8 ThickWire Ethernet - DB15 Connector

A 15-pin female DB15 connector conforming to IEEE 802.3 10Base5 standards with the following configuration:

PIN	SIGNAL
1	Control In Circuit Shield
2	Control In Circuit A
3	Data Out Circuit A
4	Data In Circuit Shield
5	Data In Circuit A
6	Voltage Common
7	Not Used
8	Option Shield
9	Control In Circuit B
10	Data Out Circuit B
11	Data Out Circuit Shield
12	Data In Circuit B
13	Voltage Plus
14	Voltage Shield
15	Not Used
Shell	Protective Ground

1.1.10.9 Serial Port - DB-25 Connector

An RS-423/RS-232 DB25 male connector with the following configuration:

PIN	SIGNAL
1	Chassis Ground
2	Transmit Data (output)
3	Receive Data (input)
4	Request To Send (output)
5	Clear To Send (input)
6	Not Used
7	Signal Ground
8	Not Used
20	Data Terminal Ready (output)
All other pins are not used.	
Shell	Protective Ground

Baud rates from 110 to 57.6 Kbaud are supported.

1.1.10.10 Sun Keyboard Connector

This is an 8-pin mini-DIN connector with the following configuration:

PIN	SIGNAL
1	Ground
2	Ground
3	Vcc
4	NC
5	Receive
6	Transmit
7	NC
8	Vcc

1.1.10.11 Audio Ports - Input/Output Connections

The Input and Output ports use 3.5 mm jacks. These jacks are only installed on the back panel if the Audio option is installed.

1.1.10.12 PC CardPort

The PC Card option requires the PC Card adapter card. This card has connectors for an internal HDS FlashPROM card and two PC Card devices, one internal and one with an external connection in the ViewStation's back panel.

1.1.11 ViewStation Software

1.1.11.1 ViewStation Server Components

- Optimized version of the X Window System server, based on X11R5 effective with HDSware version 3.0.
- Multiple X11 extensions, both internal and downloadable
- Local Motif, OPEN LOOK, and FVWM Window Managers
- Local HDSterm VT320 emulation client (used for Telnet, Serial, and Console windows)
- Local ANSI terminal client (used optionally instead of HDSterm client)
- HDSperx downloadable clients (see Section 2.0)
- Setup Mode (configured as a local X client)
- ViewStation Console Window (configured as a local shell, a remote process, or serial connection)
- Available Memory Window (configured as a local X client)
- Low Memory Warning client (configured as a local X client)
- Downloadable RAM-based or flashPROM-based server operation
- Dynamic Font Loading and Font Server support
- XDMCP Protocol support including a local Xdm chooser client
- Multiple ViewStation clients running simultaneously (running locally or remotely as X clients)
- Downloadable Setup Mode configuration
- Point to Point Protocol (PPP) serial line connections
- Serial Line IP connections (SLIP)
- Compressed Serial Line IP connections (CSLIP)
- Downloadable RGB Colormaps
- Domain Name Service
- SNMP Network Management Protocol

1.1.11.2 Setup Mode

Setup Mode provides the ability to set many communication, network, and user preference settings. Using the "HDSware host" feature selects automatic entry of many needed paths in their Setup Mode fields. The Main Menu contains eight buttons for starting all the ViewStation clients:

Setup Mode Main Menu

- Telnet Terminal Window - local HDSterm client (with Run at Restart option)
- Serial Terminal Window - local HDSterm client (with Run at Restart option)
- Console Window client (with Run at Restart option)
- Available Memory client (with Run at Restart option)
- Motif Window Manager client (with Run at Restart option)
- HDSware Configuration - sets IP address and HDSware location

- Setup Options Menu
- OPEN LOOK Window Manager client (with Run at Restart option)

Setup Options Menu

The "Setup Options" button brings up the following menu:

- Return to Main Menu
- Communications Options
- XDMCP Options
- Remote Hosts
- Domain Name Servers
- User Preferences
- Startup Options
- Restart and Shutdown Menu
- Reset NVM to Factory Defaults
- Restore Settings from NVM

Some of the principal functions are listed below as they appear in these menus:

Communications Options Menu

This menu lets you set various network specifications, such as netmask and gateway information. This menu sets the Ethernet and Token Ring communications, and also SLIP, CSLIP, and Serial Terminal Emulation communications operations.

XDMCP Options Menu

This menu allows you to select the XDM Display Manager to manage your ViewStation display, along with the type of XDMCP message to send to initiate XDM and to set XDM timeout values. This menu lets you start the Xdm Chooser, a local ViewStation client that displays available Xdm hosts for users who select Indirect or Broadcast Xdm access.

Remote Hosts Menu

This menu sets host information for up to six remote hosts, specifying for each the host name and address and other host information, including FTP or TFTP selection and access. It also permits specification of a Local and a Remote Mount Directory for each host.

Domain Name Servers Menu

This menu lets you enter the addresses of the one or two hosts to use to resolve Domain Names and the Default Suffix value for your network. There is also capability for internal name service.

User Preferences Menu

This menu allows you to set a number of user preferences, including fontpath and color map path information, keyboard and bell volume, screen saver settings, backing store and save under features, international keyboard type, and resolution and color depth settings for your monitor. This menu also allows you to enter a user password to give you Setup Mode security.

Startup Options Menu

This menu controls the ViewStation operation with the RARP and BOOTP protocols. This menu sets paths for server code download, configuration file location, and startup file location.

Restart and Shutdown Menu

This menu restarts the ViewStation. This function operates to restart the ViewStation using the ROM server image, the current RAM server image, or reload the server image from the network, depending on the user's selections.

Reset NVM to Factory Defaults

This menu lets you change all your Setup Mode selections (except the password, and screen resolution) back to their original factory settings.

Restore Settings from NVM

This menu lets you restore your settings to those from the most recently saved NVM configuration.

1.1.11.3 Setup Mode Command Buttons

The Main Menu also includes buttons for Setup Mode commands, including:

- **Apply** - for applying current settings
- **Save** - for saving current settings in NVM
- **Undo** - for removing current settings
- **Exit** - for exiting the Setup Mode client
- **Help** - for displaying Help messages about Setup Mode selections
- **Restart** - for restarting the FX server
- **Reboot** - for rebooting the ViewStation, from the network or from flashPROM

1.1.11.4 Network Services

These network services support many simultaneous sessions and clients, including:

- Ethernet Driver to receive and demultiplex multiple packet types
- Ethernet Driver for IEEE 802.3, with auto-sensing
- Token Ring Communications for IEEE 802.2 packets
- Address Resolution Protocol (ARP) based on RFC826
- Reverse Address Resolution Protocol (RARP) based on RFC903
- BOOTP protocol based on RFC 951, 1015, 1048, and 1084
- Remote Configuration based on the BOOTP protocols above
- File Transfer Protocol (FTP) RFC959 for file transfer
- Trivial File Transfer Protocol (TFTP) RFC783 for file transfer
- Network File System (NFS) for file transfer
- X11R5 Font Service
- Domain Name Service based on RFC883 and RFC1035

- Internet Protocol (IP) RFC791 and MIL-STD-1777 as an end node only
- Transmission Control Protocol (TCP) RFC793 and MIL-STD-1778
- User Datagram Protocol (UDP) RFC768
- Internet Control Message Protocol (ICMP) based on RFC792
- Serial Line Internet Protocol (SLIP) based on RFC 1055
- Compressed SLIP protocol based on RFC 1144 (Van Jacobsen compression)
- Point to Point Protocol (PPP) based on RFC 1144, 1331
- SNMP protocol based on RFC 1065, 1066, 1067; 1156, 1157, 1158 (MIB II)
- XDMCP protocol based on MIT X11R5 X Window System

1.1.11.5 Terminal Emulator

The ViewStation provides two terminal emulations for Telnet remote terminal access protocol according to RFC854, RFC855, RFC857, RFC858, and MIL-STD-1782. HDSterm provides a VT320 emulation and the TTY terminal service provides an ANSI emulation.

HDSterm is a full-featured VT320 terminal emulator that is the user interface for Telnet connections. HDSterm is also used for the Serial and Console window interfaces. This is available with server code designations Vnn-D.hds.

A TTY terminal service is also available. It runs on top of the transport level services, including the user interface for Telnet . This is available with server code designations Vnn.hds.

1.1.11.6 Font Loading

Fonts are loaded from an X11R5 font server or dynamically loaded using FTP (RFC959) or TFTP (RFC783) from the font host specified in Setup Mode.

- File Transfer Protocol (FTP) based on RFC959
- Trivial File Transfer Protocol (TFTP) based on RFC783
- Network File Service (NFS)

1.1.11.7 Server-Resident Fonts

The ViewStation has 18 X11R5 fonts and 6 aliases built into the server. These are stored as compressed fonts and uncompressed internally on demand. The standard ViewStation fonts and aliases are:

"6x10.snf.Z	-misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1"
"6x12.snf.Z	-misc-fixed-medium-r-semicondensed--12-110-75-75-c-60-iso8859-1"
"6x13.snf.Z	-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1"
"8x13B.snf.Z	-misc-fixed-bold-r-normal--13-120-75-75-c-80-iso8859-1"
"8x13.snf.Z	-misc-fixed-medium-r-normal--13-120-75-75-c-80-iso8859-1"

"9x15.snf.Z -misc-fixed-medium-r-normal--15-140-75-75-c-990
-iso8859-1"

"10x20.snf.Z -misc-fixed-medium-r-normal--20-200-75-75-c-100
-iso8859-1"

"helvB12.snf.Z -adobe-helvetica-bold-r-normal--12-120-75-75-p-70-iso8859-1"

"courR18.snf.Z -adobe-courier-medium-r-normal--18-180-75-75-m-110-iso8859-1"

"term14.snf.Z -dec-terminal-medium-r-normal--14-140-75-75-c-80-iso8859-1"

"timB12.snf.Z -adobe-times-bold-r-normal--12-120-75-75-p-67-iso8859-1"

"olcursor.snf.Z -sun-open look cursor-----12-120-75-75-p-160-sunolcursor-1"

"olgl12.snf.Z -sun-open look glyph-----12-120-75-75-p-113-sunolglyph-1"

"luBS12.snf.Z -b&h-lucida-bold-r-normal-sans--12-120-75-75-p-79-iso8859-1"

"luRS12.snf.Z -b&h-lucida-medium-r-normal-sans--12-120-75-75-p-71-iso8859-1"

"cursor.snf.Z cursor"

"deccurs.snf.Z decw$cursor"

"decsess.snf.Z decw$session"

Standard aliases are:

"FILE_NAMES_ALIASES"

"variable -adobe-helvetica-bold-r-normal--12-120-75-75-p-160-iso8859-1"

"fixed -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1"

"-DEC-MENU-MEDIUM-R-NORMAL--12-120-75-75-p-70-iso8859-1
-adobe-times-bold-r-normal--12-120-75-75-p-67-iso8859-1"

"-DEC-TERMINAL-MEDIUM-R-NORMAL--18-180-75-75-m-70-iso8859-1
-adobe-courier-medium-r-normal--18-180-75-75-m-110-iso8859-1"

"-DEC-TERMINAL-MEDIUM-R-NORMAL--14-140-75-75-m-70-iso8859-1
-dec-terminal-medium-r-normal--14-140-75-75-c-80-iso8859-1"

These specifications are subject to change without notice.

1.1.12 HDS ViewStation Number Encoding

The ViewStation uses its Serial Number and its Part Number to provide important information to you in a convenient way. The ViewStation has a label on the left rear corner of the base containing both a serial number and a model number.

1.1.12.1 HDS ViewStation Serial Number

The ViewStation's serial number is composed of the date of manufacture and the last six digits of the ViewStation's Ethernet address. The number is made up as follows:

A b cc dddddd

where

A = major revision level (circuit board or components)

b = one digit to give the year of manufacture

cc = two digits to give the month of manufacture

dddddd = the lowest six digits of the Ethernet (hardware) address

For example, a serial number of D305c8161b would show a D revision level, a manufacture year of 3 (for 1993), a manufacture month of 05 (May) and the lowest six digits of the hardware address as c8161b.

Note that the full Ethernet hardware address is 12 digits long. The highest 6 digits are the vendor code, which is 008096 for HDS. These two fields are combined to make the entire Ethernet address. In this case the full address would be:

008096c8161b

This Ethernet address is needed for starting the ViewStation with BOOTP (other starting methods don't use it). If you're using BOOTP, looking at the serial number is an easy way to get the Ethernet address.

1.1.12.2 HDS ViewStation Part Number

The ViewStation Part Numbers contain information that defines the ViewStation's configuration. You can also get this information from the "hwconfig" command from the Console Window, discussed in Section 1.2.1.

Standard configuration is base, 4 MB DRAM memory, no flashPROM, no communications board, 110 AC power, IBM kybd., mouse, manual; with video setting, bit planes, monitor, and monitor cable as appropriate.

Component level descriptions of these basic configurations are given below.

The ViewStation is composed of a KIT, which contains a Base (with Circuit Board options), an Options group, and a Monitor. There is a separate kit for the PCMCIA board and its options. All of these elements are described by values encoded in the model number. The base model number is found on the serial number label located on the back left side of the base.

The KIT consists of four parts:

Circuit Board	"CB" specified by fields 1 through 9

Base	"BA" specified by fields 1 through 11 (includes CB)
Options	"OPT" specified by fields 13 and 14
Monitor	"MON" specified by field 15

Model Number and Character position:

```
1 2 3 4 5 6 7 - 9 10 11 - 13 14 15
```

positions 1-9 are for the Circuit Board

```
1 2 3 4 5 6 7 - 9 10 11 - 13 14 15
```

positions 1-11 are for the Circuit Board +Base

```
1 2 3 4 5 6 7 - 9 10 11 - 13 14 15
```

positions 13-14 are for Options

```
1 2 3 4 5 6 7 - 9 10 11 - 13 14 15
```

position 15 is for the Monitor

```
1 2 3 4 5 6 7 - 9 10 11 - 13 14 15
```

The model number consists of a series of fields with the field location containing values indicating the configuration for that unit. Hyphens are inserted to distinguish kit component groups. The model numbers use these selectors in their appropriate fields:

Field Location	Mnemonic	Configuration Entry
1, 2	**Product**	
	LX series	L1 = FXD, 16 MHz
	FX16	G1 = FXA / B / D, 16 MHz
	FX25	G2 = FXA / B / D, 25 MHz G series = 1024 wide / 8 bit or 1280 wide / 1bit
	FX16	J1 = FXB / C / D, 16 MHz J series = 1152 or 1280 / 8 bit
	FX25	J2 = FXB / C / D, 25 MHz
	Ultra	I3 = FXI, 33 MHz
	FXV	V2 = FXV, 25 MHz (Video)
		V4 = FXV, 33 MHz (Video)

The following table shows ViewStation monitors, with fields 3,4 for the video switch setting and field 15 for the monitor. It also lists monitor part numbers and the video cables needed or supplied.

The 3,4 field sets the circuit board video configuration (field 3 = Hex switch setting; field 4 = jumpers or other video options). Settings for monitors no longer shipped are included.

Video (3,4)	Mon. (15)	Name	HDS P/N	Vendor P/N	Cable
00	obs.	19C	MO-V19C-S	C/T-20	3
			MO-V19CN-S	CN-20	3
10	obs.	19MM	MO-V19-S	VCX 68KHz	2
20	7	17M	MO-17-S	DS1792	1
31	L	17CS	MO-V17CS	KDM1767E	A
	M	17CH	MO-V17CC-HR	1760LR (1280)	A
	8	17CT	MO-V17CT-S	TFS6706ETK	A
40	obs	15M	MO-V15-S	BDC1502	1
50	3	14CM	MO-V14C-S	FW6405ATK	A
	3 obs.	14CC	MO-V14CC-S	CDS4313	5
	3	14CS	MO-V14CS	KDM1466	A
	F	14CS2	MO-V14CS-2	KDM1466 (220V)	A
	S	14CSHSP	MO-V14Cs-HSP	KDM1466 (UL544)	A
	K obs.	15C	MO-SD5500	SD5500	A
	K	15CS	MO-V15CS	KDM1566	A
	R	15CS2	MO-V15CS-2	KDM1566 (220V)	A
	L	17C	MO-V17CC	1760LR	A
	L	17CS	MO-V17CS	KDM1767E	A
	L	17CS2	MO-V17CS-2	KDM1767E (220V)	A
	G obs.	19CM	MO-V19CM	HL6955SETK	A
52	J obs.	14M	MO-V14R	X-1457-HDS	A
60	obs.	19MS	MO-V19M-1	BDC2003	1
70 (or 71)	C	19CT	MO-V19CT-S	C/T-20D, A or H	3
	D obs.	19CI	MO-V19CI	HM4020D, HM4419D	3
	P	19CS	MO-V17CTN	447E	A
72		SW72		No sync on green	
83		VGA		640x480 VGA	
93		VGA8		800x600 SVGA	
F0	A obs.	19M	MO-V19M	F7A240Y5BF	2
	N	19M1	MO-V19M-1	BDC2003	1
F1	H	17H	MO-V17H	MO-V17H (1024)	H
	I	17H1	MO-V17H-1	MO-V17H-1 (1152)	H
	J	17H2	MO-V17H-2	MO-V17H-2 (1280)	H

Cable Description (P/N and legend):

A = Cable is attached or supplied
1 = WI-Mono DB15HD to BNC
2 = WI-3Mon DB15HD to 3 BNC GHV
3 = WI-RGB3 DB15HD to 3 BNC RGB
5 = WI-RBG5 DB15HD to 5 BNC RGBHV
H = WI-3Mon-17H DB15HD to 3 BNC GHV

This field sets circuit board options

5 Option

(none)	0 = No options
SKB	1 = Sun Type 4 or Type 5 keyboard
DUAL	2 = Dual Video Display (1024 wide x 2)
TR16	4 = Token Ring (16Mb ring speed)
TR4	5 = Token Ring (4 Mb ring speed)
SDK	8 = Serial Datakey port
PKB	P = PS/2 keyboard interface

This field sets the memory configuration. These values are permitted:

6 Memory	Setting	FX, LX series	Ultra	FXVideo
0M	0 = 0 MB	No	Yes	Yes
4M	4 = 4 MB	Std config	No	No
6M	6 = 6 MB	Yes	No	No
8M	8 = 8 MB	Yes	Std.config	Std.config
10M	9 = 10 MB	Yes	No	Yes
12M	A = 12 MB	Yes	No	Yes
16M	B = 16 MB	Yes	Yes	Yes
20M	C = 20 MB	Yes	No	Yes
24M	D = 24 MB	Yes	No	Yes
28M	E = 28 MB	Yes	No	No
32M	F = 32 MB	No	Yes	Yes
36M	G = 36 MB	Yes	No	No
40M	H = 40 MB	Yes	No	Yes
48M	I = 48 MB	No	No	Yes
52M	J = 52 MB	Yes	No	No
64M	K = 64 MB	No	Yes	Yes
68M	L = 68 MB	Yes	No	No
128M	M = 128 MB	No	Yes	Yes
132 M	N = 132 MB	Yes	No	No

Note: for ViewStation Ultra, the memory must use identical SIMM pairs, so available memory is 8, 16, 32, 64, or 128 MB only.

This field sets the number of video bitplanes

7 Bitplanes

1B	1 = 1 bitplane - 256K VR (monochrome)

8B	2 = 8 bitplanes - 1 bank VR (1024 only)
8B2	3 = 8 bitplanes - 2 banks VR (>1024)

8 Used as a separator hyphen only "-"

This field specifies the flashPROM board type

9 ROM

(none)	0 = Boot Prom only
FL	1 = Flash (16 Mb)

This field specifies the communications options

10 Ethernet Com

(none)	0 = No comm. board
CM	1 = Comm board

This field specifies the AC power setting

11 Power

120	1 = 120 VAC, std. monitor outlet
220	2 = 220 VAC, IEC outlet
HSP	3 or 5 = Hospital UL544, IEC outlet

12 Used as a separator hyphen only "-"

ViewStation Option Package Model Numbers

These are entered as a 2 digit value in fields 13 and 14, according to the following table (field 13 specifies the keyboard style, and field 14 is a sequential number for other option groups):

Keyboard values are specifically defined and used as the first digit in the Options table entry:

13 Keyboard

NKB	0 = No keyboard
KB	2 = 101+ style
DKB	4 = LK-401 style
SKB4	5 = Sun Type 4 keyboard
USDKKB	6 = US Datakey
SKB5	7 = Sun Type 5 keyboard
USCRKB	8 = US Cardreader
SKB5U	9 = Sun Type 5 UNIX keyboard
LXKB	L = Lexmark keyboard

The option packages are defined as follows:

14 Option Package

0	ACCBL only

1	Mouse, Mnl, ACCBL
2	Mouse, Mnl, ACCBL, RGBCBL
3	Mouse, Mnl, ACCBL, MONOCBL
4	Mouse, NoMnl, ACCBL
5	Mouse, NoMnl, ACCBL, RGBCBL
6	Mouse, NoMnl, ACCBL, MONOCBL
7	Mouse, NoMnl, ACCBL, 3MONCBL
8	Mouse, Mnl, ACCBL, 3MONCBL
9	Mouse, Mnl, NoACCBL
A	TKBL, Mnl, ACCBL
B	TKBL, Mnl, ACCBL, RGBCBL
C	TKBL, Mnl, ACCBL, MONOCBL
D	TKBL, Mnl, ACCBL, 3MONCBL
E	Mouse, Mnl, UKCBL
F	Mouse, Mnl, UKCBL, RGBCBL
G	Mouse, Mnl, UKCBL, MONOCBL
H	Mouse, Mnl, UKCBL, 3MONCBL
M	No Mouse, ACCBL, MNL
R	Mouse, Mnl, ACCBL, 17HMONCBL

Note that the following mnemonics designate the corresponding parts:

Mouse	
(none)	No mouse
MOUS	Standard mouse
TKBL	Trackball
Video Cable	
(none)	No video cable
MONOCBL	Mono video cable
RGBCBL	RGB video cable
3MONCBL	Mono 3-connector cable - GHV
17HMONCBL	Mono 3-connector cable - GHV
Power Cable	
NOACCBL	No power cable
ACCBL	US AC cable
UKCBL	UK AC cable
HSPCBL	Hospital cable

IECCBM	IEC monitor cable
Manual	
NOMNL	No manual
MNL	Manual

The monitor value is entered in field 15

15 Monitors

The ViewStation monitors are described in the video settings table above.

(no monitor)	0 = none

Examples:

For Model number = `G1F0081-111-28A`

Description: FX16 19M 8M 1B FL CM MON

Circuit Board

`G1` = FXA, 16 MHz circuit board

`F0` = video setting for FX19M, 76 KHz

`0` = no circuit board options

`8` = 8 MB of memory

`1` = 1 bitplane (monochrome)

Base

`1` = FlashPROM (both WM, 16 Mb)

`1` = Comm board

`1` = 120 VAC power

Options

`2`= 101+ style keyboard

`8` = Mouse, Mnl, ACCBL, 3MONCBL

Monitor

`A` = 19M 76 KHz - F7A240-Y5BF

HDS ViewStation: Internal Architecture

This chapter describes the ViewStation's internal architecture, its internal file system, and console shell commands used by the ViewStation.

Section 1.0 - HDS ViewStation

1.2 HDS ViewStation Internal Architecture

The HDS ViewStation uses a UNIX micro-kernel architecture for its internal operation. This micro-kernel allows the ViewStation to run client applications locally, to use standard UNIX ports and communications techniques, and to provide the same UNIX interface for user control.

The internal UNIX micro-kernel is controlled by a shell window called the Console Window. Its operation is described below.

1.2.1 ViewStation Console Shell Window

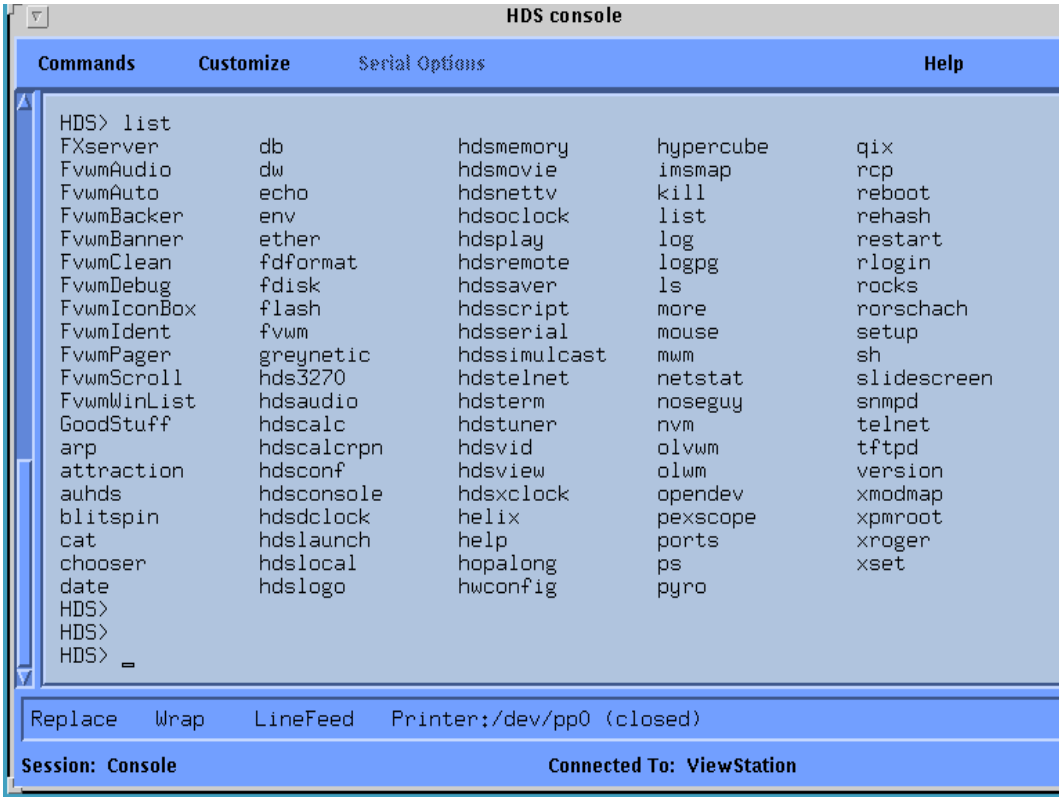
The ViewStation has a local client window, called the Console Window, which uses an HDSterm window as its display. The Console window may be displayed locally or remotely, as described below.

The Console Window has two basic functions:

- Executing commands on the ViewStation (like loading a file into FlashPROM). The easiest way to become familiar with the Console Window is to bring it up (click on its button on the Main Menu of Setup Mode) and look at its commands. The commands available from the Console window are displayed by typing "h" or "help" in the Console window.
- Displaying data or messages about the current ViewStation status.

The diagnostic and error messages are displayed automatically to track events on the ViewStation. The level of error messages (and their destination) can be changed by modifying your syslog.conf file. This is discussed in greater detail later in this chapter.

The contents of this diagnostic screen, and the commands supported, are specific to different firmware releases. Your screen may be slightly different. Console Window commands are added with each software release. You can get a full listing of available commands and HDSperx clients with the "list" command.



1.2.1.1 Access to the Console Window

The Console Window client can be displayed in three different ways; more than one window may be active at any time, so you can have simultaneous sessions. The ViewStation user might be looking at the message log while the system administrator is examining the ViewStation's network statistics from his remote ViewStation.

The three methods of Console Window access are:

- As a local display on the ViewStation, by clicking on the Console Window box in Setup Mode.
- From a remote location, by using **rlogin** or **rsh** (or **rcmd** or **remsh**, depending on your host's syntax) from the remote host to the ViewStation. This access can be controlled with the ViewStation's rhosts file, discussed in Section 1.11.2.
- From a remote ASCII terminal connected directly to the ViewStation's serial port.

Depending on your use of the Console Window, you can use one or more of these types of display to run commands on the ViewStation or analyze the ViewStation's behavior. For instance, if you want to see whether the fontpath is loading correctly, you can do it from the Console Window. These three types of display are discussed below.

1.2.1.2 Console Window as a Local Display

When you click on the Console Window box in Setup Mode, a window appears. This is the console shell of the ViewStation and accepts shell commands and displays the ViewStation's activity. It is an ordinary HDSterm window and can be repositioned or iconified with your window manager. You can display a list of supported commands by using "h" or "help".

1.2.1.3 Console Window from a Remote Host

You can open the Console shell from another network device by using an **rlogin** command to the ViewStation hostname. This login capability, either rsh or rlogin, are controlled by the ViewStation's rhosts security (available with HDSware 3.2), which is discussed in Section 1.11.2. The ViewStation reports all attempts to gain remote access.

On a remote device, either a host computer or another ViewStation, (or even on your own ViewStation) just open an xterm window on your host, then issue the command:

rlogin <ViewStation hostname> (or **rsh** or other syntax)

This puts the ViewStation console shell into the xterm window. This xterm window may be on your own ViewStation or from another device.

You can use **rsh** commands to the Console shell also, such as:

rsh hdsfx19 **nvm dump** > /tmp/nvmoutput.txt

This is a convenient way to generate a system configuration file. You can configure one ViewStation the way you want (font locations, nameserver settings, download files, etc.), and then simply dump that ViewStation's NVM to a file, which can then be used for all other ViewStations.

The **rlogin** and **rsh** commands are protected by normal system security procedures. If a configuration password is set, sensitive commands (such as burning flashPROM and displaying memory contents) to the Console window with the **rsh** process may be run only with password access. All commands may be password protected (see Section 1.11.1.3).

For example, if you open a Console window (locally or on a remote device), you can execute any of the unprivileged commands. But if you want to use a privileged command, such as burning a FlashPROM, you must first enter the password command, and enter the configuration password:

```
hds> password
hds> enter Setup Configuration password:
```

(note that the window prompt changes from hds> to HDS> - which indicates privileged mode)

```
HDS> <privileged command>
```

A system administrator might want to burn new firmware versions into all the ViewStations on the network from his remote location. This is possible using the remote shell commands and a list of host names and passwords. If you want to use the **rsh** command from a script to execute privileged commands in the Console window, you can follow this example:

```
rsh <fxname> sh -P < <password.script>
```

where <password.script> contains:

```
password
<fxname password>
<privileged command>
(such as: flash /hdssun5/bootfile/V30-D.hds)
exit
```

1.2.1.4 Diagnostic Messages from a Remote Terminal

You can connect a character-based terminal (a VT100 type) to one of the ViewStation's serial ports. The ports are normally configured for a printer, so you may need to use the terminal's printer port or use a null modem cable; check the pin assignments in Section 1.1.10. Set the communications settings as your terminal requires. Then select the "Console" entry for the Serial Port Use field in Setup Mode. This setting automatically opens the console shell of the ViewStation for the serial terminal. This method is necessary if you are checking the start-up behavior of the ViewStation; sometimes it can also be used to get information about the ViewStation if the display is hung or the ViewStation seems to be dead. As it starts up, the ViewStation displays various initialization messages, such as the keyboard and mouse state, the nameserver suffix, etc.

1.2.1.5 Console Window Security

The Console Window is a powerful tool which has the capability to make important modifications to the ViewStation's internal operations. There are two methods of controlling security on the Console Window, by limiting access to the Console Window with the ViewStation's rhosts security (see Section 1.11.2) and by controlling access to individual commands with password control.

ViewStation rhost Security

The ViewStation uses an internal rhosts file to configure the rsh and rlogin daemons and control access to the ViewStation. By default, this security is not used. This feature is available with HDSware version 3.2 and later.

The ViewStation's rhost security is set up by entering an rhosts file name in Setup Mode on the Remote Hosts menu or with the resource:

```
HDSsetup.rhosts.file: <filename>
```

in the configuration file. If no file is specified or the file is missing, rhosts security is not used. If the file is present but empty, no access to the ViewStation is allowed.

Passwords for Console Window Commands

Some Console shell commands are labeled "Privileged", such as burning new code into the FlashPROM, displaying the contents of memory, or killing local processes. These commands are treated differently and can only be used with password access. When you use these commands, you must enter the Setup Mode configuration password, if a configuration password has been set.

In addition to these local shell commands, all of the other ViewStation local clients and can have password protection extended to them on an individual basis by modifying the **passwdRequired** resource in the appropriate clients.config file. In this way all Console Window commands, even ordinary commands like **ls** and **pwd**, can be password protected. For example, the **opendev** command is password protected by entering this resource in the clients.config file as follows:

```
hds.client.opendev.passwdRequired:  true
```

Alternately, the configuration file resource:

```
HDSsetup.password_is_global: yes
```

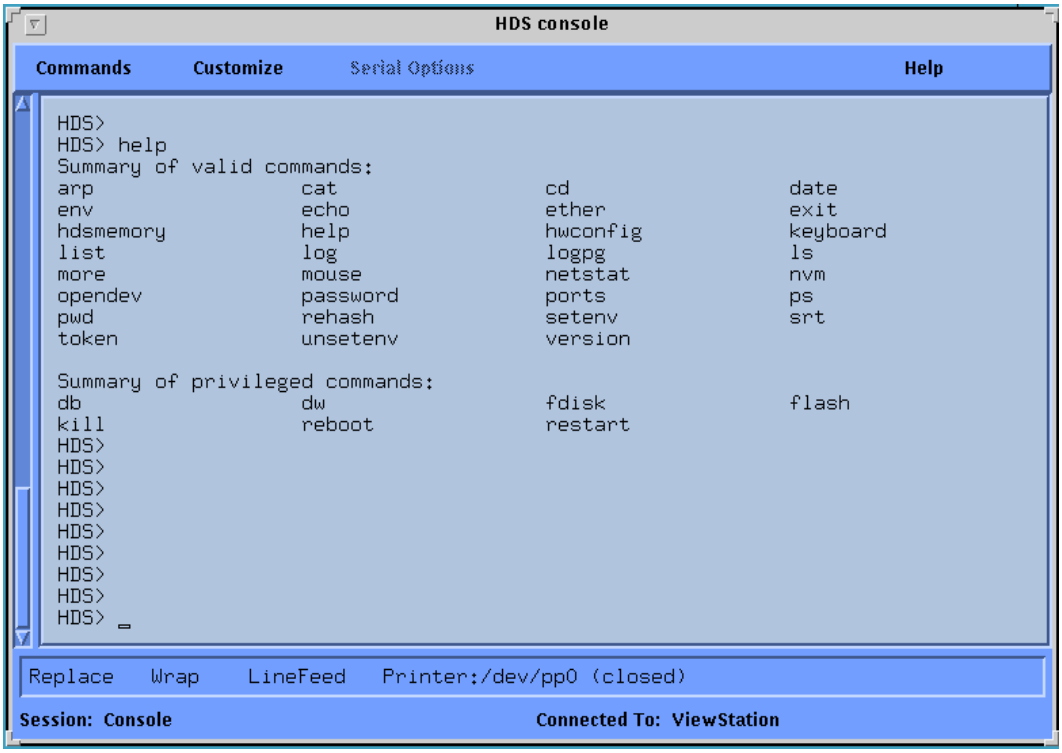
sets all Setup Mode values and all local clients to be password protected.

Note that the Help message screen which displays the available commands in partitions showing privileged and ordinary commands does NOT reflect changes you make in the password protection status. Refer to the listing of client attributes in the discussion of the clients.config file in Section 2.1 for details.

The **nvm dump** command does not display passwords, but simply reports <not displayed>.

1.2.2 Console Window Commands

The Console Window is really a shell which has access to the ViewStation's internal operation. It accepts a number of shell commands, shown on the help screen.



These commands are listed below and discussed briefly. Using the **help** command or entering the command followed by a "?" displays a brief help message showing the arguments the command accepts.

A more complete list, including local client applications, is available with the "list" command. This list includes the downloadable ViewStation clients as well as the shell clients.

1.2.2.1 Summary of Console commands:

arp	Display the Address Resolution Protocol table
cat	Displays files to the standard output
cd	Change current working directory (of the Flash File System)
date	Displays or sets the date and time
env	Display the HDSshell environment. This command is useful in analyzing the environment accessible to the ViewStation for a **rsh** command.
echo	Echo arguments to the standard output
ether	Display ethernet interface statistics
exit	Exit the Console Interface
hdsmemory	Starts the memory local client

help	Display help for given command `usage - **help** [ *command* ]` **h** and **?** also display the help messages
hwconfig	Displays the ViewStation's hardware configuration
keyboard	Reset the keyboard
list	List supported HDSshell commands including downloadable clients
log	Display all log messages
logpg	Display all log messages, one page at a time
ls	List the contents of a directory `usage - **ls** [ -1ACFLRdflqrs ] [ filename ... ]`
more	Displays a file one page at a time
mouse	Reset the mouse or trackball
netstat	Display network protocol statistics `usage - **netstat** [-r] [-m] [-s] [-tcp] [-udp] [-ip] [-icmp] [-all]` **-r** display routes **-m** display network buffer statistics **-s** display protocol statistics **-tcp** display TCP protocol statistics **-udp** display UDP protocol statistics **-ip** display IP protocol statistics **-icmp** display ICMP protocol statistics -all same as specifying every option
nvm	NVM commands `usage - **nvm dump** or **nvm save**` **dump** Display NVM Contents **save** Save settings in NVM
opendev	Provide direct access to a device. This is discussed in Section 1.2.6.2 in this chapter. `usage - **opendev** <*device_pathname*>` where <*device_pathname*> is either /dev/tty1, /dev/tty2, or /dev/pp0
password	Enter Setup Configuration password
ports	Display serial and parallel port statistics
ps	Display local task status
pwd	Display current working directory

rehash	Reread the Client Configuration Files specified in Setup
setenv	Modify environment variables
	`usage - **setenv** variable <value>`
srt	Display Token Ring Source Route table
token	Display Token Ring interface statistics
unsetenv	Remove environment variables
	`usage - **unsetenv** variable ...`
version	Display software versions

This list is subject to change. Many of these commands have command line arguments that affect their operation. Use help <command> for command syntax.

1.2.2.2 Summary of privileged commands

The following commands are privileged, that is, they cannot be used without supplying the Setup Mode password, if one has been assigned. All commands and clients can be defined as privileged, on an individual basis, by specifying the **passwdRequired** resource in the ViewStation's clients.config file.

From the prompt line of the Console Window, use the **password** command to enter the assigned password, then use the privileged command you want. When the password is entered successfully, the Console Window prompt changes from hds> to HDS> to indicate privileged mode operation. If your password entry fails, you cannot use these privileged commands.

db	Display memory using byte format	
dw	Display memory using word format	
fdisk	Formats the ViewStation's hard disk	
flash	Program Flash memory	
	`usage - flash server <filename>`	
	for example:	flash boot <*filename*>
	boot	Program Flash boot PROM
	files	[-rs] [-n] [filename]
		-rs Remove Flash server code image
		-n Display number of bytes available for Flash filesystem
	server	Program server code image into Flash
kill	Send a signal to a process, or terminate a process	
	`usage - **kill** [ -signal ] pid ...`	
reboot	Reboot ViewStation - reloads server code image	
restart	Restart ViewStation from current RAM server code image	

1.2.2.3 Privileged Commands in a Shell Script

If you want to use these privileged commands in a shell script, you must handle the password command carefully. For instance, if you want to use the "flash" command for all the ViewStations on a network, you could use one of these variations in your shell script:

Variation #1:

```
rsh <ViewStation name> sh -P << EOF
password
<the password>
flash files <path>
EOF
```

Variation #2:

Put in a file:

```
password
<the password>
flash files <path>
```

then use the "do" command with:

```
rsh <ViewStation name> sh -P < <
```

1.2.3 Console Window Messages

The messages available from the Console Window include things like the log file, network statistics, messages from HDSperx local clients, internal memory contents, status of the serial and parallel ports, and so on. They also includle messages from ViewStation clients. The messages include current activity messages, which might include the reading of a fontpath, loading a font, opening a client, etc. as well as any error messages from the host, or FTP, or the ViewStation itself. In many cases, the error messages indicate the nature of the problem; for instance, the error message "FTP Reply Error: 530 Login Incorrect" from FTP is sent when the FTP username has an invalid or empty password. Messages of this type are often deciphered for you by the ViewStation. The manual pages for each of these clients show the messages and point to the problems that generate them.

These messages are stored in a message log, which you can display using the "log" command. You can page through the logged messages with the "logpg" command. The messages are saved in a 64-line buffer so you can read them back, since they often come in large batches too fast to read. This log file buffer has a selectable size. You can change it with the resource:

HDSsetup.syslog.buffer.size

The message reporting level can be changed or these messages can be displayed to different devices, as discussed below.

1.2.3.1 ViewStation Error Message Reporting

The ViewStation has a great deal of flexibility in displaying error messages, both their level and the locations to which they are sent. These selections are made

with the syslog.conf file. This file is located in the config directory in the HDSware path. By default, syslog.conf is NOT read. If you want to use this file, you must enter the syslog configuration file location in Setup Mode in the Startup Options menu. This file is read as soon as you Apply changes (and no longer read when you remove it). This makes it easy to change your error reporting if you are having difficulty. You can also enter a resource setting for it in the ViewStation's configuration file, as follows:

HDSsetup.syslog.config.file: /FXUtilities/hds-fx/config/syslog.conf

In the syslog.conf file you specify the level of message reporting you want and specify a destination for them. The ViewStation uses eight levels of messages, from debug level to emergency. Note that specifying a level of error message sends that level and all messages more urgent than that level to the destination.

Message reporting levels:

emerg	major problem, system is unusable
alert	problem with immediate action required
crit	critical condition
err	error condition
warning	warning condition
notice	normal, but significant information
info	information on system (default level)
debug	debug-level messages on operations

Message destinations:

/dev/console	This sends messages to all ViewStation console shells, including the Console clients, all "rlogin" remote sessions, and all serial port diagnostic sessions. This is the default destination.
/dev/pp0	This sends messages to the ViewStation parallel port
/dev/tty1	This sends messages to the ViewStation's serial port 1 (RJ11)
/dev/tty2	This sends messages to the ViewStation's serial port 2 (DB25)
@hostaddress	This sends messages to the syslog daemon on the specified host. If needed, that syslog daemon can redirect the messages.

The syslog.conf file has an entry with the following syntax (note the TAB character):

*.<message level> <tab character> <destination>

The default entry for messages is:

*.info /dev/console

which sends messages from "info" level upward to the console window. If you want to set the level higher or lower, just use the desired level. If you want to send it to another destination, just change the destination entry. For example,

*.debug /dev/pp0

which sends all messages to the ViewStation's parallel port, or

*.err @hostaddress

which sends messages **err** and above to the syslog daemon on the host specified.

Note that any changes must be entered into the syslog.conf file; the filename and path must be entered in Setup Mode; and Apply changes selected in order for the changes to be effective.

You could also use the ViewStation configuration file to make these changes. In this case you would modify the syslog entry appropriately, and the ViewStation restarted to re-read the configuration file. You could also use

kill -1 <syslogd pid>

to restart the syslog daemon without restarting the ViewStation.

1.2.4 Printing from the Console Window

There are a number of methods of printing the Console window contents, depending on your current setup. If you are running from your ViewStation display in an HDSterm window, you can use the HDSterm utilities, including HDSterm's built-in print capabilities from its Commands menu. These print capabilities include Auto-print logging, printing the screen, etc. Refer to Section 2.2 for details.

Using the syslog.conf file, you can directly route error messages, at whatever level you specify, to one of the ViewStation's serial ports, its parallel port, or to a remote host. This is discussed in detail below.

From an rlogin session you can capture the entire contents of the Console window screen to a file by using the **script** utility. Start an xterm or HDSterm window on a remote host, then run the **script** program (your UNIX may not have this utility or may call it something else) before you start the Console Window in that xterm window by starting an rlogin session to the ViewStation. In your xterm window, start the script with the command:

script <*filename*> for example, **script** tmp/hdsproblem.log

rlogin <*fxname*>

which will start saving everything that follows. Then issue the rlogin command to connect to the ViewStation Console Window client. All of the activity in the window is saved to this file. To exit the Console Window client, type "exit" which closes the rlogin session, and then close the script file with "exit".

1.2.5 ViewStation Internal File System Structure

File names used on the ViewStation can include specification of a remote host and the directory structure on that host, as follows:

/<hostIP>/<path>/filename

For example:

/128.91.3.12/home/HDSware/hds-fx/image/V30-d.hds

which shows the host and path where the HDSware tape was loaded. Note the use of the leading slash. In this case

HDSware host = 128.91.3.12

HDSware path = /home/HDSware

Ordinary file management commands, such as cd, pwd, ls, etc. work in the ViewStation's Console window to show the ViewStation's internal file organization.

1.2.5.1 FXUtilities Abbreviation

There is a special case for host and path names. The HDSware host and HDSware path are abbreviated as "FXUtilities". This designation can be used instead of the full host and path name for that location.

From the example above,

FXUtilities = /128.91.3.12/home/HDSware

so you could enter:

/FXUtilities/hds-fx/image/V30-D.hds

or /HDSware/hds-fx/image/V30-D.hds

as an equivalent path and filename.

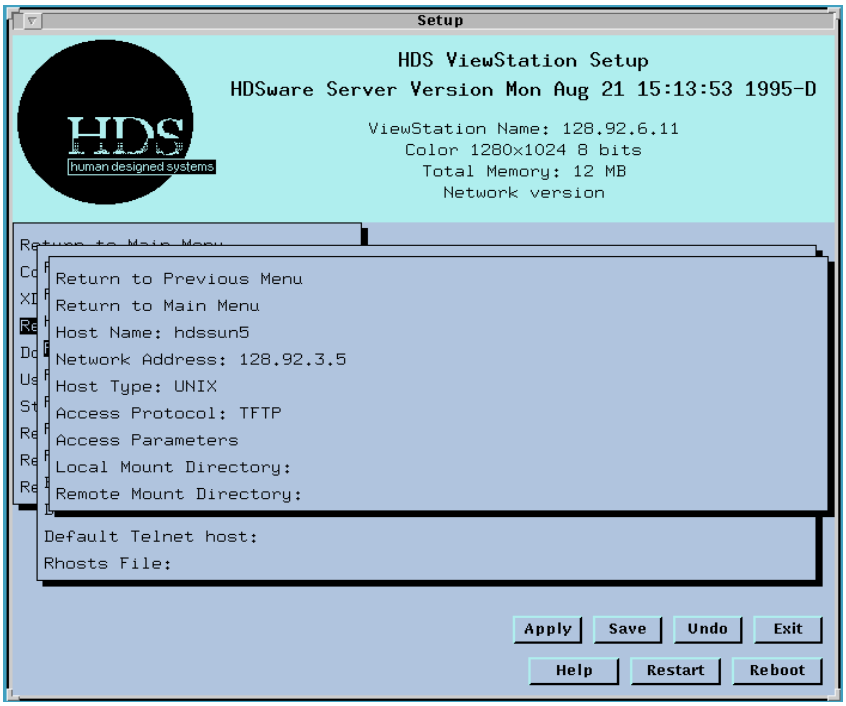
This technique is used in the ViewStation's configuration file for automatic selection of ViewStation paths once the HDSware information is specified.

With HDSware 3.1, a "BOOTPhost" abbreviation is also recognized if you are using BOOTP. This mount point uses the BOOTP host's IP address with the hd (home directory) entry in your bootptab file.

In the Console Window, the "ls" command shows the current mount points used by the ViewStation, including HDSware, FXUtilities, and BOOTPhost.

1.2.5.2 ViewStation Mount Points

The HDS ViewStation uses an internal file system that closely resembles the NFS file system, using mount points to establish local and remote paths. Local and Remote Mount Directories are entered in the Remote Hosts menu of Setup Mode for each remote host. Individual paths, such as for fonts or RGB color definition files, are accessed by means of the information the mount point provides.



The remote host address is given by the Network Address. If the Network Address field is empty, then the Host Name is used for the remote host address. Domain Name addresses are resolved by your Domain Name Server (or NIS).

1.2.5.3 Local Mount Directory

A Local Mount Directory defines an internal ViewStation location for the server to look for things. The file system maintains an internal list of these Local Mount Directories and searches that list when it opens a file. (The list contains all the specified internal file locations, like HDS$$ROM, Host Names, Local Mount Directories, and local ports, such as */dev/tty1*, etc.). If the Local Mount Directory is specified, then it is used as the mount point. If no Local Mount Directory is specified, the Host Name is prepended with leading slash and it becomes the mount point.

1.2.5.4 Remote Mount Directory

A Remote Mount Directory defines a location on the Remote Host as a starting place for the ViewStation to look for things. The Remote Mount Directory is typically a directory location where the HDSware tape was loaded or some special files are located. The Mount Point is the main directory, with subdirectories specified by the path itself. The Remote Mount Directory is prepended to the path (after the mount point is stripped off) to give a complete file location.

1.2.5.5 Mount Directory Example

For example, if the Local Mount Directory for Host1 specifies "/fonts" (the leading "/" is necessary), then a Mount Point is created in the ViewStation. The fontpath is specified as "/fonts/100dpi,/fonts/75dpi,/fonts/misc". The Remote Mount Directory for Host1 is "/usr2/lib/X11/HDS/hds-fx/fonts". When a font is requested, the ViewStation server finds the /fonts mount point and the fontpath, strips off the mount point, prepends the Remote Mount Directory, and then opens a connection to Host1 to the directory "/usr2/lib/X11/HDS/hds-fx/fonts/100dpi", etc.

This file structure can provide extra functionality for using the ViewStation on your network. If you want to have a backup font host, you could specify Local Mount Directories "/fonts" on Host1 and also on Host2. The ViewStation's mount point would first go to Host1, and if that file open failed (for whatever reason), the ViewStation would go on to Host2, and so on. The Remote Mount Directory on Host1 could be different than the Remote Mount Directory on Host2, but as long as those were correctly specified on the Host1 and Host2 menus, they would be found.

Another possible application is the handling of special fonts, such as the OPEN LOOK fonts, or special fonts for a word-processing application, that you wouldn't want to have on every network host. The following table shows the entries needed for specifying three font directories on three different hosts.

The Default Font Path on the User Preferences menu is "/fonts1/100dpi,/fonts2/Xol,/fonts3/HDS-fonts":

Host 1:

specifying

Local Mount Directory: /fonts1

Remote Mount Directory: /usr/lib/X11/hds-fx/fonts

results in

Full Font Path: /usr/lib/X11/hds-fx/fonts/100dpi

Host 2:

specifying

Local Mount Directory: /fonts2

Remote Mount Directory: /home/openwin/MITXol/fonts

results in

Full Font Path: /home/openwin/MITXol/fonts/Xol

Host 3:

specifying

Local Mount Directory: /fonts3

Remote Mount Directory: /usr3/bin/Asterix/fonts

results in

Full Font Path: /usr3/bin/Asterix/fonts/HDS-fonts

Note that if a file open to a particular font path fails, that font path is removed from the current font path, and the remaining paths are used. The failed font path is used again only after a ViewStation restart or a font path rehash.

1.2.6 Access to ViewStation Ports

The ViewStation's Ports, one parallel and two serial, are generally accessible by any device on the network with conventional naming techniques. They are designated:

parallel port = /dev/pp0

serial port 1 (RJ11) = /dev/tty1

serial port 2 (DB25) = /dev/tty2

Other ports are named the same way:

audio port = /dev/audio

1.2.6.1 The opendev Command

The "opendev" command available from the ViewStation's Console window is the entry point for a generalized bi-directional read/write capability for the ViewStation's serial and parallel ports (the parallel port is write-only). This capability is described more fully below.

1.2.6.2 Remote Access to ViewStation Ports

It is sometimes desireable to permit a process running on a host to have direct access to one of the ports on the ViewStation; in other words, letting the remote ViewStation port act as a local port on the host. One situation where this would be useful might be additional user input from a serial device, such as a digitizer tablet, modem, or barcode reader. Other devices, such as badge readers or a security key, could also be connected to the ViewStation serial ports.

The ViewStation server has a command called "opendev". This command opens a device on the ViewStation, specified as an argument to the command, for bi-directional communication. The argument to the **opendev** command may be any device driver installed at a /dev/xxx node.

The command syntax is as follows:

rsh <*fxname*> opendev [*/dev/ttyx*]

which executes a remote shell command to the ViewStation to execute the **opendev** command with the specified device. This means that standard input from the **rsh** TCP port is written to */dev/ttyx*, and data from */dev/ttyx* is written as output to the **rsh** TCP port. The ViewStation has two serial ports, designated */dev/tty1* (an RJ11, RS-232 port) and */dev/tty2* (a DB25, RS-232 port), and one parallel port designated */dev/pp0* (DB25 Centronics) for local printing.

You could also use the **rsh** command in conjunction with a **stty** command to give transparent access to the port. For example, typing a command line directly into a modem might use:

```
stty raw -echo; rsh <fxname> opendev [ /dev/tty x ]
```

The **rsh** command may be used either in a shell script or in a compiled program. The shell syntax is given above, and would normally be activated as a sub-shell to allow redirection of the file descriptors created by the **rsh** command. Terminating the **rsh** command would terminate the process.

1.2.6.3 Usepty Script

The **usepty** script to use a pseudo-tty, provided on the HDSware tape in the */hds-fx/contrib/usepty/src* directory, is also useful here. A typical command line for a pseudo-tty for port 1 might look like:

```
usepty /dev/ptyrb rsh <fxname> opendev /dev/tty1 &
```

The **usepty** process will repeatedly monitor the pseudo-tty port and wait for a process to open on the corresponding slave side. When the slave side is opened, **usepty** will execute the **rsh** command to open the port on the ViewStation. When the slave side is closed, the **usepty** process will resume monitoring the pseudo-tty port.

1.2.6.4 Implementing "opendev" in a C Program

To use this command in a C program, the typical technique would create two pipes, one for input and one for output, then fork a child process and let the child process execute the **rsh** command. The sample file below illustrates this technique (". . ." indicates omitted code segments of the user's choice):

```
...
char *cmd = RSH_PATH;          /* "/usr/ucb/rsh" or "rsh" */
char *rem_host = "hds-fx";           /* example only */
char *opendev = "opendev";
char *rem_device = "/dev/tty1"; /* example only */
int  pvin[2];
int  pvout[2];
...
pipe (pvin);          /* pipe to read from remote device */
pipe (pvout);          /* pipe to write to remote device */
if (fork() == 0) {
  dup2 (pvin[1], 1); /* stdout goes to pipe pvin */
  close (pvin[0]);
  close (pvin[1]);
  dup2 (pvout[0], 0); /* stdin comes from pipe pvout */
```

```
    close (pvout[0]);
    close (pvout[1]);
    execlp (cmd, cmd, rem_host, opendev, rem_device, (char *) 0);
    exit(1);          /* exec() failed */
    }
  close (pvin[1]);
  close (pvout[0]);

  /* Now can use pvin[0] for reading, pvout[1] for writing */
  ...
  cc = read (pvin[0], buffer, buf_size);
  ...
  cc = write (pvout[1], buffer, count);
  ...
```

1.2.7 ViewStation Flash File System

The HDS ViewStation flashPROM board or PCMCIA devices (2.5 or 5.0 MB memory cards or 20 or 42 MB hard disk drives) can be used as a local Flash File system. This can be a convenient feature in many situations, such as operating the ViewStation at a remote location where loading fonts or files from the network might be difficult.

1.2.7.1 Operation with the flashPROM Board

In normal operation, the ViewStation's server code is stored in the flashPROM memory. Depending on the server code file used, there may be some memory on the flashPROM board that is not used. This residual memory can be used for storage of other files, such as fonts, configuration files, local clients, and so on.

It is possible to use the entire flashPROM memory to store files of this type by specifying that the ViewStation load its server code from the network. This is generally contrary to the intent of using the flashPROM memory to avoid network downloading, but may be useful in some situations.

1.2.7.2 Operation with PCMCIA Devices

With the PCMCIA device installed, the Flash File system operates the same way utilizing, of course, the larger storage capacity.

The use and configuration of PCMCIA devices is described in detail in Section 1.10.

1.2.8 ViewStation Startup Files

The ViewStation has two files you can use to start processes on the ViewStation.

1.2.8.1 Server-reset Command File

The server-reset file is run by the ViewStation whenever the server is reset. This would include starting up, manual reset commands, Xdm resets, and so on. This is a good place to start local processes. This is generally preferred over the startup command file.

You use the server-reset file by listing the file location in the configuration file with the following resource:

HDSsetup.server_reset.command.file: <file name>

You can also set this file name and path in the Startup Options menu of Setup Mode.

Processes in this file will be started on the ViewStation before XDMCP starts, so it is a good place to use ViewStation clients like xmodmap, so you could remap your keyboard before your login. This is especially useful for users with multinational keyboards and need special keys for their username and password.

1.2.8.2 Startup Command File

The ViewStation's startup command file is run when the ViewStation starts up. This file specific for a ViewStation, not a user. User resources, if specified, will override these settings when the user logs in through this ViewStation. For this reason, it is used for specific ViewStation settings, such as for a print device, or color settings.

You use the startup command file by listing the file location in the configuration file with the following resource:

HDSsetup.startup.command.file: <file name>

You can also set this file name and path in the Startup Options menu of Setup Mode.

For example, suppose you have a ViewStation gray-scale monitor and you find that the HDSterm client's background color (lightsteelblue) doesn't give sufficient contrast with the text in the window. This Startup Command file would be a good place to specify a new color resource for the HDSterm. This would be specific to the gray-scale monitor but would not affect other ViewStations or other clients. Similarly, it would not be linked to your username and wouldn't set this color change when you logged in from a ViewStation with a color monitor. This resource is:

HDSterm*screen.Background: ivory

In another example, the Startup file might specify a printDevice resource to indicate the printer connections on this ViewStation.

HDSterm*screen.printDevice: /dev/tty2

A sample startup command file typically contains a reference to an Xenvironment file, which in turn contains the actual ViewStation client resources.

The Startup Command file name should be:

/FXUtilities/hds-fx/config/FXstartup.sh

The FXstartup.sh file might contain (using C shell syntax for this example):

```
setenv XENVIRONMENT /FXUtilities/hds-fx/config/FXresources
```

The FXresources file would contain resources for the ViewStation local clients, such as:

```
# These are HDSterm resources
HDSterm*screen.printDevice: /dev/tty2
HDSterm*screen.Background: ivory
# These are FX local Mwm resources
Mwm*keyboardFocusPolicy: pointer
Mwm*configFile: /usr/lib/X11/app-defaults/.mwmrc
# These are FX local OLWM resources
OpenWindows.SetInput: followmouse
OpenWindows.WorkspaceColor: White
```

HDS ViewStation: HDSware

This chapter describes HDSware, the ViewStation's software distribution, which includes the server software, fonts, local clients, and so on.

Section 1.0 - HDS ViewStation

1.3 HDSware

HDSware is HDS Network Systems' X terminal software and operating environment. One copy of HDSware is needed at each site where ViewStation terminals are used. HDSware includes a number of important files needed to operate HDS ViewStation terminals. In addition, HDSware includes HDSperx, a set of local X client applications designed to run in an HDS X terminal.

HDSware is distributed on a local hard disk drive or a CD-ROM or a tape that the system administrator should load onto an appropriate host at each site. Each ViewStation should enter this host address and HDSware location in their "HDSware" menu in Setup Mode (or in the remote configuration file, or in Monitor Mode).

1.3.1 HDSware Software Formats

The HDSware tape is distributed on a variety of media, CD-ROM, tapes and diskettes. The CD-ROM distribution contains the HDSware files in loadable directories or as a mountable filesystem. The distribution on tape consists of a **tar** file which contains a number of directories to be downloaded. The diskettes contain compressed tar files for the fonts and individual compressed tar files for the utilities. The HDSware tape is normally available on QIC150 format cartridges. Other formats, like 4mm and 8mm DAT, QIC24, TK50 Ultrix or TK50 VMS, 9-track tape, Sun, SCO, and MS-DOS diskette formats, etc., are also available.

1.3.1.1 HDSware on an Internal Hard Disk Drive

HDSware is also available on an internal hard disk drive available as optional equipment on a ViewStation. In this case, only one ViewStation per site would need this HDSware installation. The HDSware host for other ViewStations would be the hard disk ViewStation's IP address and the HDSware directory would be */dev/flash*. The other ViewStations would get their server code, fonts, and clients from the ViewStation's hard disk drive instead of a host computer's disk.

Note that there are two relevant configuration files, *config.fx.dsk* for the hard disk ViewStation and *config.fx* for other ViewStations. In the first case (the hard drive ViewStation), the HDSware host would be the IP loopback address, 127.0.0.1, and the path would be */dev/flash*; in the second case (other ViewStations), the HDSware host would be the hard disk ViewStation's IP address and the path would be */dev/flash*.

1.3.1.2 Premium Software Service and Software Upgrades

New software versions and feature enhancements for the HDS ViewStation are distributed on the HDSware CD-ROM. Customers who have ordered the Premium Software Service receive automatic distribution of the HDSware CD-ROM. HDSware is also available to Premium Software Service customers from the Internet, at *http://www.hds.com* (a password is required for downloading HDSware - this password can be obtained from your Customer Care representative).

Customers who have not subscribed to this service can receive a no-charge HDSware upgrade for one year after purchase of their HDS ViewStation.

There is a charge for upgrades on formats other than CD-ROM. For further information you can call the HDS Service Department at 800-762-0357 or email at *info@hds.com*.

1.3.2 HDSware CD-ROM Format

The HDSware CD-ROM can be used either as a mounted file system, or as a medium to transfer the files onto your own disk file system. Both options are discussed below.

1.3.2.1 HDSware CD-ROM File Naming

The HDSware CD-ROM conforms to ISO 9660 file naming conventions. You should be aware that some file names may be different, or have different extensions because of this naming convention. You may need to notify your operating system that the CD-ROM is using ISO 9660 filenames (or High Sierra conventions, an earlier version of ISO 9660). Refer to the man pages of your "mount" command to see how your operating system handles ISO 9660 names (some examples are given below).

1.3.2.2 Mounting the CD-ROM

The HDSware CD-ROM can be used as a file system (read-only). This file system uses the "mount" command, as follows:

mount <*CD-ROM device name*> <*mount point*>

The device name varies with each operating system and environment (some examples are given below), and the mount point is a directory that becomes the parent directory of the CD-ROM file system, usually named */cdrom*. Note that the "mount" command lines usually have some designation for the ISO 9660 naming, whether "hsfs" (for High Sierra) or "cdrfs", or something similar; with more recent OS versions this may be optional.

For SunOS 4.1.3

```
/etc/mount -r -t hsfs /dev/sr0 /cdrom
```

For Solaris 2.3

```
/etc/mount -r -F hsfs /dev/dsk/c0t6d0s0 /cdrom
```

For IBM RS/6000 AIX 3.2

```
/etc/mount -r -v -p /dev/cd0 /cdrom
```

For HP 700 HP-UX

```
/etc/mount /dev/dsk/c201d2s0 /cdrom -r -t cdfs
```

For SCO systems

```
/etc/mount -r -f HS /dev/cd0 /cdrom
```

DECstation Ultrix 4.x

```
/etc/mount -t cdfs /dev/rz3c /cdrom
```

For OSF/1 for DEC Alpha

```
/usr/sbin/mount -t cdfs -o noversion /dev/rz3c /cdrom
```

For SGI IRIX 5.2

```
/etc/mount -o ro,notranslate -t iso9660 /dev/scsi/sc0d510 /
cdrom
```

Consult your man pages or your system administrator for device designations and directories for your systems.

The mounted CD-ROM can be read as a normal file system.

1.3.2.3 Loading files from the CD-ROM

The HDSware CD-ROM contains an installation script, called "INSTALL.CD", to manage the selection and installation of HDSware files from the CD-ROM.

Login as root user. Insert the CD-ROM into the ROM drive; check the device name (the examples above are typical, but may vary on your system). Mount the CD-ROM and run the installation script, as below.

1.3.2.4 HDSware Installation Script

You can install HDSware easily with a script available on the CD-ROM or tape. The HDSinstall script lets you select which files to include on your host installation.

CD-ROM Installation

From the CD mount directory, type: INSTALL.CD

Tape Installation

Insert the tape in your tape device (for other types of media, use your appropriate devices). Then type (be careful with the syntax):

```
tar xvf <device name> HDSinstall
```

which finds the HDSinstall script; it is the first file on the tape. The **tar** command continues to search through the entire tape, which may take several minutes. Once the screen reports that HDSinstall is loaded, you can discontinue the search and exit the **tar** process with *Ctrl-C* to save time. Then start the script by typing:

```
./HDSinstall
```

This starts the HDSinstall script. This is a Bourne shell script and should run without modification. There are explanatory screens and a menu to select installing the entire tape or only parts of it. There are suggestions for files to include or omit.

When you've selected all your files, the HDSinstall script prompts for your HDSware directory and the name of your tape device, and then installs your selections automatically. If you don't use the HDSinstall script, you can load the tape manually. Suggested procedures are outlined in the last section of this note.

1.3.2.5 HDSware CD-ROM Contents

The HDSware software contains server code files, fonts, client application programs, utility programs, font management programs, etc. as well as a set of postscript files containing the *HDS ViewStation System Administrator's Guide*.

The CD-ROM contains a number of directories for each of these groups of files; your operating system may have changed these file names based on its ISO 9660 or High Sierra formatting. The HDSware tape distributions may have some differences in content and filenames.

HDSware CD-ROM Contents (Version 3.1)

./INSTALL.CD

./Vnn.FFS;1 ***(flashfilesystem file for disk loading)***

./HDSWARE.TAR;1

./HDSFX/VERSION;1

./HDSFX/BIN

HDS clients, such as window managers and HDSperx

./HDSFX/CONFIG

Sample configuration files

/LPDSUPPO *(LPD print spooler)*

/LPSUPPOR *(LP print spooler)*

CLIENTS.CON;1 *(configuration for HDSperx clients)*

CONFIG.FX;1 *(the FX configuration file)*

CONFIG.SYN;1 *(a config file showing syntax)*

SYSLOG.CON *(the syslog.conf file for error reporting)*

./HDSFX/CONTRIB

Other contributed files provided for the user's convenience

/ BOOTP - *source code and BSD and Sys V executables*

/ HDSRUNWM - *utility program to launch local WM*

/ NETAUDIO - *an alternate way to play audio files*

/ SNFTOBDF - *utility program for generating bdf font files*

/ USEPTY - *program for pseudo-terminal operations*

/ XMKMF - *script to make a makefile from an Imakefile*

/ XRSH - *utility program for Xrsh*

/ XSCOPE - *source code and Sun4 executables*

./HDSFX/DOCS

/ SYSADMIN

Postscript files of the HDS ViewStation System Adminstrator's Guide

./HDSFX/EXTENSIO

X11R5 extensions downloadable into the FX server

./HDSFX/FONTS

Font directories - a main font directory, with subdirectories:

/ 100DPI (X11R5 fonts)

/ 75DPI (X11R5 fonts)

/ MISC (X11R5 fonts)

/ XOL (MIT X11R5 OPEN LOOK fonts)

/ ASTERX21 (Applix Aster*x v2.1 fonts)

/ ASTERX30 (Applix Aster*x v3.0 fonts)

/ WP51 (special WordPerfect v5.1 fonts)

/ LOTUS123 (special Lotus 123 fonts)

/ SOFTPC (special SoftPC fonts)

./HDSFX/IMAGE

Server code files - server files in different configurations

README

/ VnnD.HDS;1 *(server code to download clients)*

/ Vnn.HDS;1 *(no download capability in server code)*

/ VnnBOOT.BIN;1 *(bootprom file)*

./ HDSFX / IMAGE_U

These same image files in uncompressed format (.UHD extensions)

./HDSFX/INCLUDE

Include files needed for some compilations

./HDSFX/LIB

Library directory - libraries needed for HDS clients

/SHARED

/APPDEFAU

/BITMAPS

./HDSFX/MAN

Manual pages for the HDS clients in several formats

./HDSFX/RGB

Sample rgb.txt stored in ViewStation memory

./HDSFX/UTILS

Utility program directories, with two subdirectories

/SRC *(Source code with appropriate Makefiles, etc.)*

/SRC/HDSFC960

/SRC/BDFTO960

/SRC/GETBDF

/SRC/HDSMKDIR

/SRC/HDSCRYPT *(for password encryption)*

/SRC/FLASHFIL *(scripts to manage flash file system)*

/EXECS *(executable HDS utility programs above)*

/EXECS/SUNOS4

/EXECS/SUN_SOLA

/EXECS/VAX_ULTR

/EXECS/SCO386

/EXECS/HP700

/EXECS/DG_AVIIO

/EXECS/IBM_RS60

Other utilities and sample files

These files require approximately 60 MB of storage space (or less, depending on the directories you choose to load), so you should allocate sufficient space for them.

1.3.3 HDSware Tape Contents

The HDSware tape contains server code files, fonts, client application programs, utility programs, font management programs, etc. The filenames are in UNIX for-

mat rather than the ISO 9660 format of the CD-ROM. The tape itself consists of a **tar** file which contains a number of directories. The directories are shown below with their subdirectories.

HDSware Tape Contents

./HDSinstall

./Vnn.ffs (flashfilesystem file for disk loading)

./hds-fx/image

Server code files - FX server files in different configurations

README

/Vnn-D.hds (server code to download clients)

/Vnn.hds (no download capability in server code)

/Vnn-bootprom.bin (bootprom file)

./hds-fx/image_u

These same files in uncompressed format (.uhds extensions)

./hds-fx/fonts

Font directories - a main font directory, with subdirectories:

/100dpi

/75dpi

/misc

/Xol (MIT X11R5 OPEN LOOK fonts)

/Asterx2.1 (Applix Aster*x v2.1 fonts)

/Asterx3.0 (Applix Aster*x v3.0 fonts)

/wp5.1 (special WordPerfect v5.1 fonts)

/Lotus123 (special Lotus 123 fonts)

/SoftPC (special SoftPC fonts)

./hds-fx/bin

HDS clients, such as window managers and HDSperx

./hds-fx/lib

Library directory - libraries needed for HDS clients

/shared

/app-defaults

/bitmaps

./hds-fx/utils

Utility program directories, with two subdirectories

/src (Source code with appropriate Makefiles, etc.)

/src/hdsfc960

/src/bdfto960
/src/getbdf
/src/hdsmkdir
/src/hdscrypt (for password encryption)
/src/flashfilesys (scripts to manage flash file system)
/execs (executable HDS utility programs above)
/execs/SUN4
/execs/VAX_Ultrix
/execs/sco386
/execs/HP700
/execs/DG_AViiON
/execs/IBM_RS6000
Other utilities and sample files

./hds-fx/rgb

Sample rgb.txt stored in ViewStation memory

./hds-fx/man

Manual pages for the HDS clients in several formats

./hds-fx/config

Sample configuration files
/lpd-support (LPD print spooler)
clients.config (configuration for HDSperx clients)
config.fx (the FX configuration file)
config.syntax.fx (a config file showing syntax)
syslog.conf (the syslog.conf file for error reporting)

./hds-fx/docs
/sysadmin (Postscript files of the *System Administrator's Guide*)

./hds-fx/include

Include files needed for some compilations

./hds-fx/contrib

Other contributed files provided for the user's convenience
/bootp - source code and BSD and Sys V executables
/xscope - source code and Sun4 executables
/xmkmf - script to make a makefile from an Imakefile
/snftobdf - utility program for generating bdf font files
/usepty - program for pseudo-terminal operations

/xrsh - utility program for Xrsh

./hds-fx/extensions

X11R5 extensions downloadable into the FX server

These files require approximately 30 MB of storage space (or less, depending on the directories you choose to load), so you should allocate sufficient space for them.

1.3.4 Automatic ViewStation Configuration

When the HDSware is loaded, start your ViewStation and go into Setup Mode. Select "HDSware" from the Main Menu. There are three entries to make for HDSware automatic configuration.

- Set your ViewStation's IP address
- Set the host IP address where you loaded the HDSware
- Set the path to the HDSware directory

The ViewStation stores this information in its configuration file. It creates an abbreviation for "*/hostIP/path*" as "HDSware" or "FXUtilities". The configuration file entries for HDSware paths are then */HDSware/hds-fx/fonts*, and so on. In this way, the ViewStation configures itself automatically as soon as it knows the location of the HDSware software.

Suppose you created a directory for the software called "HDSware" in the following path: */usr/lib/HDSware*. The ViewStation uses this as a mount point and creates full paths from it; such as a fontpath equal to: */hostIP/usr/lib/HDSware/hds-fx/fonts*, and so on for other variables.

Note that these are equivalent paths to the ViewStation:

```
/hostIP/usr/lib/HDSware/hds-fx/fonts/100dpi
/FXUtilities/hds-fx/fonts/100dpi
```

and that the second entry is the default entry in the configuration file, where it can be used as soon as "HDSware" or "FXUtilities" is defined.

This automatic configuration of the ViewStation depends on two things: first, the correct specification of the host and path, and second, the correct operation of the file transfer protocol (TFTP by default) used to move the file.

When you restart the ViewStation, it reads the configuration file just created and enters these paths in Setup Mode so these important settings are made for your ViewStation automatically. You can save these values in NVM by clicking on the Save button.

At this point, you can make other settings important for your system, such as communication gateways, domain name servers, your list of remote hosts, etc. in the appropriate Setup Mode fields.

1.3.5 HDSware Upgrades

If you have ROM storage, you can load the new server and bootprom code into your FlashPROM with the Console window command:

flash server *</hostIP/path/filename>*

flash boot *</hostIP/path/filename>*

These procedures are discussed in more detail in Section 1.10.

Upgrading a hard disk drive takes a few more steps. First, make the file containing the HDSware 3.0 files you want. A sample file is:

./hds-fx/utils/src/flashfilesys/disk.files

Use this text file (with your changes) to prepare the disk file. Keep track of the size of the files on your list and make sure there is room on the disk for them. Remember to make any desired additions or deletions to the file list, or changes to your individual *config.fx.dsk* and *clients.config.dsk* files before you put them on the disk.

When your file is complete, prepare the disk file with the command:

flashfilesys <*text_filename*> <*disk_filename.ffs*>

For example, with Version 3.0.3, the file system file might be V303.ffs.

This reads the text file, finds the actual files listed in it, and then puts them all together into a single disk file. Make sure you have enough space on your host to create this file. Note that a complete .ffs file, with default settings, is included on the HDSware CD-ROM.

Then load the disk file onto the hard disk with the command:

flash files <*disk_filename.ffs*>

1.3.6 Loading the HDSware Tape Manually

These instructions describe loading HDSware on different systems without using the HDSinstall script. The instructions may have to be modified slightly to correspond to the requirements of your system. Use them as a guide, particularly noting differences such as device designations. The crucial items to watch for are that the file transfer is done in binary format and that the correct destination directories are created.

1.3.6.1 Cartridge tape handling

- **QIC150 tape handling** (Unix tape format):

  On your host system, move to the destination directory for your utility programs and font files. Then issue the **tar** command to extract the files from the tape:

  tar xvf */dev/rst8* (use the device name appropriate to your system)

  The files will be read into the current directory as individual subdirectories. Note also that the fonts.dir and fonts.alias files are present in the font directories.

  Some systems, such as HP and SGI, sometimes require byte swapping of the data. The tar command prompts you for the byte swapping option.

- **DEC TK50 cartridge tape** (Ultrix systems):

On your host system, move to the destination directory for your font utilities and font files. Then issue the tar command to extract the files from the tape:

tar xvf /dev/rmt0h

- **DEC TK50 cartridge tape** (VMS systems):

  Refer to Section 4.3.5 and related material in Section 4.3.

- **4mm and 8mm DAT tape**:

  These tape formats also use the **tar** command. The device names used for these tape devices are dependent on your system. The HDSware 4mm DAT tapes use a block size 512 and a blocking factor of 20 as default settings.

  tar xvf /dev/rst0

  Some systems, such as Ultrix systems, use other settings, and may need entries on the command line to read the tape. This can be set on the **tar** command line; for example, this command sets a blocking factor of 40:

  tar xvbf 40 /dev/rmt0h

  Check your system for the correct setting and syntax of this command.

- Other formats

  Special formats, such as HP QIC cartridges and 9-track tapes, work normally on those tape devices. The device names used for these tape devices are dependent on your system.

1.3.6.2 Diskette handling (5.25" or 3.5"):

- **Sun and SCO diskettes**:

  On your host system, move to the destination directory for your utility programs and font files. Then issue the **tar** command to extract the files from the diskettes:

  tar xvf <device name>

  You will be prompted to insert the diskettes in order. The files will be read into the current directory as individual subdirectories.

- **MS-DOS diskettes**:

  These diskettes are high density MS-DOS format (1.2 MB for 5.25" diskettes, 1.44 MB for 3.5" diskettes). There are a number of diskettes containing the fonts and client application programs. The files are compressed tar files copied onto MS-DOS diskettes. Transfer the files as binary files from the MS-DOS host to an appropriate UNIX directory. (With an FTP transfer use the **binary** command; with MS-DOS transfers use the **xcopy /b** command.)

  With the files in the UNIX directory, rename them to comply with UNIX syntax and uncompress them:

  mv 100dpi.trz 100dpi.tar.Z and so on for all compressed files

  uncompress 100dpi.tar.Z and so on for all compressed files

Then use the **tar** command to extract the files and create their separate directories:

tar xvf 100dpi.tar

The files will be read into the current directory as individual subdirectories.

1.3.7 HDSware Utility Programs

There are a number of utility programs provided with HDSware. These are programs for ViewStation management and handling fonts.

There are two HDSware directories, one with executable files for a variety of host computers, and another with the source code and appropriate makefiles for your own compilations.

1.3.7.1 Compiling the Utility Programs from Source Code

Generating executable files for the utility programs for other host types should present few problems, since the source files use standard X Window System libraries. A makefile is included for each of the programs and may require slight modification for some systems.

If your host machine doesn't correspond to or emulate one of the machines for which an executable file exists, you will have to compile the source code for your machine. The compilation is straightforward and uses normal compilation routines. The */contrib* directory has a shell script, xmkmf, which creates a makefile from an Imake file.

1.3.7.2 bdfto960 Font Utility Program

This program lets you convert .bdf fonts to HDS ViewStation .snf format. Since the HDSware tape contains the standard MIT fonts, and many specialized fonts, you will probably only use this for your own special fonts.

The fonts included in the MIT X distribution are .bdf fonts (Bitmap Distribution Format), which means that they are a generic type of font file intended to be compiled into different .snf (Server Natural Format) required by each server. The X distribution also provides a compiler, **bdftosnf**, to create the .snf fonts, but this compiler creates correct .snf fonts only if the machine doing the compilation is the same architecture as the server using the .snf files.

If you have the standard X distribution, you can create the .snf fonts needed for the ViewStation by using the **bdfto960** font conversion utility. This is similar to the X distribution's **bdftosnf** except that **bdfto960** converts fonts to the ViewStation format on machines of any architecture.

Create a special directory for the new HDS .snf files. Copy the .bdf files to the new location and then use the **bdfto960** utility to convert them.

The **bdfto960** font conversion utility converts a .bdf file to an HDS ViewStation .snf format (that is, Least Significant bit and byte first, scanline padding = 1, and glyph padding = 4).

Compiliing large numbers of fonts can sometimes be a lengthy process. This little shell script program will help you handle the conversion process by handlling the font files in a loop.

For the C Shell: */bin/csh*

```
foreach i (*.bdf)
echo $i
bdfto960 $i > $i:r.snf
end
```

For the Bourne shell: */bin/sh*

```
for i in *.bdf
{
echo $i
bdfto960 $i > `basename $i .bdf`.snf
}
```

Note that ` is an accent grave; be careful with spaces in these commands.

When the conversion to .snf files is complete, you must make a *fonts.dir* file for each directory. This can be done with the **hdsmkdir** utility, described below.

1.3.7.3 hdsmkdir Font Utility Program

The **hdsmkdir** utility program creates a fonts.dir file for the .snf font directory you've created. It is similar to the **mkfontdir** program included with the MIT X distribution except that it works on machines with any type of architecture. Erase any existing fonts.dir files before you start.

The **hdsmkdir** utility uses the following command line:

hdsmkdir [<*dir 1*> [<*dir 2*> ...]]

where[<dir 1> [<dir 2> ...]] specifies which directories should be used to create the fonts.dir file(s). If no directory is specified, the current directory is used.

1.3.7.4 hdsfc960 Font Conversion Utility

If you do not have .bdf font files to convert with the **bdfto960** utility, you can use the **hdsfc960** utility program to produce .snf files for the HDS ViewStation. If you do have .bdf files, this utility is not needed. It is used to convert fonts that you have only in some server format, such as the DEC .dwf font files, or for fonts that are in non-bdf generic format, such as PostScript font files.

The **hdsfc960** font converter is a special utility program that automatically creates .snf files for the HDS ViewStation from any type of font files available to and readable by a target X Window System server. The **hdsfc960** utility reads the font information with standard X protocols and creates an .snf font from that data. For instance, if you have DECwindows server using its .dwf fonts, you can use the

hdsfc960 command to convert some or all of the DECwindows fonts into HDS ViewStation .snf fonts.

To use the **hdsfc960** program, you must first create a directory for the new HDS .snf fonts. It can be in any location, but should be specifically for the HDS fonts and should be empty. For example, the directory might be named: */usr/lib/X11/fonts/HDS/snffonts*.

The **hdsfc960** program converts, in its default setting, ALL the font files currently available in the target server's fontpath. You can use **xlsfonts** to see what fonts are in the target server's fontpath. You probably want to restrict (temporarily) the target server's fontpath to the fonts you want.

Enter the command with the address of the target server (the host running the .snf fonts you're converting) and the destination directory of the HDS ViewStation .snf fonts you're making:

hdsfc960 -d *target IP address:display* <dest. directory>

or, in this example

hdsfc960 -d 128.91.3.12:0 */usr/lib/X11/fonts/HDS/snffonts*

The **hdsfc960** program works automatically; it locates the fonts in the target server's fontpath (it uses **xlsfonts**), reads the font data (and reconstructs the .bdf format), then creates the correct .snf font files (using **bdfto960**), creates a *fonts.dir* file for the fonts (using **hdsmkdir**), and puts them in the directory you specified. Note that the command program, **hdsfc960**, must be in your search path and you must have execute privileges for it. The other utility programs, **getbdf**, **bdfto960**, and **hdsmkdir**, must also be present to run **hdsfc960**. You may find it necessary to log onto the target server where the fonts are located rather than trying to execute these commands from a remote location.

The command also accepts other arguments for specific usage:

hdsfc960 [-f *filename*] [-d *display*] [-help] [<dest direct.>]

where:**[-f *filename*]** specifies a file name of fonts to convert.

If you don't want to convert all your fonts, you can specify a file which contains the fonts you've selected. The file must have font file names in it, with one entry per line. For instance, you could use the command:

xlsfonts -1 > *font.list*

or, to select some particular adobe fonts from the whole list:

xlsfonts -1 -fn "*adobe*" > *font.list*

to generate this file. The -1 (one, not lowercase L) specifies that **xlsfonts** prints one font name per line. This file is redirected to the *font.list* file, which you can edit to remove any fonts you don't want converted.

[-d *display*] specifies the target display of the fonts.

This is the IP address or name of the server host followed by the server display where the fonts can be found. This must be shown as a display, <IP host:display>. If this display is different from the display you're working on, you must specify

this argument because the default value is the current DISPLAY environment variable. If you are not running **hdsfc960** on the target server, you must make sure your host has access privileges on the target server.

[-help] gives some help information about the **hdsfc960** command.

1.3.7.5 getbdf Font Conversion Utility

The **getbdf** font conversion utility creates a .bdf format font file from the font information supplied by the target server.

The **hdsfc960** utility uses this program automatically, so unless you have special requirements, you probably won't use this as a separate utility program.

The **getbdf** utility uses the following command line:

```
getbdf [-d display] -fn <font name>
```

where [-d display] specifies the target server for the font named.

1.3.7.6 hdscrypt Encryption Utility Program

The **hdscrypt** program is used for the encryption of user passwords in a remote configuration file. If you are using this ViewStation feature on your system, you will need this encryption utility. This program is based on a public-domain routine included with MicroEMACS written and copyrighted by Dana L. Hoggett. The name has been changed to **hdscrypt** to avoid confusion with other crypt programs.

The encryption program is used to protect the integrity of user passwords when the ViewStation is started with Bootp and a configuration file containing a user password is used. The configuration file, because it is loaded with the TFTP protocol, must be generally accessible, but the *bootptab* file, since it belongs to the root user, can be hidden and protected.

The user password is entered as the key in the **hdscrypt** program and the encrypted value obtained. The password is entered as plain text in the **T128** field in the *bootptab* and the encrypted value entered in the configuration file. When the file is loaded, the ViewStation compares the plain text password from its Bootp message with the configuration file password it has decrypted with its own internal **hdscrypt** program. If the two values match, the session proceeds. If they don't match, the session fails.

The **hdscrypt** directory includes the **hdscrypt** source file and a makefile in addition to several executable files for various host architectures.

There are examples of encryption with hdscrypt in Section 1.11.1.6.

1.3.7.7 flashfilesys Utility Program

These scripts are discussed in detail in Section 1.10.5.

1.3.8 HDSware Contributed Programs

There are a number of contributed programs included with HDSware.

1.3.8.1 bootp

HDSware includes source code for bootp and executable files for both BSD and Sys V type hosts. The operation of bootp is discussed in detail in Section 3.2

1.3.8.2 hdsrunwm Launcher Program

The hdsrunwm launcher program is a host-based script to launch your ViewStation window managers.

1.3.8.3 xscope Diagnostic Utility

Xscope is a diagnostic program that records the X requests between the client and the server. This is very useful in analyzing problems at a low level.

The pexscope utility program is included with HDSware 3.2 as an HDSperx client. Pexscope is identical in function to xscope and is easier to use as a local client. Pexscope man pages are in Section 2.6.17.

1.3.8.4 xmkmf Utility Program

Xmkmf is an X utility to make a makefile.

1.3.8.5 snftobdf Font Conversion Program

This is a font utility program to generate .bdf font files from .snf font files. You must know the .snf format parameters in order to use snftobdf.

1.3.8.6 usepty Utility Program

This is a utility to use a pseudo-tty connection. This is discussed in Section 1.2.6.3.

1.3.8.7 xrsh Utility Program

This is an X utility for doing remote shell commands.

HDS ViewStation: Network Connections

This chapter describes the ViewStation connecting to Ethernet networks of different types, to Token Ring networks, and serial connections directly to a host or with PPP, SLIP, and CSLIP protocols.

Section 1.0 - HDS ViewStation

1.4 Network Connections

The HDS ViewStation and the X Window System are designed to work on a network, that is, a connection that includes a number of different devices, host computers, other ViewStations or workstations, etc. The X Window System is an environment which makes communications on the network appear to the user as if all devices are the same and that there is a direct connection between the ViewStation and the host computer (or computers).

The ViewStation supports up to four separate IP addresses for its connections, one for the Ethernet connection, one for the Token Ring connection, and one for PPP or SLIP connections on each of its serial ports.

■ Ethernet Connections

Normal connectivity is with the Ethernet interface, either Twisted-Pair, ThinWire, or ThickWire. The HDS ViewStation also supports the IEEE 802.3 Ethernet interface. Although both use the name "Ethernet" they are different and incompatible. The ViewStation automatically detects the difference and uses the appropriate type of interface.

■ Token Ring Connections

The ViewStation can use a Token Ring connection if it is equipped with a PC Card adapter board and a Token Ring PC Card module.

■ Serial Connections

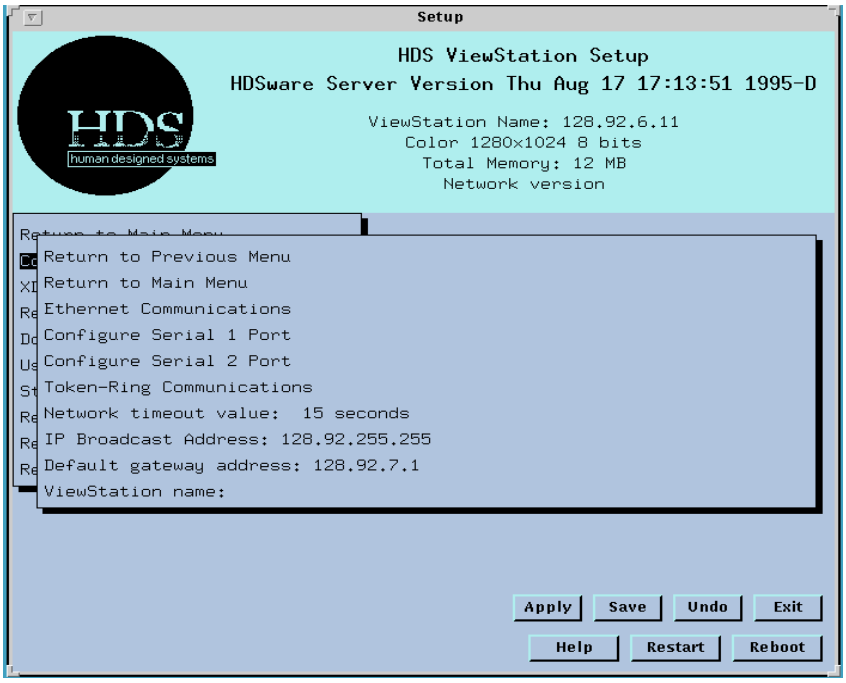
The ViewStation can also connect to host computers by means of serial connections, using either of the ViewStation's serial ports. The serial connection can be

either for TCP/IP communication using the PPP, CSLIP, or SLIP protocols, or a direct serial connection to the computer (either one using a direct serial connection or a modem connection).

The direct serial connectivity can be especially useful when using the ViewStation's local terminal emulators, either the HDSterm client to make a connection as a VT320 terminal, or the HDS3270 client to make a connection as an IBM 3270 terminal, or the Tandem 6530 client when connecting to a Tandem computer. The direct connection lets the ViewStation's emulator and GUI (Motif, OPEN LOOK, or FVWM) run even if the host is not using a network or is not running the X Window System.

1.4.1 Communications Menu

The ViewStation's communications settings are controlled by the Communications Menu in Setup Mode, shown below:



The Communications Menu gives you settings for all your ViewStation communications interfaces.

The **Ethernet Communications** menu lets you select settings for your Ethernet interface, including your IP address.

The **Configure Serial 1 Port** menu lets you to select the operation for serial port 1, the RJ11 port. These options are discussed in Section 1.4.4.

The **Configure Serial 2 Port** menu lets you to select the operation for Serial Port 2, the DB25 port on the optional Communications board. These options are discussed in Section 1.4.4.

The **Token Ring Communications** menu lets you set an IP address for your Token Ring interface as well as other settings for your Token Ring network.

The **Network timeout value** field sets the time interval that a request waits for a response. If you make a request to logon to some host on the network, the request will wait for the specified length of time before it either makes the connection, or gives up and ends the request process. If your timeout value is 15 seconds, a logon request will fail if it has to wait longer than 15 seconds. This value should be some brief time, 15 or 20 seconds, (based on the host's response time and load) so you will know a process has failed without too much waiting.

The **IP Broadcast Address** field sets the network wildcard address, typically 0.0.0.0 or 255.255.255.255. The system administrator sets the correct value for this field. Some systems prefer to use a local network broadcast, such as, 128.91.0.0 or 128.91.255.255. Your system administrator will choose this value.

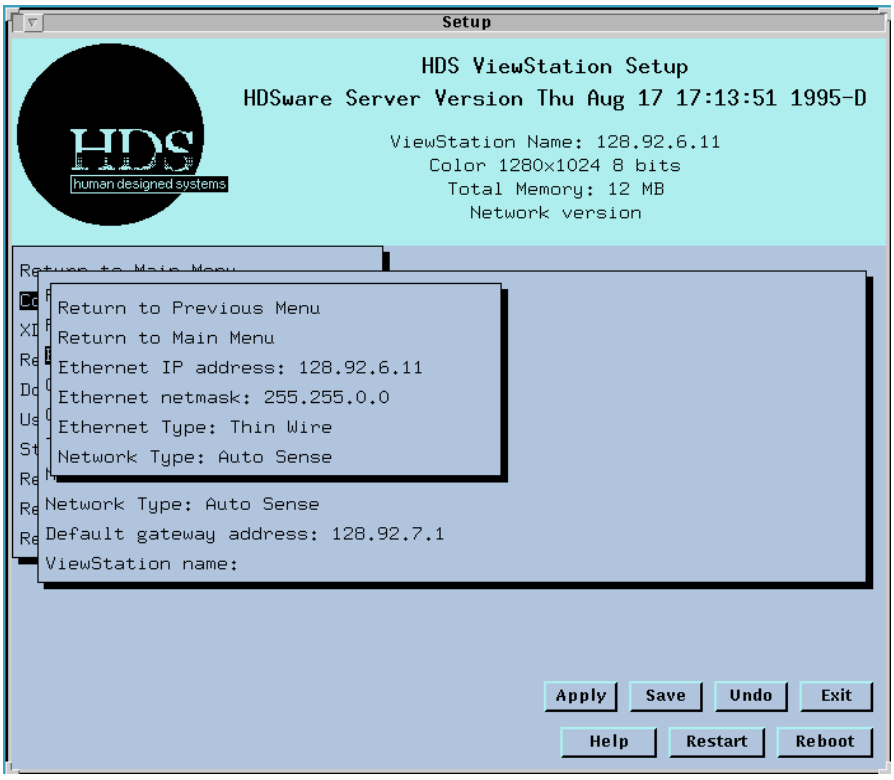
The **Default Gateway address** field lets you specify a gateway host. A gateway host computer is a computer on your local network that connects to other networks. Refer also to network routing discussed in Section 1.4.5.

The **ViewStation Name** field lets you enter the ViewStation's name, which is used to identify the ViewStation on the network (if available to a name server or entered in the host's */etc/hosts* file) and also as the display name in the ViewStation's setting of the display environment variable (with keystrokes Alt-F12, etc. See Section 1.8.2.3 for more detail).

1.4.2 Ethernet Connections

Connecting the HDS ViewStation to the Ethernet network involves two steps: the physical connection and the logical connection. The physical connection is usually a simple process of attaching the network cable to the connector on the back panel. The logical connection involves selecting the type of connection, obtaining an address on the network, and making some other simple settings. Obtaining a network address for the ViewStation is the function of the system administrator and is discussed later in the chapter along with other addressing procedures.

1.4.2.1 Ethernet Communications Menu



The Ethernet Communications Menu offers several important settings for your Ethernet connection.

The **Ethernet IP address** field lets you set the IP address for the ViewStation's Ethernet connection (this may also be set in the HDSware menu on the Main Menu, or by some network service like BOOTP or RARP). The actual value is assigned by the system administrator appropriate to your network.

The **Ethernet netmask** field specifies which parts of the your IP address must match the IP addresses of other devices on the local network. If the indicated parts of the addresses don't match identically, a gateway host computer must be used to obtain access to those other addresses.

For instance, entering a netmask of 255.255.0.0 means that the first two groups of numbers in the address must match identically, but that the last two groups can be any numbers. Thus, if your IP address is 128.91.2.2, you can connect directly to any other devices with addresses 128.91.nnn.nnn. You would have to use a gateway to connect to a device with an address 128.92.nnn.nnn. The netmask setting defines the addressing characteristics of your local network.

In precise terms, access to the gateway is based on the comparison between the result of a logical AND operation with the ViewStation IP address and the network mask to the result of a logical AND operation with the host IP address and the network mask. If the comparison shows identical values, the address is on the

local network. If the comparison shows different values, access must be obtained through the gateway. This field is password protected.

The **Ethernet Type** field selects the Ethernet type. The ViewStation has Twisted-pair as its factory configuration, with ThinWire and ThickWire Ethernet on an optional Communications board. The Twisted-pair interface is sensed automatically when connected, so no setting is necessary in Setup Mode.

The Setup Mode setting is for ThinWire (see the following sections for the effect of this setting on Twisted-pair connections). You can also select the ThickWire connection in Setup Mode or in Monitor Mode.

The **Network Type** field selects the Ethernet type, either generic "Ethernet" or IEEE 802.3 Ethernet. The default setting is Auto Sense, so the ViewStation makes this decision for you if you don't know which type is in use.

Refer to Section 1.4.2.7 for a more complete discussion of these Ethernet types.

1.4.2.2 Physical Connection to the Ethernet Network

There are three kinds of physical Ethernet networks: Twisted-pair, ThinWire, and ThickWire. These Ethernet networks are most easily distinguished by the connectors they use. These connectors are RJ45, BNC, and DB15, respectively.

The Twisted-pair Ethernet uses an RJ45 connector and wire similar to telephone jacks and cables to connect to the network. ThinWire Ethernet uses a thin, flexible coaxial cable with a T-type BNC connector for each node connection on the network. It is sometimes called thinnet or cheapernet. ThickWire Ethernet uses a rigid cable with nodes that connect to the devices with a thick coaxial cable ending in a DB15 connector. ThickWire Ethernet frequently uses an intermediate connecting device called a Medium Access Unit.

The HDS ViewStation has no special requirements for its connection with any of these Ethernet types, but all networks are somewhat different so you should consult with your system administrator to see what is needed for your network connection.

Twisted-pair Connection to the RJ45 Connector

Twisted-pair Ethernet uses an RJ45 connector on the back panel. This is a plastic telephone-type connector. Make sure it is fully inserted and locked.

(Note that the ViewStation Twisted-pair interface runs with SQE and Link Integrity enabled, in compliance with the IEEE 802.3 10BaseT standards. Some systems do not observe this standard completely. Hubs and routers for Twisted-pair networks should be set according to the standard. If you observe delays in echoing characters on the Twisted-pair network, there may be a problem with your settings. Refer to the next section for a more complete discussion of this issue.)

ThinWire Connection to the BNC Connector

The HDS ViewStation is set up to use ThinWire Ethernet in its factory configuration. You can make the physical connection just by attaching the Ethernet cable to the BNC connector on the back panel. You must use a T connector to connect the

Ethernet to the ViewStation. If you have only a single Ethernet cable, you must use a T connector with a terminator on the other side since you cannot connect it directly to the ViewStation's BNC connector.

ThickWire Connection to the DB15 Connector

ThickWire Ethernet uses the DB15 connector on the back panel. (The DB15 coaxial cable doesn't usually connect directly to the ThickWire Ethernet, but attaches to a Medium Access Unit (MAU), which in turn connects to the thick coaxial cable.)

1.4.2.3 Twisted-pair Ethernet with Link Integrity and SQE Settings

The Twisted-pair Ethernet connection runs with Link Integrity and SQE enabled, in compliance with the IEEE 802.3 10BaseT standards. Some systems do not observe this standard completely and need to have other settings. If you have a hubs and routers on your network, you may find that you need to disable the Link Integrity setting of the Twisted-pair interface. Link Integrity pulses are sent at the end of each transmission to verify communications between the hub and devices on the Twisted-pair network.

The ViewStation can disable Link Integrity either by software or hardware settings. If you use the Twisted-pair interface with the Ethernet type set to "ThinWire" (on the Communications menu of Setup Mode), the ViewStation uses Link Integrity; this is the Default setting. If you set the Ethernet type to "ThickWire" with the Twisted-pair interface in use, then Link Integrity is turned Off.

You can also control the Link Integrity pulse setting by selections on Jumper 16 on the main circuit board. With jumpers 1-2 selected for JP16, Link Integrity is enabled; this is the factory setting. With jumpers 2-3 selected for JP16, Link Integrity is disabled.

Another setting that is often thought to be related is the SQE signal. An SQE signal is sent on the collision detection line to verify that collision detection is working. Some systems want to disable these signals, because they increase the traffic on the network. The ViewStation uses the SQE signals internally only, as communication between the Ethernet controller and the Ethernet transceiver chips on the main circuit board. Since the ViewStation does not send the SQE signals on the network, requirements for the presence or absence of SQE signals is not a concern for ViewStation users.

1.4.2.4 Selecting the Ethernet Physical Type

The factory base configuration for ViewStation has the Ethernet Twisted-pair connector. The ThinWire and ThickWire connectors are present on the optional Communications board. The server uses the ThinWire setting as its default configuration. Since the Twisted-pair connector is auto-sensed, no selections need be made for the ViewStation to communicate on its Twisted-pair port. (Refer also to the previous section for the interaction of Twisted-pair with Thin- or ThickWire Setup Mode selections.)

If you use another type of Ethernet connection and are starting the ViewStation with Bootp, you can change the Ethernet type by using the ViewStation's Monitor Mode, which is discussed in detail in Section 1.6.2 of this manual.

You may be connected to only one type of Ethernet at any given time.

- **Choosing Twisted-Pair Ethernet**

Connect to the Twisted-pair port; select ThinWire in Setup Mode. This is the Default configuration and setting.

- **Choosing ThinWire Ethernet**

Connect to the ThinWire port; select ThinWire in Setup Mode (the default setting)

- **Choosing ThickWire Ethernet**

Connect to the ThickWire port; select ThickWire in Setup Mode

The Twisted-pair Ethernet interface is auto-sensed; all that is required is that you make the Twisted-pair connection. To use ThinWire Ethernet, just connect the ThinWire connector; the ThinWire interface is the Factory Default setting in Setup Mode. If you want to use ThickWire Ethernet, you must choose the ThickWire setting in Setup Mode (the Ethernet Connections submenu of the Communications Options menu). If you are starting for the first time and do not have a flashPROM board which allows you to get into Setup Mode before you connect to the network, you can modify the Thin / Thick Ethernet setting in Monitor mode.

1.4.2.5 Logical Connection to the Ethernet

Setting up an HDS ViewStation in an X environment for the first time is a fairly simple procedure, but it involves a number of steps and processes that may be unfamiliar to users. The process amounts to identifying the ViewStation to the other parts of the network and exchanging addresses so everyone can communicate with each other.

In the simplest case, you enter the Ethernet IP address number assigned to your ViewStation in Setup Mode (in the HDSware menu or in this menu) and don't change it after that. In another case, you can use the BOOTP or the RARP utilities to determine the IP address from the network (if those services are available on your network). These techniques are discussed in Section 1.6. These network-based addressing techniques allow the IP address (and other Setup Mode entries) to be modified by the system administrator remotely without having to change the addresses on each device.

The IP (Internet Protocol) address allows all devices on the network to communicate with the ViewStation and vice versa. If your network is simple, you just have to enter the ViewStation address. If the network is more complex, including connections to other networks, you must also set the network mask and the gateway address fields also (these are discussed in the Setup Mode sections where this data is entered). You may have to give the ViewStation more than one IP address (one for Ethernet, one for Token Ring, one for SLIP, etc.). The ViewStation supports up to four IP addresses and routes packets from one network to another. Refer to Section 1.4.5 in this chapter for a discussion of the ViewStation's routing.

The IP address setup is simple but depends on the system administrator assigning addresses and adding the ViewStation addresses to the configuration lists and other network-level functions. This discussion won't discuss those tasks in detail but will show the general process required. All systems are different and the system administrator must make the adjustments for your network arrangement, for the host computers and their requirements, and the locations of various lists and addresses on your system.

1.4.2.6 Network Addressing

Each device using network communications has a unique, permanent hardware address, a 48-bit number assigned at the time of its manufacture. This is called the Ethernet address. For the ViewStation this number is assigned by HDS for each unit. (This hardware address is shown on the boot screen, with the "hwconfig" command in the Console window, and on a label on the back panel). This is an identification for the ViewStation that other devices use in network communications. See also Section 1.1.12.1.

In addition, each device on a network has a logical address, called the Internet Protocol address, or IP address, which must be unique for its own network. The IP address is assigned by the system administrator according to your particular network configuration and conventions. These assignments are done in an orderly way to avoid duplication of an address, but also to group addresses into network subdivisions, or to conform to national and international addressing conventions for wide-area communications, and so on.

These addresses and their forms are important because they are used to control access to network broadcasts, or netmasks, or access to network gateway hosts. These things are discussed in Section 1.5.6.1.

An IP address is in the form *nnn.nnn.nnn.nnn*, with each field ranging between 0 and 255; a typical address would be: 128.91.6.11. Usually the first two number groups represent a unique address in the general network and the last two number groups apply to local networks and individual units.

For systems using the Remote Configuration option, the filename for each unit is usually given as the hexadecimal representation of the IP address. Thus the remote configuration filename might be:

IP address: 128.91.6.11 Configuration filename: 805B060B

The IP addresses are usually given to the other devices on the network with a communication technique called ARP, or Address Resolution Protocol. This is a message by the ViewStation to all devices on the network which requests, "Match my hardware address with my IP address." All the devices on the network register these two addresses for the ViewStation so they know what device is communicating and where it can be found. This requires that the ViewStation information and assigned address be entered in the network's ARP data base, which must be done by the system administrator (some systems do this automatically when a new device is connected to the network). Refer to Section 1.6.5 for more details about ARP operation.

The Console Window, described in Section 1.2.1, can be used to display the ViewStation's routing table. This shows the current settings for various communications connections, such as the serial or network connections, the netmasks, etc. This routing table shows the Ethernet connection (shown as ec0) and the serial connection (shown as sl0). The netmask is shown, including the length in bits.

1.4.2.7 IEEE 802.3 and "Ethernet" Networks

The name Ethernet covers the generic "Ethernet" as well as the specific IEEE 802.3 standard Ethernet. These are not the same logical interface. The IEEE 802.3 interface uses a different header packet format (Ethernet using one field as a packet type rather than the packet length of the IEEE 802.3 interface). This difference in header format makes the two interfaces incompatible.

IEEE 802.3 Ethernet is becoming more common and is often used on IBM RS6000 host computers. You may not be aware of these differences because of the informal usage of the name "Ethernet". The difference may only appear when you connect to another device which doesn't support IEEE 802.3 and find it doesn't communicate.

The HDS ViewStation automatically detects both Ethernet and IEEE 802.3 packets and communicates with both using the proper interface. When the ViewStation first connects to its Ethernet, it sends two packets, one in Ethernet format and one in IEEE 802.3 format. Based on which packet format is used for the reply, the ViewStation uses that interface format for the responding host. This is the Network Type field in the Ethernet menu, with Auto-sense mode as the default setting for the ViewStation. You can also specifically select the interface type.

1.4.2.8 Use of Network Trailers

Data is transported over a network in packets. Trailer encapsulated packets are just modified IP packets with the IP packet header moved to the end of the packet. The ViewStation receives trailer encapsulated packets but does not transmit trailer packets.

1.4.3 Token Ring Interface

The HDS ViewStation supports Token Ring communications with its PC Card adapter card and a Token Ring "HDS PC Card Token-Ring 16/4 Adapter" card. This is an available option for all ViewStation models.

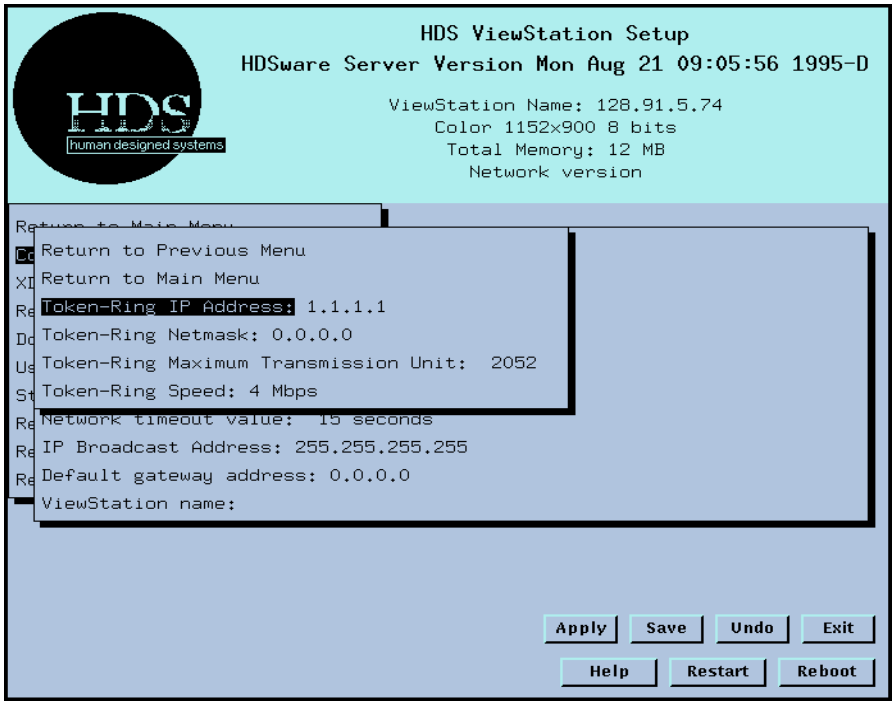
The ViewStation's Token Ring communication complies with IEEE 802.5 standards and uses IEEE 802.2 packet types. With HDSware version 3.2, Token Ring support is built into the bootprom and the Token Ring Communications menu allows your IP address entry and other settings.

Earlier HDSware versions (version 3.0.2 and bootprom v3.0.2-E) required some manual settings and interface control in the configuration file. [If you are using the v3.0.2-E bootprom with 4Mb capacity, you must have a jumper set (position JP7 on the ViewStation FX; position JP6 on ViewStation Ultra) to indicate the higher bootprom capacity. This applies to user upgrades only; factory installation adds the jumper automatically.]

ViewStation local clients, such as the IBM 3270 terminal emulator, add powerful capabilities to the Token Ring communications feature. This terminal emulation permits direct connection to IBM mainframe systems on Token Ring networks (at the same time keeping other ViewStation connections and emulations open).

An additional ViewStation feature, the HDS Developer's Kit, allows ViewStation customers to write and compile their own application programs to run on the ViewStation. This includes OSF/Motif applications which could be used on IBM mainframe computers without having either X or Motif running on the mainframe.

1.4.3.1 Token Ring Communications Menu



The Token Ring Communications menu offers a number of settings:

The **Token Ring IP Address** is the ViewStation's IP address on the Token Ring network.

The **Token Ring Netmask** is the netmask appropriate for the Token Ring network. For a discussion of netmask operation, refer to the Ethernet netmask section above.

The **Token Ring Maximum Transmission Unit** field lets you set the MTU for your Token Ring network. The default MTU is 2052.

The **Token Ring Speed** field reports the ring speed. You can select either 4 or 16 Mbps by means of a jumper on the ViewStation's circuit board described below.

1.4.3.2 Token Ring Specifications

ViewStation Hardware Interface

The HDS ViewStation uses "tr1" to designate the token ring interface to the PC Card internal slot and "tr2" to designate the PC Card external slot.

Token Ring communications can use either shielded twisted pair connectors (called Type 1 cabling) or unshielded twisted pair connectors (called Type 3 cabling). Cable adapters (DB9 or RJ45) are supplied with the Adapter card.

Network cable quality often determines whether 4 or 16 Mb ring speed is appropriate. Token Ring speed is jumper selectable for either 4 Mbps or 16 Mbps. This is controlled by jumpers on the JP9 block for the ViewStation FX and on the JP11 block on the ViewStation Ultra. Jumper pins 5-6 select the ring speed for the internal PC Card slot and jumper pins 7-8 select the ring speed for the external PC Card slot. Ring speed 4 Mbps is selected when the jumper is in place, and ring speed 16 Mbps is selected when the jumper is off. This configuration should be selected with the initial ViewStation order so the correct configuration is installed, tested, and shipped.

ViewStation Software Interface

The hardware interface is used in the boot monitor to set the Token Ring IP address and netmask as follows (tr1 designates the internal PC Card slot and tr2 designates the external slot). You can make these settings in the Setup Mode menu above, or in Monitor Mode at boot time, or in the configuration file:

```
ip trx nnn.nnn.nnn.nnn

nm trx nnn.nnn.nnn.nnn
```

ViewStation Token Ring communications supports source routing, so it will work on bridged networks. BOOTP and XDMCP are supported on the Token Ring network for both direct and broadcast access.

1.4.3.3 ViewStation Token Ring Controls

From the ViewStation's Console window, the "srt" shell command displays, adds, and deletes source route entries for token ring communications.

```
srt         displays all source route entries

srt -s      add/modify a source route entry

srt -d      delete a source route entry
```

If the Token Ring adapter becomes unplugged from the Token Ring network, the user can issue a command to re-insert the adapter onto the network:

```
ifconfig trX up
```

(where trX = tr1 for the internal slot and tr2 for the external slot)

This reconnects the token ring adapter.

The "token" shell command displays token ring interface statistics.

1.4.4 ViewStation Serial Connection

The ViewStation has two serial ports, Serial 1 (RJ11) with all ViewStations and Serial 2 (DB25) with the optional Communications board. These ports are configured from the Setup Mode Communications menu. The serial ports are designated */dev/tty1* and */dev/tty2* respectively.

1.4.4.1 ViewStation Serial Port Physical Connection

Using the ViewStation's serial ports for a connection involves selecting the physical port (RJ11, called Serial Port 1, or DB25, called Serial Port 2) and its communication configuration (baud rate, etc.). Available options are shown by toggling through the Setup Mode options. These choices are simple and can be made in Setup Mode on the Communications Options menu. Pin settings for the serial ports are found in Section 1.1.10.3 and 1.1.10.9.

The mode of serial communication can be direct, using the ViewStation's local terminal emulators, or it can use the TCP level interface with PPP, CSLIP, or SLIP protocols.

1.4.4.2 Serial Port Troubleshooting

Most problems with serial communications are cable problems. If you have no communication at all (no characters on the screen, or no activity), it is most likely a cable problem. Use a breakout box to analyze the transmission on each line or use another cable.

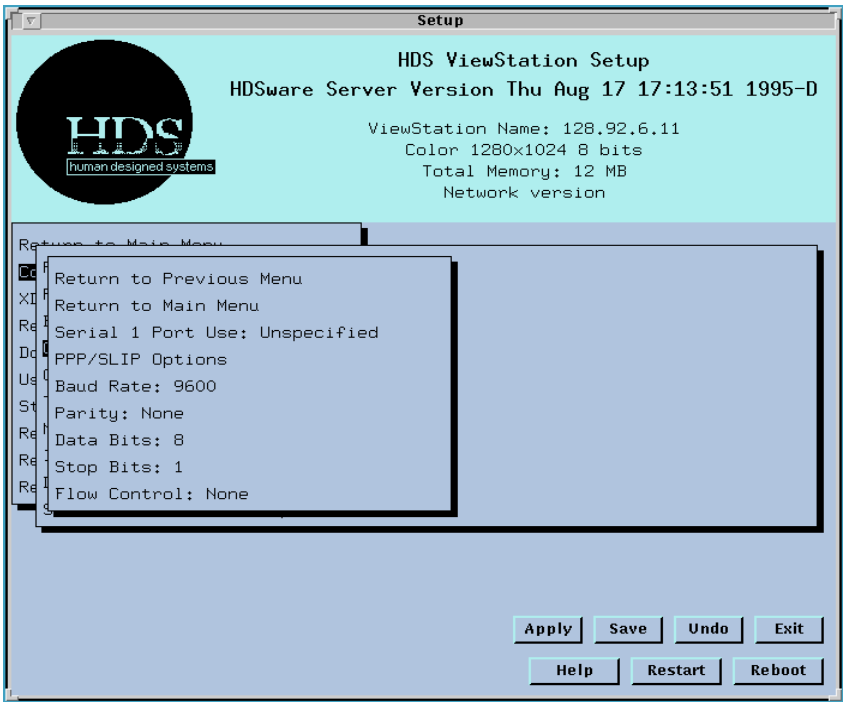
If you have characters on the screen, but they are garbled or nonsense, you can generally conclude that the cable is OK but some setting is wrong - the two devices have mismatched data settings, or baud rates, or parity checking. Check the recommended and selected settings for both devices and try other settings.

Other problems, like a print job losing characters or stopping before completion, can be a flow control problem. Verify the hardware or software flow control mechanism and try other settings.

Problems with SLIP, CSLIP, and PPP are generally configuration issues. Look at the man pages for these protocols to see the options and expected behavior. Sometimes routers and their settings need to be part of your configuration, since they may have to pass the PPP packets (if needed in your network environment).

1.4.4.3 ViewStation Serial Port Menu

The Communications Options menu contains fields for the serial port settings and is described in detail in Section 1.5.6.1. Click on the **Configure Serial 1 Port** line and you'll see this screen:



The field **Serial 1 Port Use** (or **Serial 2 Port Use**) determines how the particular serial port is used; it has options Unspecified, Normal SLIP, Compressed SLIP, PPP, or Console (the Console entry is discussed in Section 1.2.1). This is the assignment of the logical port to a connector.

Choose one of these configuration options, such as PPP or CSLIP (this compressed format uses "Van Jacobsen compression", RFC 1144; the remote host must also use this technique to decompress the data it receives). CAUTION: Select PPP, or SLIP options only if that is your only connection. If you have other connections to the host or Ethernet and use PPP only occasionally, you should select "Unspecified" as your serial port use and use the "Start PPP" option on the pull-down menu. The reason for this "Unspecified" selection is that if "PPP" is your Serial Port Use selection, the ViewStation will send repeated requests to the host to start the PPP daemon, even if you aren't currently using a PPP session.

The **PPP/SLIP Options** menu is discussed in the next section.

Now set the communications settings as required by the remote host connection, things like baud rate, parity, etc. When you have made all your selections, click on Apply Settings, (Save in NVM if you wish) and exit Setup Mode.

When you try to log onto the remote host you specified for CSLIP (using Telnet), the ViewStation opens the CSLIP communications port using the settings you specified. (Note that the communications interface is forced to 8 data bits and no XON / XOFF when using SLIP or CSLIP. Most Unix systems use 7 data bits and Even parity with their serial connections; the parity bit is required but ignored on input. SLIP and CSLIP require 8 data bits to communicate. The ViewStation auto-

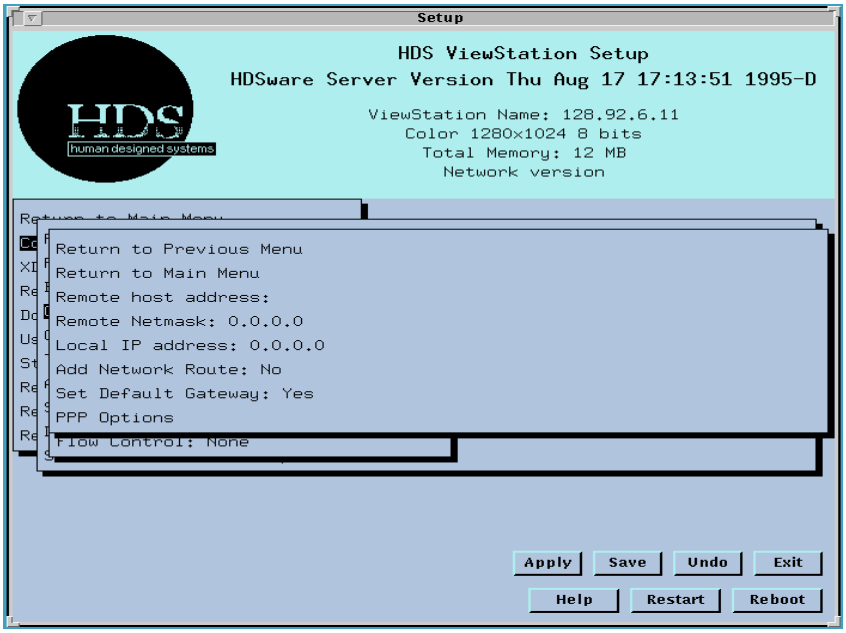
matically sets its serial ports to 8 data bits, and no XON / XOFF when the SLIP or CSLIP mode is active. The previous communications setting is restored when the SLIP mode is terminated. You shouldn't save settings in NVM while the SLIP interface is active unless you want these SLIP communication values saved for that port.)

You can log onto another host, using either Serial Port or Ethernet, with another Telnet session if you wish. The ViewStation looks at the host address you use in your Telnet session and then selects the appropriate interface as its connection, either the Serial Port 1 or Port 2 remote host, or some other host on your Ethernet network. The Default Telnet host field in Setup Mode lets you select one host as your Default connection.

Your remote host addresses for the serial ports should be different; if you use the same remote host address, the last one you set (with Apply settings) will be used. Note also that some things, like Domain Nameservice, rely on the flexibility and free access to many machines found on a network. If your font host, or name daemon, are running on a host that is not your SLIP host, they will work correctly if the physical connection to that host is present and active. You should realize that transferring font files over the SLIP serial lines will be slow, so most of these network processes (daemons and servers) should typically be on an Ethernet connection host.

The selection of the physical port should consider that CTS / RTS is the only flow control available, and this is only available on Serial Port 2, the DB25 port. The XON / XOFF flow control can't be used with CSLIP or SLIP. Note also that CSLIP and SLIP protocols must run with 8 data bits and no parity.

1.4.4.4 PPP/SLIP Options Menu



The **Remote host address** field lets you set the IP address of the host for your PPP or SLIP connection. This is not needed if your PPP host does address negotiation.

The **Remote Netmask** field sets the netmask appropriate for that host.

The **Local IP address** field lets you set the ViewStation's IP address for this serial port connection. This is required for SLIP connections and for PPP if Address Negotiation is not used. If PPP Address Negotiation is used, this IP address is assigned by the PPP host.

The **Add Network Route** field lets you specify whether the network route should be added to your route table. This can also be done manually, refer to Section 1.4.5.3.

The **Set Default Gateway** field lets you set a default gateway host for this connection.

The **PPP Options** menu gives you settings for your PPP configuration. This is discussed in detail below.

1.4.4.5 PPP Protocol for Serial Communications

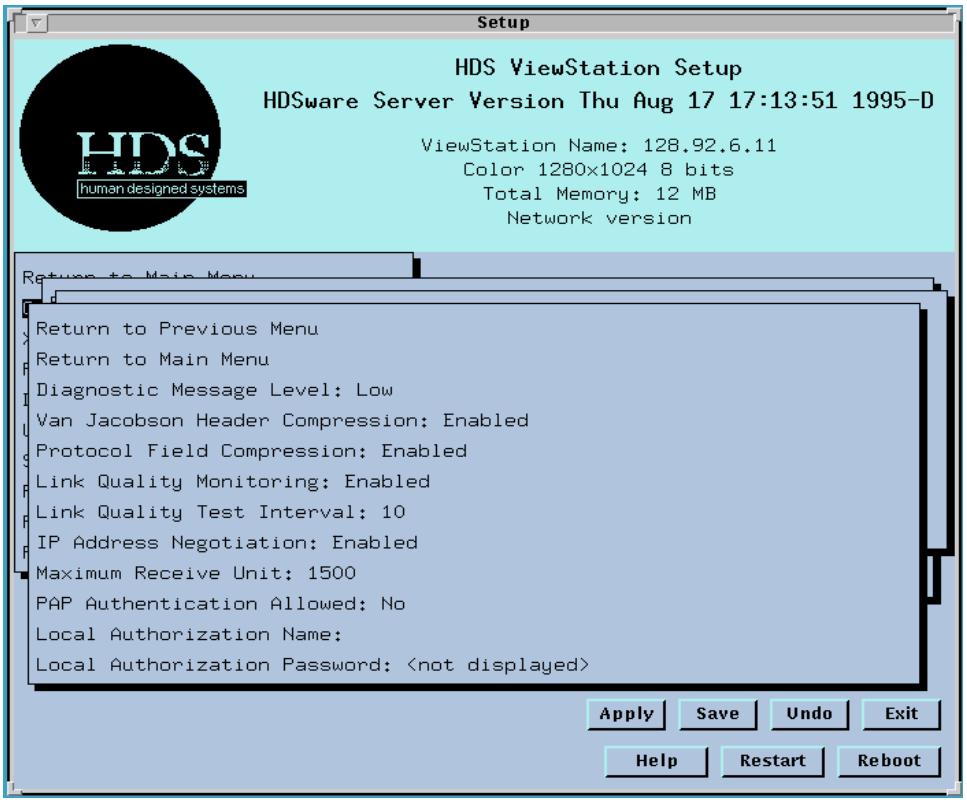
You can use the PPP Protocol (otherwise called Internet standard Point to Point Protocol) to run IP connections to X on a host computer. PPP is defined by RFCs 1171 and 1172. The PPP protocol is more than a superset of SLIP and CSLIP. The PPP protocol will not use or recognize SLIP or CSLIP packets.

PPP provides link-level error detection, IP address negotiation and assignment, Address / Control fields and Protocol field compression, Link-level authentication (defined by PAP in RFC 1172 and CHAP in RFC October 1991 Draft), Link-status monitoring by LQM (LCP Draft August 1991), as well as support for SLIP (RFC 1055) and CSLIP (RFC 1144). PPP corrects many of the deficiencies of SLIP and permits more features and adjustments in the serial communications interface.

PPP must be installed on your host computer; it is frequently included with commercial operating systems. There are many commercial programs which offer PPP connections for communications or you can obtain PPP from the Internet. It is included with all HDS ViewStation terminals.

To select the PPP Protocol, go to the Communications Options menu and click on **Configure Serial Port** line (for the line you want). Select the PPP option on the **Serial Port Use** line if your ViewStation uses only the PPP connection. Then click on the **PPP/SLIP Options** menu and then the **PPP Options** line. CAUTION: Select PPP, or SLIP options only if that is your only connection. If you have other connections to the host or Ethernet and use PPP only occasionally, you should select "Unspecified" as your serial port use and use the "Start PPP" option on the pull-down menu ("Serial Options" in the Serial Terminal window). The reason for this "Unspecified" selection is that if "PPP" is your Serial Port Use selection, the ViewStation will send repeated requests to the host to start the PPP daemon, even if you aren't currently using a PPP session.

The **PPP Options** menu contains a lot of PPP settings, which are explained briefly below. More detailed explanations are provided with the PPP products.



The **Diagnostic Message Level** field selects the level of message reporting for PPP. The values are Low, Medium, and High with the default setting of Low. These messages can help you in PPP negotiations and Medium is a useful setting for getting set up for the first time.

The **Van Jacobson Header Compression** field, when Enabled, allows the ViewStation to use TCP header compression (also known as CSLIP compression) if supported by the remote host. When Disabled, the ViewStation will not attempt or allow TCP header compression. The default setting is Enabled.

The **Protocol Field Compression** field enables or disables the compression of PPP protocol fields. The default value is Enabled.

The fields **Link Quality Monitoring** and **Link Quality Test Interval** control the generation of Link Quality packets. Link Quality Monitoring, when enabled, requests that the remote host send Link Quality Report packets periodically, at a rate determined by the Link Quality Test Interval field. When Monitoring is disabled, the ViewStation will not require or participate in link quality monitoring. The default setting is Enabled with 10 sec interval.

The **IP Address Negotiation** field will disable IP address negotiation. This requires that both PPP participants know the address of their peer before connecting. The default setting is Enabled.

The **Maximum Receive Unit** field allows setting the maximum PPP receive unit size. It can range from 128 to 1500. The default setting is 1500.

The **PAP Authentication Allowed** permits a remote host to ask for or provide PAP authentication when establishing a link. PAP authentication involves the transmission of cleartext passwords and can be a security problem. The default setting is Disabled.

The **Local Authorization Name** and **Local Authorization Password** fields give the hostname and host password which the ViewStation will use for PAP authentication if requested by the remote host.

1.4.4.6 Direct Serial Connections

The ViewStation uses the Serial Terminal Window as its serial interface. This window can use either Serial Port 1 or 2 (RJ11 or DB25) with an HDSterm window to simulate a VT320 character-based terminal connected directly to a host. This serial connection behaves as if it were a VT320 terminal connected to the remote host computer. This terminal emulation window allows you to do VT320 terminal commands from the ViewStation. If the host is not running the X Window system, the HDSterm window is a simple VT320 port.

The HDSterm window is a DEC VT320 terminal emulation client running on the ViewStation, so you can use it (starting from Setup Mode) anytime you want to open a serial terminal window.

Similarly, other HDS ViewStation terminal emulation local clients have the same type of connection, a direct serial line to the host and the appropriate emulation, whether IBM 3270 or Tandem 6530, and so on, running on the ViewStation.

You open the serial terminal window on the ViewStation screen by clicking on the **Serial Terminal Window** button in the Main Menu of Setup Mode. This opens an HDSterm window with VT320 terminal emulation. You are first asked to specify the Serial Port 1 or 2 (RJ11 or DB25) you want to use. If you haven't selected terminal mode for that port (in the Communications Options menu), you are prompted to do so.

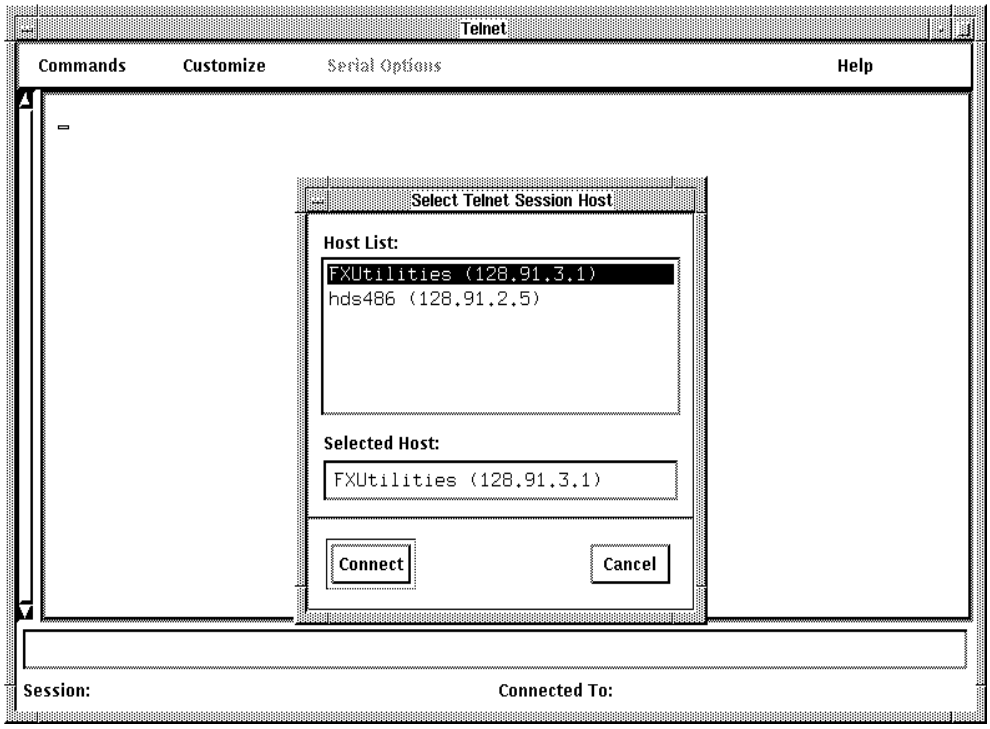
An HDSterm window, labeled "Terminal" opens for your terminal mode connection. At this point, you can log into your host computer, or initiate a command sequence to your modem, or whatever you do on the serial connection.

If you want to open a TCP connection after your serial connection is established, go to the the Serial Options pull-down menu. This menu has five entries, Start PPP, Start SLIP, Start CSLIP, Short Break, and Hangup. You can use the appropriate commands to open your connection to the modem or computer. Note that PPP, or other protocol, must be running on your host. Other TCP settings, such as gateways and routing information, should be in place on your host. If you want to hangup the modem, click on the Hangup button. Use the Commands pull-down menu to close the window.

If you are using a server code version without HDSterm, you can generate Break and Hangup signals from the keyboard. With focus in the Terminal window, the

Pause key creates a short BREAK (drops the data line for 300 ms.), the *Shift-Pause* key creates a long BREAK (drops DTR and the data line for 3.0 secs.).

This HDSterm window can be used as a permanent, character-based terminal. Most frequently it is used to open a modem connection, or some other terminal function, and then convert the connection to PPP, SLIP, or CSLIP communications which could then run an X-based IP connection over the serial line.



1.4.4.7 Serial Connection Example

This section describes an example of making a direct serial connection. It is only an example; some settings or methods may be different on your host system.

For a host connection:

Make sure your communications parameters on both the host and the ViewStation's serial port are correct. This includes the cable pin assignments and connection, the baud rate, data / parity bits, etc.

When you open the Serial Terminal window, you will get a login prompt from your host automatically. On some systems, you may have to hit a carriage return once or more.

For a Modem connection:

Make sure your communications parameters on both the modem and the ViewStation's serial port are correct. This includes the cable pin assignments and connection, the baud rate, data / parity bits, etc.

When you open the Serial Terminal window, give the modem commands to open the connection, such as (for Hayes-compatible modems):

atdt 6102778300

You may need to do some additional modem configuration with other modem commands. Check your modem's manuals for more details about the commands you can use. From that point, the modem will connect to the host computer and you will get a login prompt.

1.4.4.8 PPP Connection Example

This example describes starting a simple PPP session. It assumes that the PPP is correctly configured on the host system and that a serial connection can be opened to the host.

- Use the ViewStation's Serial Terminal window to open a serial connection to the host, either directly or with a modem. When the connection to the host is complete and PPP has been started on the host computer, go on.
- Select "Start PPP" from the Serial Options pull-down menu of the ViewStation's Serial Window. This closes the Serial Terminal window (but the modem connection remains open). You may notice a delay when the Serial Terminal window closes which is caused by the PPP negotiation with the host, but there is an interval permitted for you to open the Telnet session before the PPP connection times out. When the Serial Window disappears, open a Telnet session.
- Open a TCP/IP connection to the host with Telnet. Use the IP address of the PPP remote host. If you have HDSware 3.2, this new IP address for the PPP connection will be entered automatically for the Serial port you're using. If you have an older HDSware version, you will have to enter the new IP address for the ViewStation and restart it (the connection will remain open).

If you have PPP address negotiation enabled (the default), the remote host will determine and set an IP address for the ViewStation. If address negotiation is disabled, you must enter the ViewStation's PPP IP address on the PPP/SLIP Options menu.

It may be advantageous to establish a separate logon on the remote host for users making a PPP connection to a host. If your regular Ethernet login on the host is "bill", you may want to have a separate login account on the host for "billPPP". This would allow the billPPP logon to start a shell script that aids your PPP connection. This script might include a line to start the PPP daemon on the host (usually just starting */etc/pppd*, or whatever the configuration on your system).

1.4.4.9 SLIP Protocol or CSLIP Protocol for Serial Communications

You can use the SLIP (Serial Line Internet Protocol) or CSLIP (Compressed SLIP) protocol just as with PPP. If you use the Compressed SLIP protocol, you must have the ability to handle compressed packets on the remote host computer (Van Jacobsen compression, RFC 1144).

You must set your SLIP IP address for the appropriate serial line on the PPP/SLIP Options menu.

1.4.5 ViewStation IP Addresses and Route Table

The ViewStation, with HDSware versions 3.2 and higher, supports an internal user-definable route table. It supports up to four IP addresses (one for Ethernet, one for Token Ring, and one for each serial port - either an assigned address for SLIP connections or a negotiated one for PPP connections). SLIP and PPP addresses can be different. The SLIP address is entered in the Setup Mode menu (Configure Serial Port - PPP / SLIP Options); an address is needed for PPP connections if Address Negotiation is turned Off. If IP Address Negotiation is turned On for PPP, the IP address is assigned dynamically by the host when the connection is made.

1.4.5.1 PPP Configuration and Network Routes

Support for network routes is available with HDSware version 3.2 and higher. Earlier HDSware versions used the resource:

HDSsetup.remote.serial1.netmask

to establish a network route for SLIP or PPP. This has been replaced with two resources:

HDSsetup.serial1.add.network.route (default NO)

HDSsetup.serial1.add.default.gateway (default YES)

The first setting lets you add a specific network route; the default setting of No disables the feature. The second setting allows you to set the ViewStation's gateway as the remote host's IP address; this field is not available from Setup Mode.

More complicated network configurations are discussed below.

1.4.5.2 Typical PPP Configurations

There are several possiblities for using the ViewStation in a PPP environment. There are three possibilities for an IP address and nine possibilities for a network, which are described below. Most cases are handled correctly with the ViewStation's default assignments.

For instance, connecting a ViewStation from your home to the subnet in your office, or connecting from your office subnet to another subnet, or connecting from your office subnet to the Internet are all handled correctly with the ViewStation's default configuration.

If you have more complicated network arrangements, look at the definitions and tables below:

- Leaf - an isolated host
- Subnet - a network completely specified by one netmask (all hosts on the subnet have IP addresses that match the netmask and all IP addresses that match the netmask are found on the subnet [including gateways])

■ Internet - all other networks

ViewStation	Remote Host		
	Subnet	Internet	Leaf
Leaf	Case A*	Case D*	Case G*
Subnet	Case B*	Case E*	Case H
Internet	Case C	Case F	Case I

* indicates that the default settings on the ViewStation will work correctly unless multiple PPP interfaces are running.

Case A: the ViewStation is a leaf and connects to a subnet

ViewStation default configuration works correctly

Case B: ViewStation is on a subnet and connects to a subnet

- Standard configuration (one PPP connection):

  Remote host is default gateway should be Yes (default)

  Add network route should be No (default)

- Alternate configuration (multiple PPP connections):

  Remote host is default gateway should be No

  Add network route should be Yes

Case C: ViewStation is on an internet and connects to a subnet

Remote host is default gateway should be No

Add network route should be Yes

Case D: ViewStation is a leaf and connects to an internet

Remote host is default gateway should be Yes (default)

Add network route should be No (default)

Case E: ViewStation is on a subnet and connects to an internet

Remote host is default gateway should be Yes (default)

Add network route should be No (default)

Case F: ViewStation is on an internet and connects to an internet

You will have to add host or net routes manually using the ViewStation's local **route** command

Case G: ViewStation is a leaf and connects to a leaf

Remote host is default gateway should be No

Add network route should be No (default)

Case H: ViewStation is on a subnet and connects to a leaf

Remote host is default gateway should be No

Add network route should be No (default)

Case I: ViewStation is on an internet and connects to a leaf

Remote host is default gateway should be No

Add network route should be No (default)

1.4.5.3 ViewStation Local Route Command

HDSware version 3.2 includes a local route command to let you add or delete routes from the ViewStation's route table. Use the route command in the Console Window. The route command is:

route [add | delete] [net | host] <destination gateway> [metric]

This command lets you add or delete routes or networks and hosts for the ViewStation. The "metric" argument must be included for adding a route; it is the number of hops to the final destination.

The ViewStation supports as many as four IP addresses. Its communication is limited to those IP addresses and the local subnets appropriate for them. The concept of a "route" simply tells the ViewStation how to get to a different IP address. This specified route gives a path for any packets with IP addresses not on the ViewStation's known communications paths.

For a ViewStation (IP = 128.91.6.11) follow these examples:

To add a route for a host on a different net, say 198.89.6.3:

```
route add host 198.89.6.3 128.91.2.5 2
```

This command tells the ViewStation that any packets addressed to "198.89.6.3" should be sent to address 128.91.2.5, which is the "destination gateway" computer which will then forward the packets as needed.

The command to add a route for a network is:

```
route add net 128.92.0.0 128.91.2.5 2
```

which routes all packets addressed to the 128.92 network through the 128.91.2.5 gateway.

To delete the route for a network, the command is:

```
route delete net 128.92.0.0 128.91.2.5
```

1.4.5.4 ViewStation Local Ping Command

The ViewStation has a local "ping" command included with HDSware version 3.2 or higher. The man pages for the ping command are given in Section 2.6.18.

The ping command is useful in determining if a host is reachable from the ViewStation. It is an easy way to see if the gateways and routes are working. Simply type the ping command with the host you want to reach:

```
ping <hostname>
```

The return message reports if the host is alive and reachable or not.

HDS ViewStation: Setup Mode

This chapter describes the ViewStation's Setup Mode client and its menus and settings.

Section 1.0 - HDS ViewStation

1.5 Setup Mode

Setup Mode is a ViewStation client that lets you enter and modify ViewStation settings. Setup Mode provides a simple user interface to make these settings, with Help menus, editing tools, and so on.

Many of its settings can be entered automatically with the HDSware configuration feature. They can also be entered remotely by the system administrator by means of the ViewStation's configuration file. Setup Mode can also be run from a remote device on the network.

1.5.1 Entering Setup Mode

Entering Setup Mode is simple. An icon labeled "Setup" appears in the lower right corner of the root window screen each time you turn on the ViewStation. Moving the mouse to the icon and clicking the mouse button brings up the Setup screen.

1.5.1.1 Setup Mode with Hidden Icon

The Setup Mode icon can be removed from the screen by using the "Hide Icon" resource in the ViewStation's configuration file, *./hds-fx/config/config.fx*. This is sometimes useful for simplifying the user's screen or for reasons of security.

If you have the Setup Mode resource set to "Hide Icon", Setup Mode doesn't appear on the screen unless you hit the Setup Mode hotkey. This is a definable key sequence which you can set with the hotkey resource in the Setup client's defaultArgs table in the *./hds-fx/config/clients.config* file:

```
hds.client.setup.defaultArgs: -xrm '*hotkey: Shift Alt<Key>F1'
```

In this example the new hotkey would be Shift-Alt-F1. The default assignments for the hotkeys are as follows:

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Setup Mode Hotkey	Alt-SysRq	Alt-Help	Alt-PrSc	Alt-PrintScreen

1.5.1.2 Remote Setup Mode Operation

You can enter a particular ViewStation's Setup Mode from a remote device by making a rsh (remote shell) connection to that ViewStation, then starting the Setup client and specifying your device's display for Setup Mode. The command line would be:

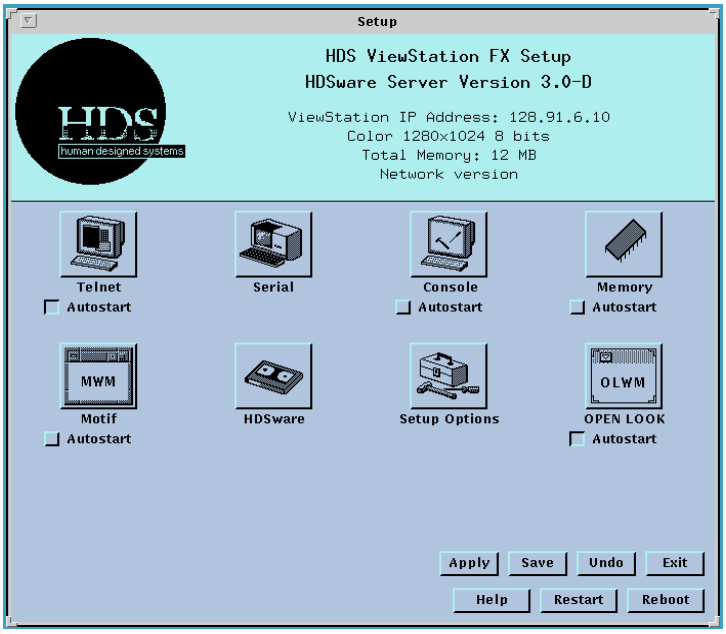
```
rsh <FXname> setup -display <remote_IP:0>
```

Note that only one Setup Mode can be active on a ViewStation, so any local Setup session will be be closed when the remote session starts. The local session will restart automatically when the remote session ends.

1.5.2 Setup Mode Operation

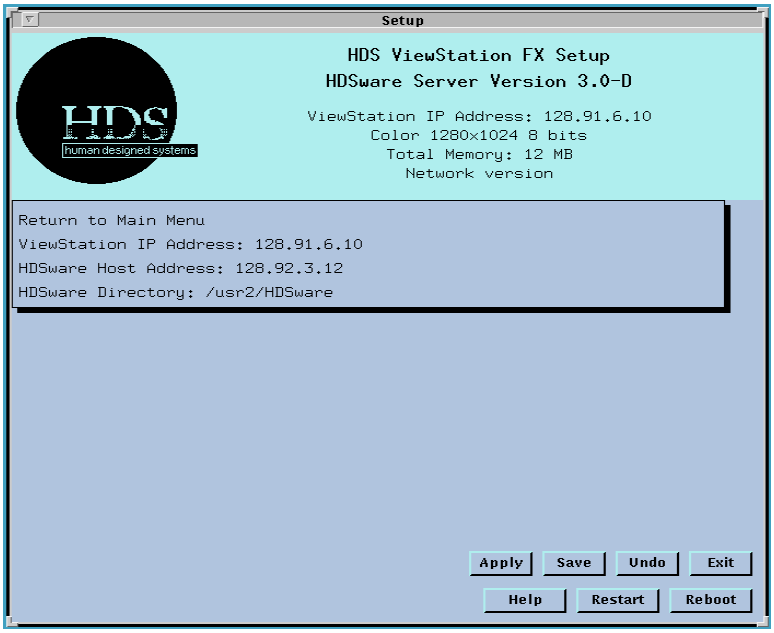
The operation of Setup Mode is intuitive, with a Main Menu showing buttons and boxes for ViewStation local clients and a Setup Options button containing layers of menus offering you choices and editing fields for other settings. Mouse movement selects different options and the left mouse button clicks select an indicated button or option. The selections are shown in reverse video and for important changes you are asked to confirm your selections with another submenu.

The discussion that follows helps you understand the contents of the fields and what selections are available. It can also help you see the forms addresses must have, the range of values permitted for a particular field, and so on.



1.5.2.1 Quick Starting with Setup Mode

The HDS ViewStation Setup Mode lets you configure your X terminal quickly and easily. The HDSware button of the Main Menu of Setup Mode (shown below) lets you enter your IP address, and your HDSware host and path, and the ViewStation does the rest automatically. This automatic configuration covers almost all of the important ViewStation settings.



Enter the IP address for your ViewStation. This IP address is assigned by the system administrator.

Enter the host address of the computer (the HDSware Host Address) where you loaded the HDSware tape.

Enter the directory location of the HDSware tape on the HDSware Host.

Click on the Save button to save the entries in NVM.

Restart the ViewStation. This is needed to register your new IP address. The ViewStation finds the HDSware host and path and then registers that location for all the files and paths it needs, such as for fonts, clients, etc.

The ViewStation automatically sets these fontpaths, paths to your HDSperx clients, configuration files, and other important data in your current Setup Mode selections. This makes setting up the ViewStation quick and easy. You should save theses settings in NVM.

Other important settings, such as gateways, Domain Name servers, etc., are not set this way. If you need these settings, you will have to enter them in the appropriate Setup Mode fields, as is discussed in the remainder of this section. These settings, and all settings in Setup Mode, can also be set automatically and remotely by using a Configuration file. This is discussed in Section 1.7.

1.5.2.2 Setup Mode Editing

The Setup Mode client has a number of user-defined fields. There is full keyboard editing capability for entries in these fields. The cursor keypad allows you to move the cursor left and right within a field and the up and down arrow keys move the cursor to the beginning and end of the field. The *Home*, *End*, *Page Up* and *Page Down* keys also move the cursor to the beginning or end of the field.

You can erase a field by pressing *Ctrl-U*. You can also use the *Backspace* and *Delete* keys to delete characters to the left. The *Insert* key toggles the keyboard into Insert mode. This is especially useful for editing long field entries.

1.5.2.3 Setup Mode Editing Keys

The editing keypads on the keyboards are active in Setup Mode on most keyboards.

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Cursor keys	yes	yes	yes	yes
Insert function	Insert	InsertHere	Ins	Insert
Delete function	Delete	none	Del	Del
Home/PageUp	Home/PgUp	Prev	none	Home/PageUp
End/PageDown	End/PgDn	Next	none	End/PageDown

1.5.2.4 File Names and Paths in Setup Mode

The ViewStation uses an internal file structure which is very similar to NFS, using host names and mount points for easier specification of file names and paths in a complex network environment. A detailed description of these meanings, as well as examples of usage, is found in Section 1.2.5.

In general, you can specify a file name or a path for Setup Mode fields by using a leading slash, then entering the host name (or IP address), the directory path, and the file name, with each field or directory level separated by a slash. For example,

/<hostname>/<path>/<filename>

or, more specifically,

/hdssun5/usr3/home/openwin/bin/olwm/.openwin-menu

/128.92.3.12/home/usr2/john/budget.doc

The ViewStation understands file names containing all of these elements, with host names as specified in the Remote Hosts menu in Setup Mode (or by some network service) and other entries separated by slashes. Details about other permitted naming techniques, using mount points and Local and Remote Mount Directories, are contained in Section 1.2.5.

This naming structure has an important special case - "HDSware" or "FXUtilities". The HDSware or FXUtilities abbreviation is used for the HDSware host and HDSware path you specified. This abbreviation is used in the configuration file for specifying the locations of the HDSware software.

1.5.2.5 Setup Mode Password Protection

The ViewStation Setup Mode has its own password protection for important settings. Some fields, such as IP address, logon information and passwords, etc. are protected from alteration by a user's Setup password. Commands in the Console shell can also be covered by this password, including all internal commands and downloadable clients. This is discussed in more detail in other sections (sections 1.2.1.5 and 1.11) of this manual.

The user password can be entered in Setup Mode from the User Preferences menu or by the remote configuration file; the security considerations of downloading the password from a configuration file are discussed in Section 1.11 along with a description of the remote configuration feature.

The User Preferences Menu has a line for entering the Configuration Password. Entering your password in this field protects your Setup Mode entries and clients from unauthorized use and alteration. If you don't enter a password here or enter an empty password, the verification process is not used.

If you have entered a password, a menu asking for your password appears whenever you try to edit a password-protected field. When you enter the correct password, you can proceed with editing the field. You may edit other protected fields in this Setup Mode session without reentering the password. If you exit Setup Mode, or iconify it, you must reenter the password in the next session to edit a protected field.

Setup Mode protected fields include most data for IP addresses, network settings like netmask and gateways, Telnet and auto-start settings, and all information for font and RGB host connections. Resetting NVM to Factory Defaults is also password protected. Specific protected fields are identified in the appropriate sections

1.5.3 Remote Configuration of the ViewStation

The ViewStation's remote configuration feature is for system environments where there are large numbers of ViewStations which need to be configurable from a remote location, either when first installed or updated or monitored at a later date by the system administrator. The remote configuration feature can also be used to specify or change things like IP addresses, RGB colormaps, fontpaths, etc. for every unit on the system at the same time without making the changes on each individual ViewStation. This is discussed in Section 1.7 of this manual.

Remote configuration allows the Setup Mode settings of any ViewStation to be downloaded from a file located on a host computer, designated the HDSware Host by default, when the ViewStation is turned on. This remote configuration feature uses the BOOTP protocol to determine the location of the configuration file, and the TFTP protocol loads the file from the configuration host to the ViewStation. These elements must be present to use the remote configuration features.

Note that the ViewStation is set at the factory to use these remote configuration feature. This means that default configuration specifies that the IP address is determined by BOOTP, that TFTP is the file transfer protocol, and that the server

code is downloaded from a boot file. If you are not using remote configuration, you must change these settings.

Using the remote configuration feature is simple for the user, since all the settings are set automatically by the system configuration file. For the system administrator, it requires setting up a configuration file and making sure the various tools for downloading the configuration are in place, including the correct BOOTP files and so on.

1.5.4 Setup Mode Main Menu

Setup Mode is a Client application program run locally by the ViewStation. It is active all the time, acting either by itself or concurrently with other clients. The only difference is that Setup's window cannot be resized, though it can be moved or iconified with a window manager client.

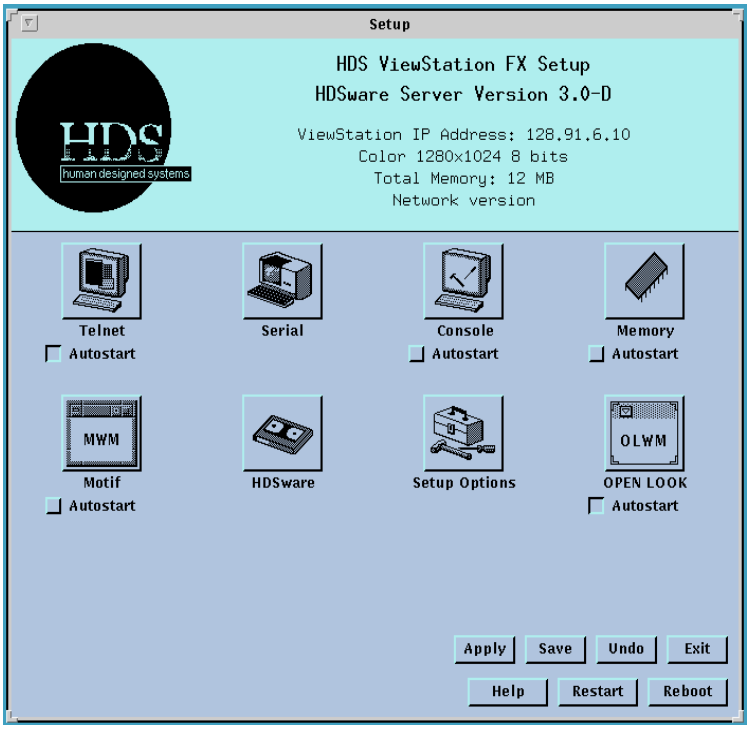
1.5.4.1 Setup Mode Heading Bar

The top bar of the Setup Mode Main Menu contains configuration information about the ViewStation, including the monitor resolution, installed memory, the firmware version, whether the ViewStation is running from its flashPROM or from the network, and the assigned IP address (after you have entered it and rebooted).

1.5.4.2 Setup Mode Selection Buttons

Setup Mode has a Main Menu containing buttons to select various ViewStation clients, or to display information about the ViewStation. Selecting the Setup Options button from the Main Menu brings other pull-down menus onto the screen with more detailed information or options. You select an item by moving the mouse cursor to the item, which will be displayed in reverse video, and clicking on it with the left mouse button.

Each selection from the Main Menu will be discussed, and the pull-down menus shown with definitions and recommendations for settings for each field. The Setup Mode Main Menu is shown below:



When you make changes to a Setup Mode field, in some cases the change will be effective immediately, such as User Preference items. In other cases, you must click on the Apply Changes button or restart the ViewStation for the settings to take effect, as with setting the IP address. The settings can be saved into Non Volatile Memory (using the Save button) so they are always available on power up.

A message line at the bottom of the Setup Mode window tells you what is necessary to make the changes effective, whether clicking on the Apply Changes button or restarting the ViewStation. A message also appears next to some fields indicating "At restart: <*this value*>".

1.5.4.3 Setup Mode Function Buttons

The cluster of buttons in the lower right corner of the Main Menu let you execute various functions, which are described below in more detail.

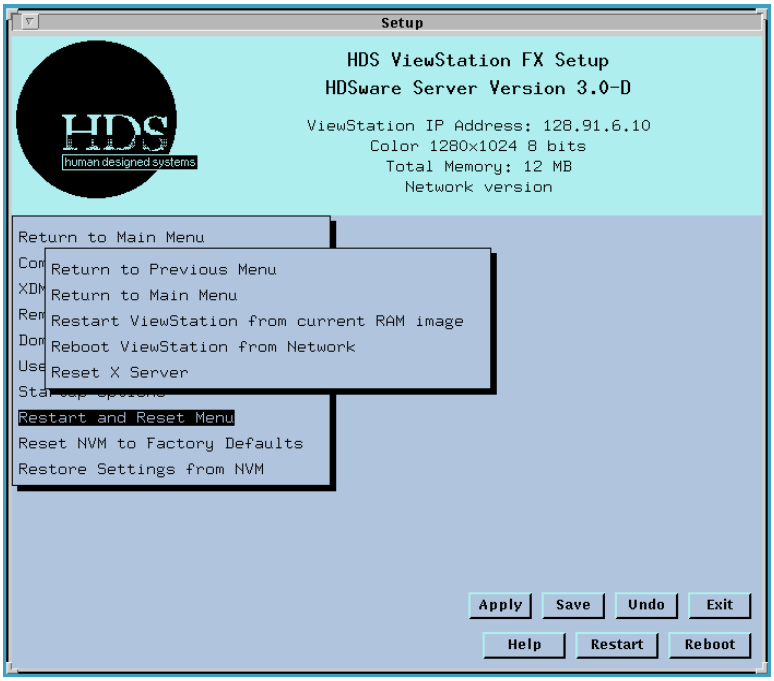
- The **Apply** button executes the changes you have made so they are effective immediately. Some settings, like the IP address, are only effective after you restart the ViewStation. The required action is shown on the bottom of the Main Menu. This button applies the changes, but you also may want to save them in NVM. If you don't save them, these applied settings will stay set only until you turn the ViewStation off.
- The **Save** button applies the changes and saves the settings you have made in Setup Mode into the ViewStation's Non Volatile Memory. These settings will be used whenever you turn the ViewStation on. You are asked to confirm this action with another pop-up menu. If you use the remote configuration feature,

some NVM settings will be set from the Configuration file. These settings and the user settings will be merged at powerup, with the config file's settings overriding the corresponding NVM settings.

- The **Undo** button allows you to remove any changes you've made in Setup Mode since your last Save in NVM or Apply Changes operation and restores the settings which were in effect before you began your current Setup Mode session. You are asked to confirm this action with another pop-up dialog.
- The **Exit** button executes an Undo Changes command and allows you to exit the Setup Mode client and return to your other processes. When you exit Setup, the Setup icon again returns to the lower right part of the screen.

  Note that if your window manager uses icons as part of its graphical interface, the ViewStation's Setup Mode will be treated like any other X client, with a header bar and icon. In that case, you probably won't want to exit Setup Mode from this menu, but just iconify it.
- The **Help** button shows you how to use the Help function. If you want more information about any button or function, move the pointer to the button and press the right mouse button. A message will appear describing the button you've selected.
- The **Restart** button terminates all the ViewStation's current clients and starts the ViewStation again. A pop-up dialog offers several choices about how you want to restart the ViewStation. A pop-up dialog asks you to confirm the selection.
- The **Reboot** button reboots the ViewStation, either from flashPROM or from the Network, as you have specified in the Startup Options menu. A pop-up dialog asks you to confirm the selection.

A more complete set of restart and reboot options, including ROM or Network restart is available from the Restart and Reset menu shown below from the Setup Options menu.



1.5.5 Starting ViewStation Clients

The ViewStation has a number of internal clients which can be started from the Setup Mode menu. These are shown with large buttons on the Main Menu of Setup Mode. They are:

- Telnet Terminal Window (using the HDSterm or ANSI window interface)
- Serial Terminal Window (using the HDSterm or ANSI window interface)
- Motif Window Manager (if loaded with your server code choice)
- OPEN LOOK Window Manager (if loaded with your server code choice)
- Console Window (using the HDSterm or ANSI window interface)
- Available Memory

Where appropriate, there is a smaller toggle button below that lets you select automatic starting of that client when you restart the ViewStation.

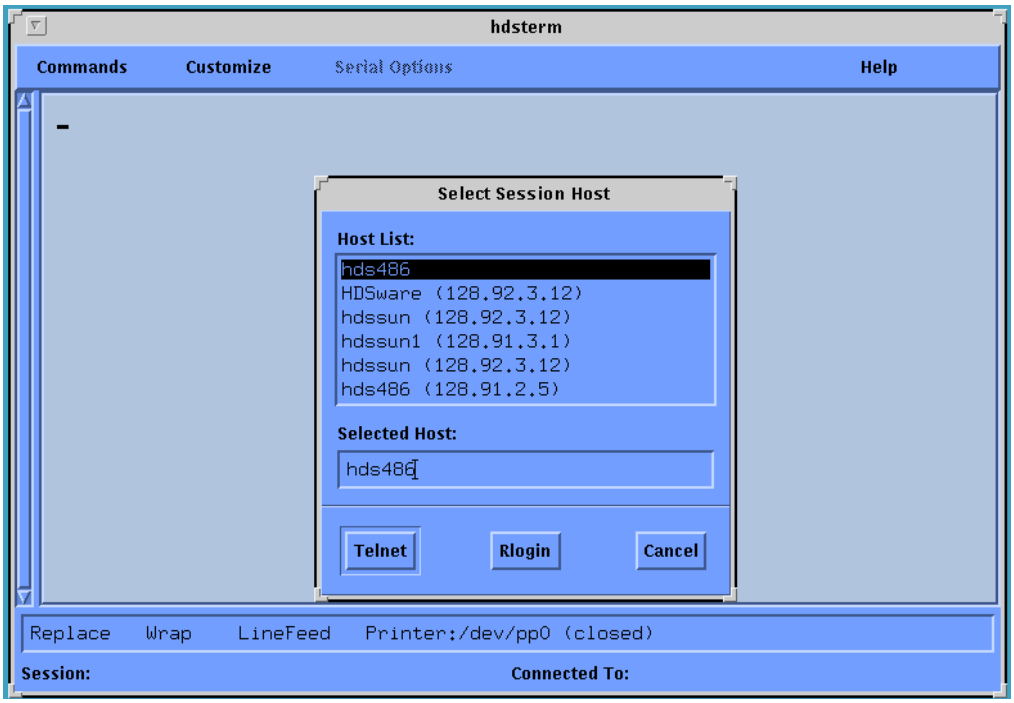
Other ViewStation clients, such as HDSperx desktop tools, are available from the window manager menus.

1.5.5.1 Telnet Terminal Window Selection

1.5.5.2 Serial Terminal Window Selection

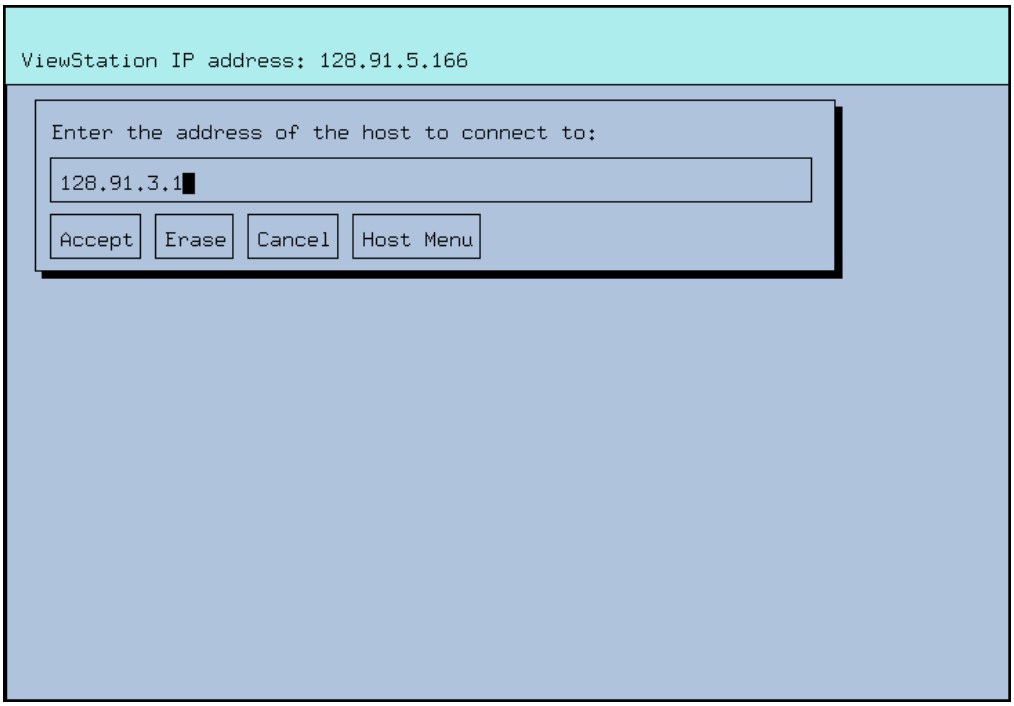
The Telnet and Serial Terminal Window buttons also use a local client, HDSterm, as their window interface. You can start individual sessions of the HDSterm client, either locally or remotely, by using shell commands. This is discussed more fully in Section 2.2 of this manual.

The HDSterm window for Telnet is shown below:



If you choose to load a server code file with no HDSterm client, the ANSI local client will be used. This is specified by the file name itself, as follows:

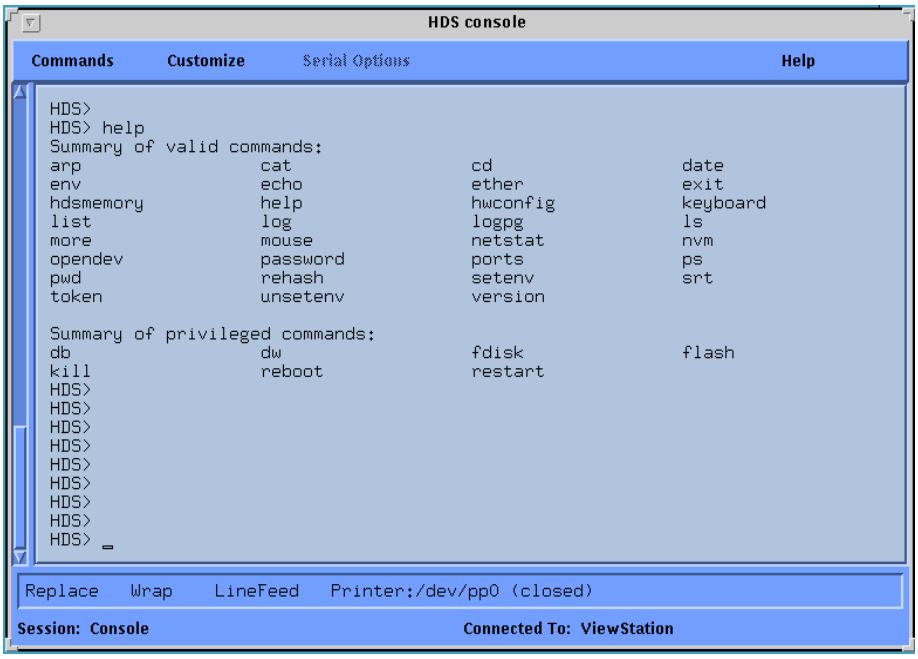
- Vnn-D.hds (the -D specifies "download" capability, which includes HDSterm, the Motif libraries for other clients (including MWM), and HDSperx clients.
- Vnn.hds uses the tty Telnet client with a more limited emulation and does not download clients. This server code uses less memory.



1.5.5.3 Console Window Selection

The Console Window button brings up a local ViewStation client which displays various diagnostic messages about the ViewStation operation and lets you run a number of ViewStation commands. This Console Window client uses an HDS-term window for its interface. This local client is discussed more fully in Section 1.2.1

The Console Window runs a shell task, providing a command line interface to the ViewStation's internal operating system. This is a powerful and convenient way to use the ViewStation's features.



1.5.5.4 Available Memory Selection

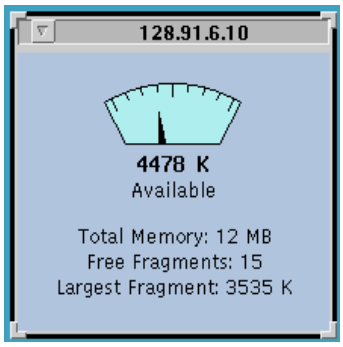
The **Available Memory** button starts a local ViewStation client. The memory window shows the memory currently available on your ViewStation; the IP address is shown in the title bar. The Available Memory client can run multiple sessions and can be displayed on other devices, so a system administrator can use this to monitor memory use.

The amount of memory installed in the ViewStation is shown along with the amount currently available. These values are reported as numbers and also with a "gasoline-guage" graph in kilobytes.

If you press “f” while the window has focus, the number of fragments and the size of the largest will also appear. Memory information about font caching can be seen in the Console Window with the “ps” command.

The available memory value can be somewhat misleading because of memory fragmentation. The observed effects of the number of memory fragments depend on the amount of memory you have installed, how long you have been running, your current settings (such as backing store), and the requirements of your clients.

The Available Memory window shows the size of the largest remaining fragment of memory, which is useful for clients which want a single block of memory. Memory fragments that are adjacent are coalesced into a single block, so you may see the number of fragments decrease when a memory block is freed.



1.5.5.5 Low Memory Warning Client

In the User Preferences menu you may start another local ViewStation client, the **Low Memory Warning** client, which displays a low memory warning whenever your memory usage reaches the value you specified in the Setup field. This is a convenient way to protect yourself from catastrophic memory failure.

1.5.5.6 Motif Window Manager

The ViewStation's local Motif window manager is currently version 1.2.3.

1.5.5.7 OPEN LOOK Virtual Window Manager

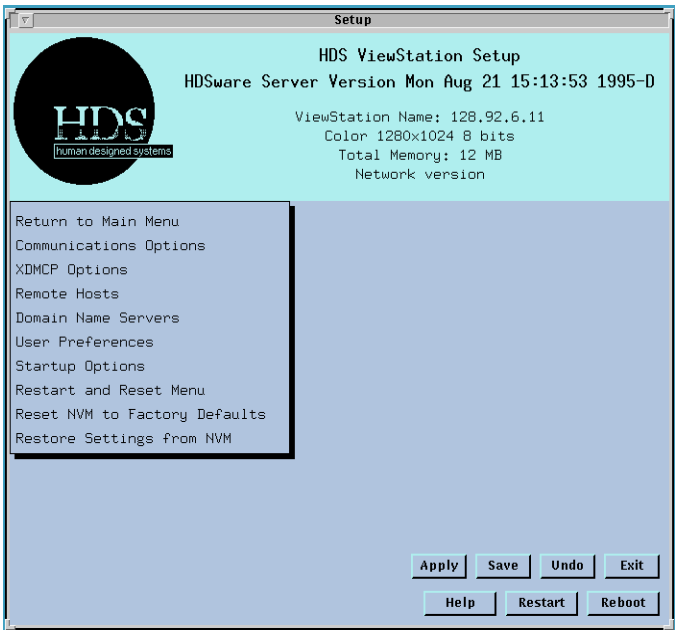
The ViewStation's local OPEN LOOK window manager is currently version 4.0 with HDSware version 3.2.

These window managers are full, licensed applications that run locally on the ViewStation. This capability is based on the power and memory of the ViewStation, and offers you considerable advantage by off-loading this important application from your host CPU and memory.

You can start the window managers by clicking on their buttons, or start them automatically by clicking on the "Run at Restart" button. Remote starting of the window manager is another option that offers additional power and flexibility. This is discussed in Sections 2.4 and 2.5 of this manual.

1.5.6 Setup Options Selection

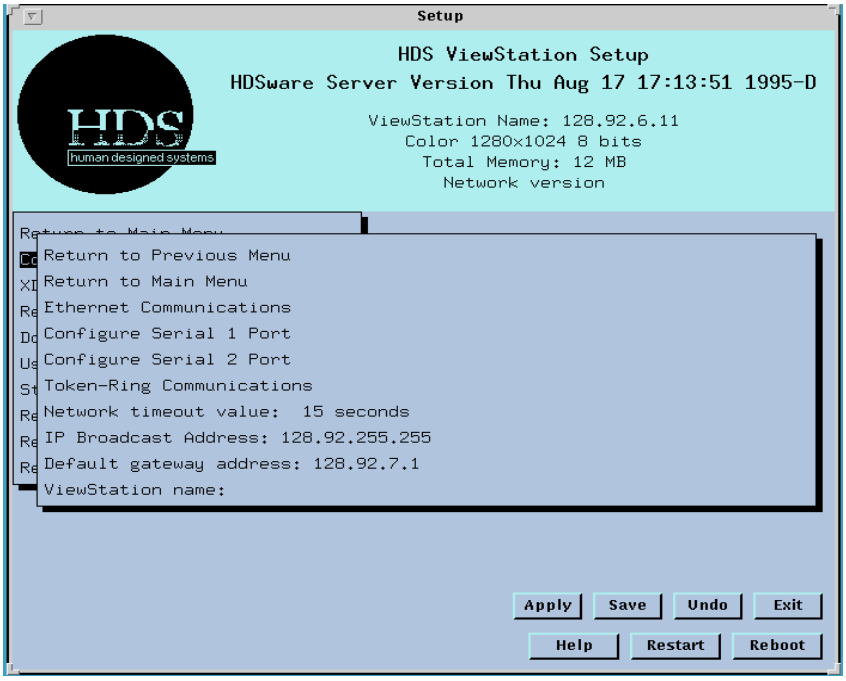
Setup Mode menus are all available from the Setup Options menu. These are discussed individually below.



Click on each of these menu lines to bring up submenus for those selections. These are discussed in detail below.

1.5.6.1 Communications Menu

The Communications Menu is shown below:



The Communications Menu gives you settings for all your ViewStation communications interfaces.

The **Ethernet Communications** menu lets you select settings for your Ethernet interface, including your IP address.

The **Configure Serial 1 Port** menu lets you to select the operation for serial port 1, the RJ11 port. These options are discussed in Section 1.4.4.

The **Configure Serial 2 Port** menu lets you to select the operation for Serial Port 2, the DB25 port on the optional Communications board. These options are discussed in Section 1.4.4.

The **Token Ring Communications** menu lets you set an IP address for your Token Ring interface as well as other settings for your Token Ring network.

The **Network timeout value** field sets the time interval that a request waits for a response. If you make a request to logon to some host on the network, the request will wait for the specified length of time before it either makes the connection, or gives up and ends the request process. If your timeout value is 15 seconds, a logon request will fail if it has to wait longer than 15 seconds. This value should be some brief time, 15 or 20 seconds, (based on the host's response time and load) so you will know a process has failed without too much waiting.

The **IP Broadcast Address** field sets the network wildcard address, typically 0.0.0.0 or 255.255.255.255. The system administrator sets the correct value for this field. Some systems prefer to use a local network broadcast, such as, 128.91.0.0 or 128.91.255.255. Your system administrator will choose this value.

The **Default Gateway address** field lets you specify a gateway host. A gateway host computer is a computer on your local network that connects to other networks. Refer also to network routing discussed in Section 1.4.5.

The **ViewStation Name** field lets you enter the ViewStation's name, which is used in name resolution on the network (if available to a name server or entered in the host's */etc/hosts* file) and also as the display name in the ViewStation's setting of the display environment variable (with keystrokes Alt-F12, etc. See Section 1.8.2.3 for more detail).

1.5.6.2 Ethernet Communications Menu

This menu lets you select settings for your Ethernet interface. Its contents are discussed in detail in Section 1.4.2.1.

1.5.6.3 Configure Serial 1 (2) Port Menu

This menu lets you configure your serial ports, either for direct serial connections or for PPP / SLIP / CSLIP communications. This menu and submenus for PPP / SLIP Options and PPP Options are discussed in Section 1.4.4.3.

1.5.6.4 Token Ring Communications Menu

This menu lets you configure your Token Ring interface. Its options are discussed in detail in Section 1.4.3.1.

1.5.6.5 XDMCP Options Selection

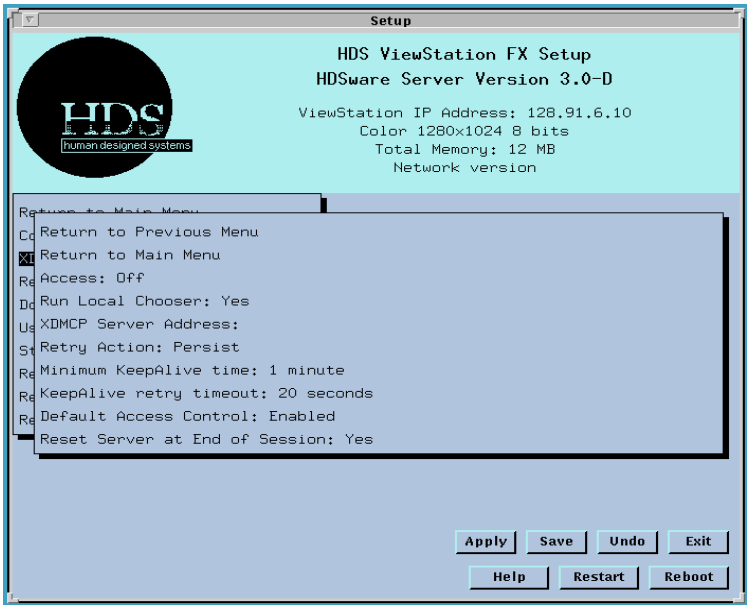
XDMCP (X Display Manager Control Protocol) is a host-based handshaking protocol to establish a connection with a host and to manage your display. It works with a host client to control your logon process, start the clients you specify, and set up configuration parameters whenever you turn on your ViewStation.

Xdm is a powerful client, and a very useful one. The X11R3 xdm managed all the displays listed in its Xservers file. With X11R4, xdm offers the user the choice of managing the display. With X11R5, additional command functionality and control is available with Xdm, now called XDMCP (Xdm Command Protocol).

Entering the ViewStation address in the Xservers file turns complete control over to xdm as with the previous version. But the XDMCP menu also offers the user the option of requesting xdm to control the display, or running without xdm. In this case, the ViewStation address should NOT be entered in the Xservers file, but these items selected in the ViewStation Setup Mode menu. With these settings the ViewStation automatically sends requests as specified to xdm to control its display whenever the ViewStation is turned on.

Using xdm and configuring its files is discussed in more detail in Section 3.3.

The XDMCP Options Menu is shown below:



The **Access Type** field specifies whether you use xdmcp and which host you want to have manage your display. There are five values for this field:

Off (default value) specifies that you do not want to use XDMCP.

Broadcast specifies that you want to accept an offer to manage your display from the first host that answers your broadcast request (your IP Broadcast Address is set in the Communications Options Menu). You can use the local HDS Chooser client to give you a menu of available hosts.

Direct specifies that you want to communicate only with the xdm client running on the host shown in the XDMCP Server Address field.

Indirect specifies that you want the host given by the XDMCP Server Address field to act as an Indirect host. This Indirect host usually forwards the request to another host computer. You can also use the local HDS Chooser client to give you a menu of available hosts.

Hosts access method specifies a file of connection options in a host file. The "Hosts" access method allows you to specify multiple hosts with individual access methods for your XDMCP request. You can set up an Xdm Hosts file by selecting the Hosts access method and entering the path to your Xdm hosts file in the "Hosts File" line. A sample file is given below:

The Xdm hosts file contains a list of hosts, one per line. Each host entry can be one of three XDMCP query types - Direct, Indirect, or Broadcast. Host names starting with an asterisk ('*') are queried via Indirect requests. Host names not starting with an asterisk are queried via Direct requests. A host name of 'BROADCAST' specifies to perform a broadcast request. Each host entry also has an associated retry timeout. The timeout parameter specifies how many seconds to send requests to a host before trying the next host in the list. The default timeout is 0 seconds. Hosts are tried in the order in which they appear in the hosts file.

Characters after a '#' character are comments.

This is a sample XDMCP hosts file:

```
# XDMCP hosts example file

*hdssun1 # Indirect request to hdssun1, timeout=0

hds486 timeout=10 # Direct request to hds486

hdshp timeout=15 # Direct request to hdshp

BROADCAST # Broadcast request
```

This file specifies that at start_time + 0 seconds, an indirect request is sent to hdssun1 and a direct request is sent to hds486. At start_time + 10 seconds, a direct request is sent to hdshp in addition to requests to hdssun1 and hds486. At start_time + 25 seconds, a broadcast request will be sent in addition to requests to hdssun1, hds486, and hdshp.

The **Run Local Chooser** field lets you select whether you run the HDS ViewStation's local Chooser client. When you use the Indirect or Broadcast access for Xdm, this client puts up a menu of available Xdm hosts. It tests the network to see which hosts respond to your Xdm request and then lets you select the one you want. This is a client running on the ViewStation. It is described in detail in Section 2.6.4. It works with both R4 and R5 Xdm versions.

The **XDMCP Server Address** field shows the IP address of the Direct or Indirect Access host shown in the previous field. The default value is 0.0.0.0.

The **Retry Action** field shows whether you want the ViewStation to keep trying to get the xdm display manager to respond or not. The values are Persist and Stop. If you want to use xdm, you probably want the requests to persist until you get a response.

The **Minimum KeepAlive time** field gives the minimum time the ViewStation waits after it has received its last message (of any kind) from the host computer before sending a KeepAlive packet to the display manager. This lets the ViewStation continue its session even if there is no activity from the host computer. The value is an integer value in minutes, with a default value of one minute. This means that the ViewStation sends a KeepAlive packet to the display manager after 1 minute of inactivity. It sends another after 2 minutes, another after 3 minutes, and so on. Any ViewStation activity from the keyboard or the mouse will wake up the session again and reset the KeepAlive time to 1 minute. The net effect is that your session stays alive even if you leave for the weekend.

The **KeepAlive retry timeout** field sets the time the ViewStation waits for a response from the host after it has sent its KeepAlive packet. This is an integer value in seconds, with a default value of 20 seconds. The ViewStation waits 20 seconds after its request for a response; if it doesn't receive one, it closes the connection because the display manager hasn't answered. After it sends the first request (at one minute of inactivity) it sends another request after 2 seconds, another after 4 seconds, and so on until the retry timeout value is reached.

The **Default Access Control** field sets the ViewStation's Access Control security, also called MIT Magic Cookie. The default setting is Enabled. Access control means that Xdm establishes an .Xauthority file with the initial logon. This file is used for security control of other login connections. Refer to Section 1.11.3 for more details and methods of using or modifying Access Control security.

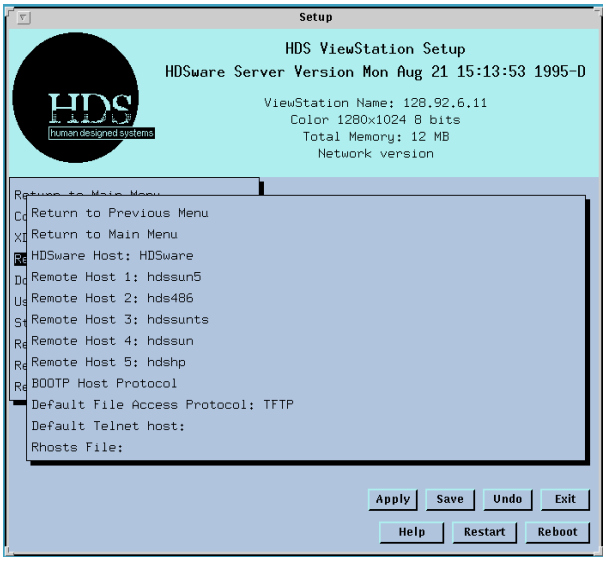
The **Reset Server at End of Session** field selects whether the ViewStation's server software resets itself at the end of an Xdm session. The default setting is Enabled. Depending on your system configuration, this may or may not be desireable, since a server reset removes a lot of the user's settings.

These settings and timeout values should be worked out with consideration of the settings for the corresponding values for xdm on the host computer.

Note also that all of these xdm variables are saved in NVM because they are used very early in the power-up process. If you change the IP Broadcast address value, which can be changed by Apply Changes for the current session only, xdm will use the IP Broadcast address set in NVM when it starts. If you are using the Broadcast address for xdm, you should save its value in NVM each time you edit the field.

1.5.6.6 Remote Hosts Menu

The Remote Hosts menu lets you specify six hosts, selecting their individual configurations, which then become the host entries on your "Chooser" Telnet menus. The menu is shown below:



Clicking on one of these lines brings up another menu for individual host selections:

The **Remote Host** list gives you submenus for each of six hosts. See next section.

The **BOOTP Host protocol** field lets you specify the file transfer protocol for your BOOTP host. If you select FTP, you can also enter the FTP access parameters in this menu. The "/BOOTPhost/" is a ViewStation mount point, so "BOOTPhost" is a recognized hostname and path (/) in path entries.

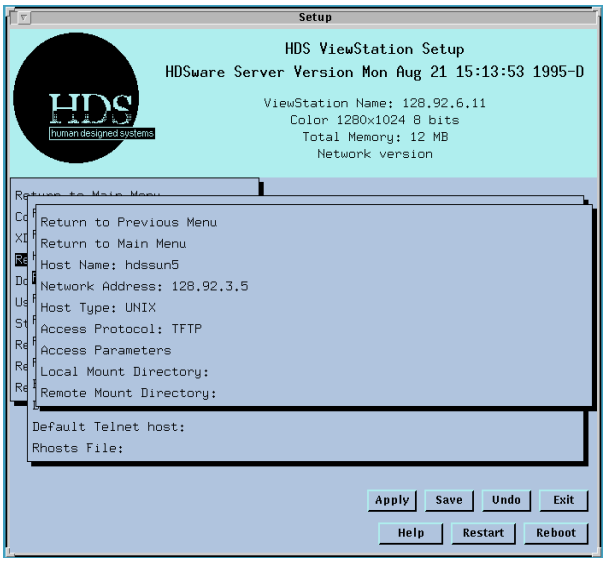
The **Default File Access Protocol** field sets the file access protocol for hosts not listed in the Remote Hosts menu. This is the access protocol the ViewStation will use when connecting to those hosts.

The **Default Telnet host** field lets you specify the host computer you want to log onto when you open a Telnet window. When you run Telnet, this computer is the one whose address is automatically put in the Telnet connection field. If you want to log onto this host automatically, you can set the toggle button at the Setup Mode Main Menu and enter the IP address of the host computer you want as your primary connection (either an Ethernet or a SLIP host) in this field. If you have Domain Nameservice running on your system, you can enter the name of the host computer instead of its IP address. Refer to the Domain Nameserver Menu for more details on name server operation.

The **Rhosts File** field lets you enter a location for your ViewStation's rhosts file. If you are doing the rhosts security yourself, it will probably be in your home directory. If the system administrator is managing the rhosts, this file will typically be in a privileged system directory. Refer to Section 1.11.2 for more details.

Remote Host Submenus

There are six submenus for remote host entries. These menus allow you to specify particular things for the six hosts you use most frequently. These six hosts appear on your Telnet selection menu. This is discussed below.



The **Host Name** field is the name of the host. If you don't specify a Network Address, this name will be used as its address and must be a valid IP address or host name (if you are using Domain Name Service or NIS).

The **Network Address** field is the IP address of this host.

The **Host Type** field specifies whether the host is Unix-based or VMS-based.

The **Access Protocol** field specifies the type of file transfer the ViewStation uses for download processes (such as loading font files or boot files) on this remote host. There are three selections, TFTP, FTP, and NFS, for Trivial File Transfer Protocol, File Transfer Protocol, and Network File Service. The first two protocols are very similar; TFTP is a simpler version of FTP. The TFTP protocol requires only a host name and filename to execute the file download, while FTP requires a valid logon name, a password, and (optionally) an account name in addition. This extra material makes the FTP protocol more secure, but it may be somewhat slower for small files overall because of this logon verification process. TFTP may give a somewhat better response for simple file transfer (such as font loading), though performance depends on a variety of host and network factors. Other factors, such as system security, may also be important in your installation.

The selection of TFTP or FTP affects the next line item, Access Parameters. If FTP is listed as the Access Protocol, the FTP username and password must be entered on the Access Parameters submenu.

The NFS file transfer protocol is more involved and requires access permission to specified file systems on various hosts. It must be configured by the system

administrator (with root permission) but gives very powerful and fast file transfer responses. This protocol is recommended for large files and file systems.

The **Access Parameters** field brings up a submenu for entering the FTP username and password. If TFTP is selected, clicking on this line brings up a message box indicating that Access Parameters are not needed.

The **Local Mount Directory** field lets you specify a directory as a local mount point. Paths to files or directories on this remote host are then specified by prefacing the remote path with the Local Mount Directory. For example, if the Local Mount Directory is /fonts, and the remote path is given as /usr/lib/X11/HDS, then the actual search path is /fonts/usr/lib/X11/HDS. If you do not specify a Local Mount Directory, the default is "hostname", where hostname is the name given in the Host Name field above. A detailed example appears in Section 1.2.5 of this manual.

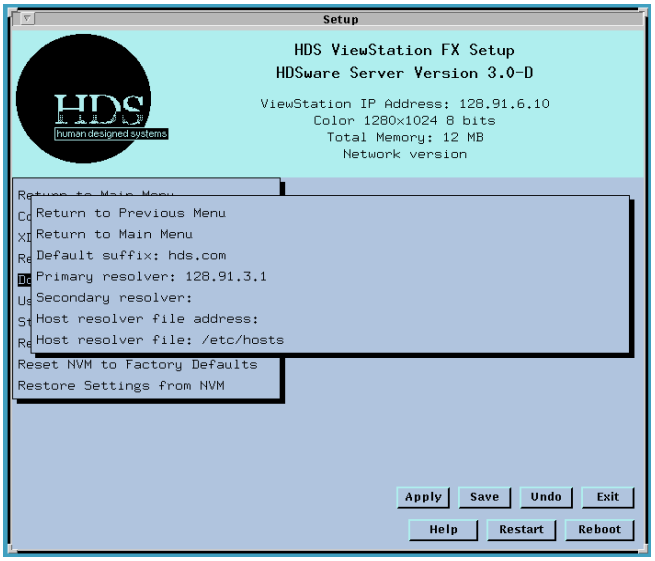
The **Remote Mount Directory** field specifies the path on the remote host corresponding to the specified local mount point. This is useful to simplify pathnames for fonts and server or configuration files.

1.5.6.7 Domain Name Servers Selection

The Domain Name Servers Menu is for users who have a Domain Name Server running on their network. The name server daemon reads data from a data file (typically */etc/named.boot*) which identifies a name with an IP address, and listens on the network for queries about names.

The ViewStation can use device names for the any of its remote hosts. If a field must have an IP address instead of a name, Setup Mode will prompt you for an IP address when you enter a name.

The Domain Name Server Menu shows:



The **Default Suffix** field gives the ViewStation information about your device names. A large system will have long host names, such as *hdssun5.softwarelab.hds.com*, so you may want to use the short name for convenience. In this case *hdssun5* would be the name you use and the rest of the name, *softwarelab.hds.com*, would be the suffix. The ViewStation automatically adds this suffix to the names you use when it sends the request to the name daemon on the network.

Be careful to name your hosts so you can communicate with other hosts in the local domain (all hosts with the same suffix) easily and not exclude other hosts and names by having the incorrect suffix.

The **Primary Resolver** field is the IP address of the host computer which is running the name daemon and will try to resolve the names you send on the network. This entry must be an IP address, not a name.

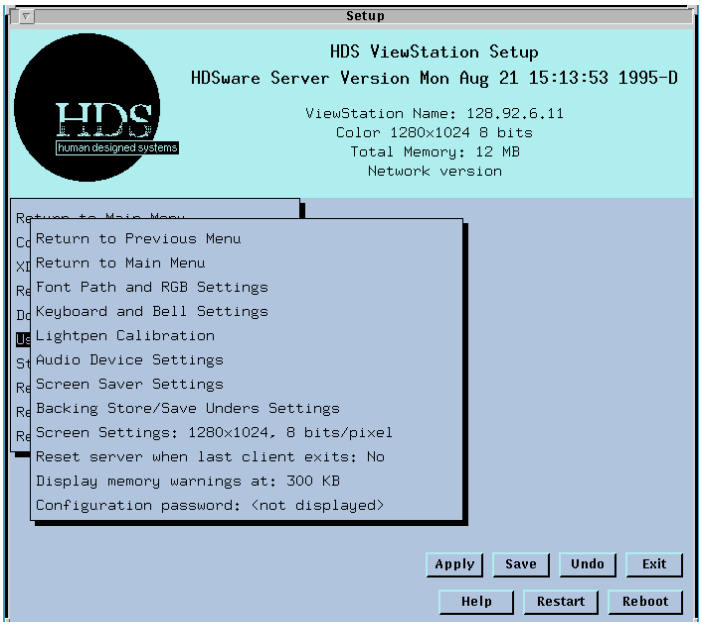
The **Secondary Resolver** field is the IP address of another host computer which listens for name queries and can interpret names. This host computer is a backup for the Primary resolver.

The **Host resolver file address** field gives the IP address of the host computer the ViewStation uses for its own address resolution. It will look at the hosts table, typically */etc/hosts*, on this computer for address resolution.

The **Host resolver file** field gives file name of the host resolver file, typically */etc/hosts*, on the specified host.

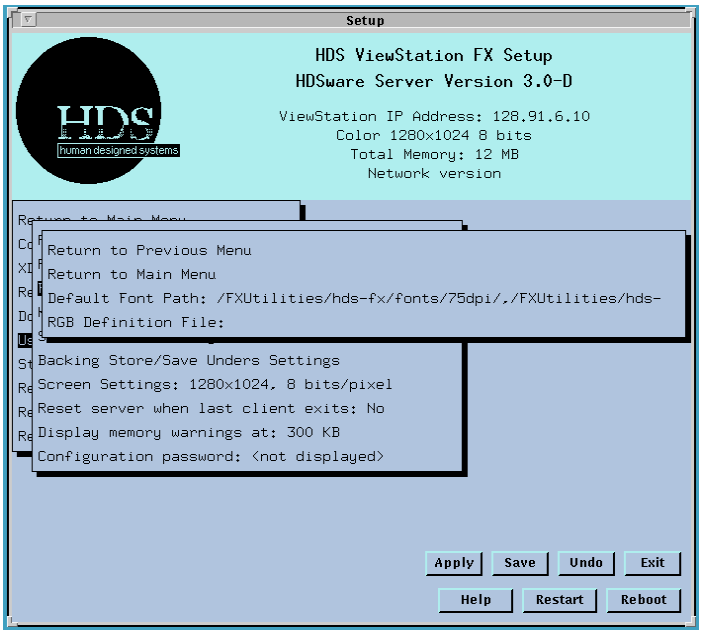
1.5.6.8 User Preferences Selection

This menu allows you to set some important user preference settings, such as fontpaths, keyboard settings, display configuration, and some memory options, such as backing store and save unders, as well as your Setup Mode password.



The **Font Path and RGB Settings** line brings up a submenu for specifying the paths to the font files and RGB color definition files you want the ViewStation to use.

The **Default Font Path** line lets you enter your fontpaths. You can enter the font-paths with the hostname first, then the path name, including additional paths, separated by a comma or space. You can also use the Local Mount Directory and Remote Mount Directory features to specify fontpaths, see Section 1.2.5. The ViewStation supports a host-based font server simply by adding the font server to the fontpath. This is discussed in greater detail in Section 1.9.3.



If you have problems registering the fontpath, or with loading fonts, refer to Section 1.9.2 and 1.9.3 where font handling is discussed in greater detail.

The **RGB Definition File** line lets you enter the path to your RGB color definition file (usually called rgb.txt). It works the same way as the fontpath field above.

Note on Downloading the RGB Color Definition File

The HDS ViewStation has the X11R4 standard color name definitions, consisting of more than 700 colors, stored in its server image. It also has the capability to download a user's custom color definition file into memory.

If you want to use a custom color definition file, follow these steps: First, create a text file containing your color definitions. This is usually named *rgb.txt*. Then load the text file into a convenient location on your system and enter the path in the RGB Server line in the User Preferences menu in Setup Mode.

A typical *rgb.txt* file has the following form:

```
<red>   <green>   <blue>      <name>

<red>   <green>   <blue>      <name>

<red>   <green>   <blue>      <name>

...

<red>   <green>   <blue>      <name>
```

where <red> <green> <blue> are intensity values for each of those colors in the range 0 to 255. The <name> is an arbitrary name given to the color. For example, the colors used for the HDS ViewStation Setup Mode screen are:

```
143    143    188      light steel blue
```

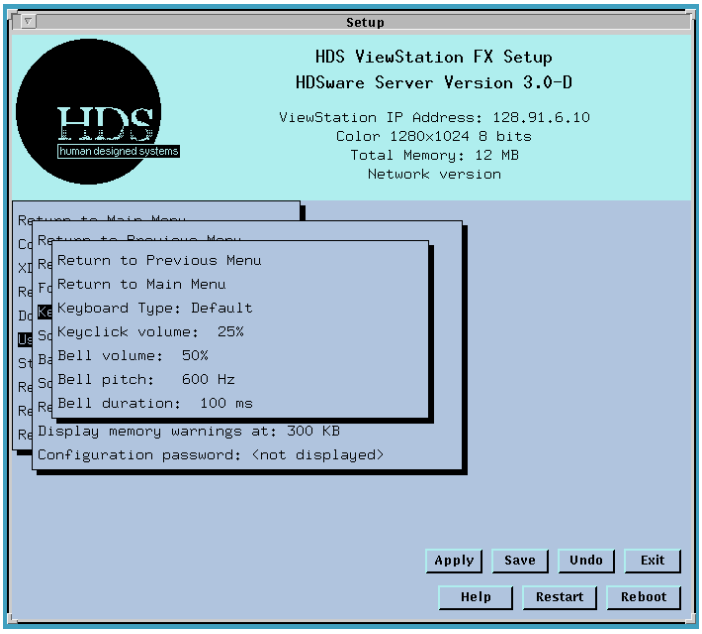
```
175     238     238       pale turquoise
```

These color names are part of the standard X11R4 color definition file. You use these names in a xterm command line to create a colored window.

These color names can also select gray shades on gray scale monitors. A gray shade is a color designated as a percentage of black / white, such as:

```
175     175     175       gray67
```

The next line brings up the **Keyboard and Bell Settings** submenu:



The **Keyboard Type** field is for multinational keyboards. The settings select special keyboard types which have keyboard maps stored in the ViewStation firmware. If you want other keyboard settings, you can generate your own **xmodmap** file. These entries are a convenience to international ViewStation users.

The **Keyclick volume** field lets you set the volume of the keyclick. Values are 0, 25%, 50%, 75%, and 100%. Clicking the mouse button on this field selects each value in turn. When you select a value, the keyclick sounds so you know its volume and can make the selection.

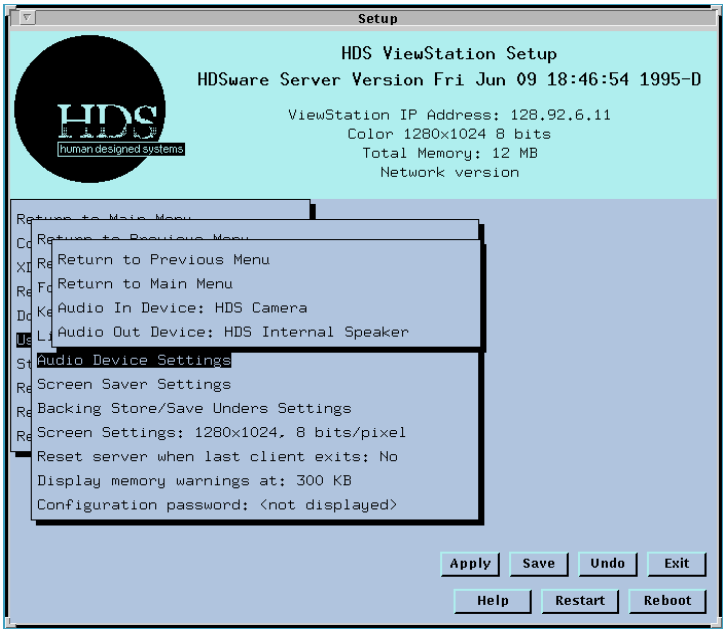
The **Bell volume** field lets you set the volume of the bell. Values are 0, 25%, 50%, 75%, and 100%. Clicking the mouse button on this field selects each value in turn. When a value is selected, the bell is rung so you can see how loud the setting is.

The **Bell pitch** field lets you select the frequency of the bell tone. Values may be from 100 Hz to 10 KHz, but values around 600 Hz are typical. When you enter a new value, save it in NVM, then return to this menu and try it by selecting different Bell volume settings, which actually rings the bell.

The **Bell duration** field lets you select the length of the bell sound. Values about 10 ms. are typical. To enter a new bell duration, move to this field and click the mouse button.

The **Lightpen Calibration** line lets you calibrate your FTG Data Systems lightpen if you have one installed.

The **Audio Device Setting** line brings up a submenu for selecting the audio input and output devices you are using on your ViewStation.



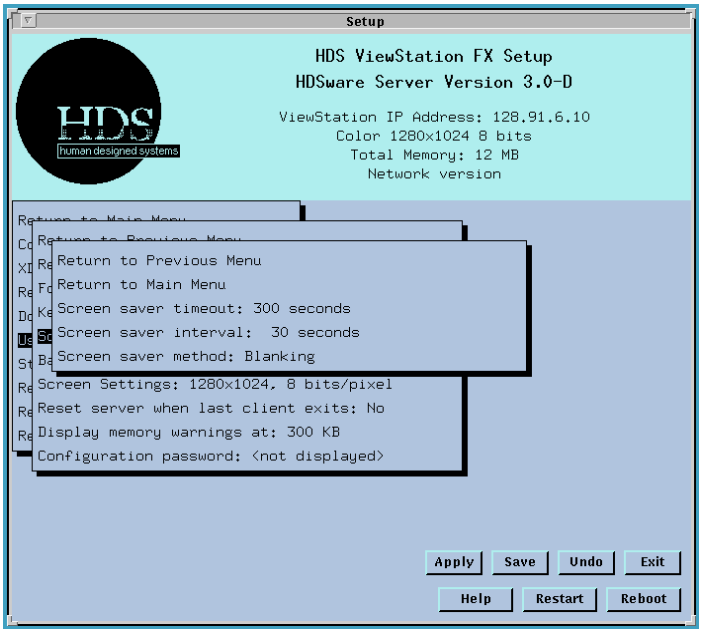
The **Audio In Device** line toggles between several input devices, HDS Camera (with built-in microphones), HDS headset, the CallPort integrated Speaker / Microphone (with echo-cancelling), HDS microphone, HDS tuner, VCR, and Line Level Audio. This setting selects the appropriate input level for the HDS audio device. These settings can also be made in the HDS audio client, see Section 2.7.1.

If you are using your own audio equipment, you will have to determine which input level settings are appropriate.

The **Audio Out Device** line toggles between several output devices, HDS internal speaker, HDS external speakers, HDS headset, and the CallPort integrated Speaker / Microphone (with echo-cancelling). These settings can also be made in the HDS audio client, see Section 2.7.1.

If you are using your own audio equipment, you will have to determine which output level settings are appropriate.

The next line, **Screen Saver Settings**, brings up a submenu for your screen saver selections.The next submenu, shown below, lets you select options for the ViewStation's Screen saver feature.



The **Screen saver timeout** field sets the amount of time the ViewStation waits after the last user activity (keyboard or mouse) before activating its screen saver.

To enter a new timeout value, move to this field and click the mouse button. Disabling the screen saver means that it is never used and the screen stays visible. Selecting a timeout value of 0 disables the screen saver. If it is disabled, the value shown on the timeout line is "Disabled".

The **Screen saver interval** field lets you set the interval between the refresh of the X logo on the screen saver low intensity screen.

The **Screen saver method** field determines whether the screen saver feature uses a the Energy Star screensaver, a blank screen, or a low intensity X logo displayed randomly at various screen locations to minimize image burn in on the display screen. The selections are Energy Star (the default), Blanking, and X logo.

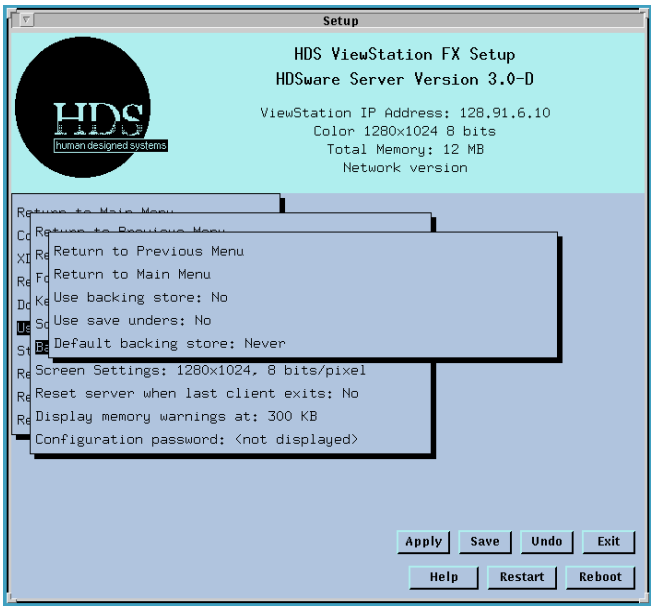
The Energy Star setting is recommended for ViewStations with an Energy Star monitor. The ViewStation with the Energy Star setting blanks the screen after the timeout value, then waits for the screensaver interval and goes to Energy Star mode (which allows the Energy Star monitor to go to a low-power setting). When the ViewStation is active again, the video signal "wakes up" the monitor and the screen display resumes. Setting the screensaver interval to 0 skips the intermediate blanking phase.

Monitors without Energy Star capability should use the ViewStation's X logo screensaver (or one of the other screensaver images from HDS screensaver client), since the Energy Star setting turns the monitor Off and On, which gives the screen an unusual appearance while starting.

Using a screen saver is recommended since a monitor can be damaged by long term operation without an active video signal.

You can also use the HDSperx Screensaver client as a screen saver. This has the added features of a screen lock and various animations for the saved screen. This is discussed fully in Section 2.6.8.

The **Backing Store/Save Unders** submenu lets you select options for the way these memory options are used on the ViewStation. These are important fields which affect the ViewStations's memory usage and performance.



The **Use backing store** field lets you select whether the ViewStation uses backing store. Another field selects which type of backing store is used and under what circumstances. The options are Yes and No; with a mouse click toggling between the two values. If you select a new value, the new value appears on the line with the note:

Use backing store: No (at reset: Yes)

to indicate that the new value will take effect only after the ViewStation server resets, or a restart occurs.

The **Use save unders** field specifies whether the save unders feature of the View-Station is used. The options are Yes and No. Clicking the mouse button toggles between the two settings. A note is added to the entry line that the new value takes effect after server reset.

Note that backing store must be enabled to use save unders. If you select Yes to use save unders, backing store is automatically enabled. In a similar way, if you disable backing store, save unders are also disabled. You may enable backing store and leave save unders disabled.

The Default backing store: field selects the type of backing store function applied to the window on the screen. There are two options: "never", which specifies that

backing store is used only if a clients specifically requests it, and "when mapped", which specifies that backing store is used for mapped windows unless the clients specifically sets the backing store attribute for this window "Not Useful". Clicking the mouse button cycles between these choices. The last line shows that the new value becomes effective only after restart.

Notes on Backing Store and Save Under Features

Backing Store is a server function that stores a copy of the contents of a window on the screen in the ViewStation's memory. When that screen window is covered by another window, its contents on the screen are erased. When the window is exposed again (requiring a screen refresh), the window contents are retrieved from the copy in the ViewStation server memory, rather than from the client (over the network). This function reduces network traffic and increases the speed of restoring the window.

Save Unders is a window storage utility similar to backing store, but typically applying to transient events, such as small pop up windows or child windows. Save unders specify that only the portion of a window under a pop up window are saved just before the child window appears, and are redisplayed when the pop up window disappears. This may be thought of as an operating property of the child window rather than the backing store memory storage, which is a property of the window whose contents are saved off screen.

In both cases, backing store and save unders are specified primarily by the client, but the capabilities must be enabled by the Server. If the fields are set to No, neither backing store nor save unders are used. If a client requests them, the Server indicates that they are not available. If the clients don't make specific requests, the ViewStation uses the setting selected in the Default Backing Store field, "never", and "when mapped". The "never" option means that backing store is not used. The "when mapped" option means that the ViewStation uses backing store on all mapped windows.

In practical terms, the speed of redrawing the window depends on the relative speed of reading from memory compared to the speed of redrawing the window from the client over the network. Redrawing from memory is adversely affected by memory fragmentation, especially with large and / or color windows, while redrawing from the network is very dependent on network traffic, host speed and loading, and so on.

Balancing all of these factors, the recommended setting for the ViewStation is both backing store and save unders Yes, but with the Default Backing Store field set to "Never". Your configuration or environment may make other choices more desireable. For example, setting the Default Backing Store to "When mapped" is useful with PPP connections, or with hosts located across one or more gateways.

This suggested setting provides the usefulness of backing store when the client requests it but minimizes the memory utilization on the ViewStation. A client request for (or prohibition of) a different type of backing store is processed correctly unless the previous field says No backing store is to be used.

The Default Backing Store setting "Always" is not supported, since it is contrary to the specification of the X Window System. Forcing Backing Store on client windows which are unmapped is not permitted behavior for an X server and may result in incorrect client behavior. The "When Mapped" setting is the strongest implicit behavior the server can impose on the client.

The **Screen Settings** field permits you to change the pixel resolution and the number of colors (or gray shades) your ViewStation uses. Your current settings are shown in the field and you can cycle through the permitted values. Of course, you must have sufficient memory installed for these changes to take place. If you don't have sufficient video memory installed, you won't be able to make any changes. For instance a ViewStation with 1280 x 1024 pixels can be used at lower resolution, such as 1152 x 900 or 1024 x 864, but a lower resolution ViewStation, such as a 1024 x 864, cannot be run at a higher resolution, such as 1280 x 1024. The same is true of the color depth. A 256 color (or gray-scale) ViewStation can be run as a monochrome unit, but the monochrome cannot be run at 256 colors.

Note that these changes require that you restart the ViewStation. Be sure you save this new value in NVM, then restart the ViewStation base from the Restart Menu.

Most users rarely use this feature. However, you might sometime need to change the **Screen Settings** field to accommodate clients that work only in one environment (such as **xgif**, which only runs on 256-color displays, or some SUN clients that only run on 1152x900 pixel displays). You could also make these changes to improve the execution speed, since 1 bit graphics run somewhat faster than 8 bit graphics. The lower screen resolutions permit a faster screen refresh and so might provide a display with reduced flicker. These are usually fairly subjective decisions and choices.

The **Reset server when last client exits** field is an important one. It determines whether the ViewStation server resets itself when its last client exits. This is often important for network operations and the orderly starting and stopping of processes on the host computer. Check with your system administrator to see if you need to set this to Yes. The default setting is No.

The **Display memory warnings at** field starts a local ViewStation client which monitors your installed memory and displays a low memory warning on the screen when your memory usage reaches the level you set in this field. You can also start another local ViewStation client, the Available Memory client, by using the **Available Memory** option on the Main Menu.

The **Configuration password** field lets you enter your password as protection for Setup Mode fields. Some important values, such as the IP address or Telnet host address, can be protected from unauthorized alteration by setting a password for Setup Mode with this field. If you don't enter a password here, any field in Setup Mode can be changed without password protection.

There are important considerations about the Configuration password when your system is using the remote configuration features of the ViewStation. The password is part of the configuration file, so questions of transport (TFTP) and pass-

word encryption are necessary for system security. These things are discussed in Section 1.11.1. The ViewStation server uses a built-in encryption utility to decode a password downloaded in an encrypted form.

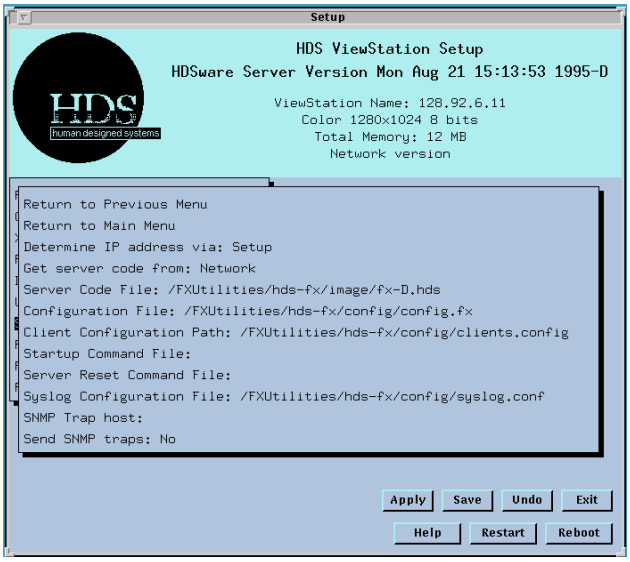
Special considerations about the password apply when you use the Reset NVM to Factory Defaults field. This field is password protected, so you must enter the password to execute the command. This resets the contents of NVM, except for the password field. If you want to clear the password, you must follow these steps.

- Go to the User Preferences menu and use your password to edit the password field
- Enter an empty password (press the *Return* key) and confirm your selection
- Save the new (empty) password in NVM

This procedure will clear the password from NVM.

1.5.6.9 Startup Options Selection

The **Startup Options** menu is shown below:



This menu selects whether you operate the ViewStation with the server software stored in flashPROM or whether you load server software from the network as a boot file. The ViewStation's boot up choices are discussed fully in Section 1.6.

The **Determine IP address via** line has three options: Setup, BOOTP, and RARP. This line lets you select how the ViewStation determines its IP address.

The **Setup** option lets you set your IP address in Setup Mode (on the HDSware or Ethernet communications menu) in the conventional way. The assigned IP address and the internal hardware address are registered on the network as needed (usually done automatically by the ARP protocol).

The **BOOTP** option (the default value) lets you use a BOOTP daemon running on a host on your network to find your IP address (and, optionally, download the server code). If the BOOTP daemon finds the IP address, it is used by the ViewStation. (If downloading the server code was specified, the boot file name and path are optionally supplied by the BOOTP daemon as part of its BOOTP response.)

The **RARP** option uses the RARP (Reverse Address Resolution Protocol) daemon to determine your ViewStation's IP address at startup. (If the ViewStation knows the IP address of another network host, it uses ARP to find the host's hardware address. If the ViewStation knows its own hardware address, it uses RARP to find its assigned IP address.) The implementation of the RARP daemon varies from system to system and there are many ways of configuring RARP so different file names and syntax rules may apply.

The next field, **Get server code from:** has two options.

The **Network** option means that the ViewStation downloads its server code from the network and puts the server code in its RAM memory. Note that the Network option uses the file transfer protocol you've specified and sends a request to the TFTP or FTP daemons running on the network to send the bootfile. The download file name can be specified either with BOOTP (which finds the boot filename as listed in the */etc/bootptab* file) or from the default boot file designated in the next entry, the FX Server Code File.

Note that a ViewStation with a hard disk drive installed uses a Network boot, but uses the ViewStation's loopback addresss (IP is 127.0.0.1) as the network destination for the file.

The **ROM** option means that the ViewStation uses the server code in its flash-PROM and loads that into its RAM memory.

The **FX Server Code File:** field specifies the location of the server image to be downloaded from the network. The path is interpreted using the Local Mount directory as described in the Remote Hosts menu section.

The **Configuration File:** line lets you enter the path to your configuration file. The path is interpreted using the Local Mount directory as described in the Remote Hosts menu section..

The default value is a path specifying your FXUtilities host and the sample configuration file supplied on the FX Utilities tape.

The **Client Configuration Path:** field shows the path to your HDSperx clients and their configuration files. The default value is specified by the FX Utilities feature and probably shouldn't be changed. This clients.config file is discussed in more detail in Section 2.1.

The **Startup Command File:** shows the path to your ViewStation startup command file, which specifies special commands and resources used by ViewStation clients. This file is only read when the ViewStation is booted. More details are discussed in Section 1.2.8. of this manual.

The **Server Reset Command File:** shows the path to your ViewStation server reset command file, which contains special commands and resources read by the ViewStation when it resets. This is the recommended place for local commands, like xmodmap, that you use to configure the ViewStation. This is discussed in more detail in Section 1.2.8.

The **Syslog Configuration File:** shows the host and location of your syslog.conf file, which specifies the type and destination for the ViewStation's error reporting.

The next two fields contain settings related to the SNMP (Simple Network Management Protocol) protocol. If your network is running SNMP, you can set these fields. If you don't use SNMP, you can ignore these fields.

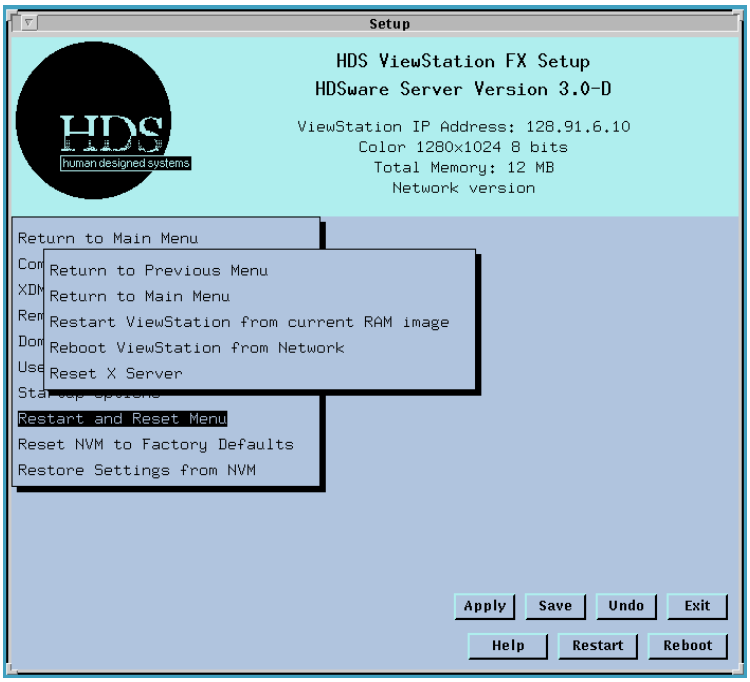
The **SNMP Trap host** field lets you enter the IP address of the host computer collecting SNMP information. This host collects and analyzes data about the network. The ViewStation sends its SNMP message (called a trap) to this location whenever the ViewStation is powered up or restarted, if the Send SNMP Traps field is set to Yes.

The **Send SNMP traps** field tells the ViewStation whether to send data to the indicated host. The values are Yes and No. If you select the Yes option, the ViewStation sends a "coldstart" SNMP trap to the SNMP host each time the ViewStation restarts. This is a message that the ViewStation is on the network and has just started up.

1.5.6.10 Restart and Reset Menu Selection

The **Restart and Reset Menu** option allows you to restart the ViewStation from the Main Menu. This closes all network connections and then restarts with the configuration you specify (RAM or ROM / Network). Restarting the ViewStation is required for registering some functions, like the IP address, entered from Setup Mode.

When you click on the **Restart and Reset Menu** field, a submenu appears with several options.



The **Restart ViewStation from current RAM image** option restarts the ViewStation using the server code located in its RAM memory. This selection is always present.

The **Restart ViewStation from ROM/Network** option goes to the boot file directory specified in the Startup Options Menu, either the ROM (the flashPROM board) or the Network, and loads the server code again, and then runs from the new code. In each case, this is a complete restart, just as if you had turned the ViewStation Off and On again.

The **Reset X Server** option lets you reset the ViewStation's server software. This command resets things the server has set up, like running clients, loading fonts, or assigning root window properties, but does not reset or reread internal server settings, like the Reset Server Command file, the Startup Command file, or the clients.config file.

1.5.6.11 Reset NVM to Factory Defaults Selection

The **Reset NVM to Factory Defaults** selection allows you to change your NVM settings to their original factory settings. This is a very drastic step and will erase all your settings, including the ViewStation's IP address. You would probably only do this if you were moving the ViewStation to a new location or had some problem with conflicting settings.

There are two fields which are not reset to "Factory Defaults", but retain their current NVM values:

- Screen Settings - The current monitor settings are preserved.

■ Setup Mode Password - This is changed by the procedure outlined below.

The **Reset NVM to Factory Defaults** field is password protected. Clicking on this field brings up a dialog box asking you to confirm this command and giving you a warning message.

Special considerations about changing the password apply when you use the Reset NVM to Factory Defaults field. This reset field is password protected, so you must enter the password to execute the command. This resets the contents of NVM, except for the password field (and the fields mentioned above). If you want to clear the password, you must follow these steps:

■ Go to the User Preferences menu and use your password to edit the password field

■ Enter an empty password (press the *Return* key) and confirm your selection

■ Save the new (empty) password in NVM

■ Run the Reset NVM to Factory Defaults command.

This procedure will clear the password from NVM and let you reset everything to factory defaults.

You can also reset your NVM to Factory Default settings from Monitor Mode with the "factory" command.

1.5.6.12 Restore Settings from NVM

This selection lets you reset your current NVM settings to the values you last Saved in NVM. This is a convenient way to undo all the temporary changes you might have made.

HDS ViewStation: Starting the ViewStation

This chapter discusses various methods of starting the ViewStation and some common problems sometimes encountered with starting the ViewStation.

Section 1.0 - HDS ViewStation

1.6 Starting the ViewStation

There are three ways to start the ViewStation These are discussed in the *HDS ViewStation Series Owner's Manual*, Chapter 3. These starting methods use:

- Local Storage (flashPROM or internal hard disk)
- Monitor Mode
- BOOTP (network service on your host)

These methods and related considerations are discussed below. ViewStations with 4 MB of memory boot with a special configuration file, see Section 1.6.2.5. Booting from VMS hosts is discussed in Section 1.6.2.6.

1.6.1 Starting from Local Storage

If your ViewStation has a flashPROM card or a disk drive installed, all you have to do is turn on the ViewStation and the monitor. The ViewStation's software is stored in the local memory and is loaded automatically. You will see boot information on a white startup screen and then the root window with the Setup icon in the lower right corner.

1.6.1.1 Starting from a ViewStation Hard Disk

If you have an HDS ViewStation with an internal hard disk drive, your startup procedure is the same as with a flashPROM card.

1.6.2 Starting from Monitor Mode

If you don't have local memory for booting your ViewStation and you don't have the BOOTP network service running, it is convenient to start the ViewStation with Monitor Mode. At the bottom of the white startup screen a message appears: "Press any key to retry or press m to enter Monitor Mode". When you press "m", you enter Monitor Mode and a screen appears with a list of Monitor Mode commands. Follow this example to see how Monitor Mode operates.

1.6.2.1 Monitor Mode Key Mappings

When you first turn on the ViewStation, the white screen you see is called the Boot Monitor. If you want to go directly to Monitor Mode to make your settings, you can press the "m" key while the "Initializing" message is displayed (3 seconds); otherwise you have to wait until the boot process finishes in order to get into Monitor Mode.

There are two other keystrokes that can be useful, a "Pause" function to pause (and resume) the boot process, and a "Debug message" function to display more detailed messages (two levels of debug messages, activated with one or two keypresses).

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Pause (Resume)	Pause	F1 (Hold)	Pause	Pause
Debug mess. (1x or 2x)	PrntScrn	Help	PrSc	PrintScreen

1.6.2.2 Monitor Mode Screen

Monitor Mode is a special ViewStation utility to help you get started. When you enter Monitor Mode a screen appears with a list of Monitor mode commands:

```
HDS ViewStation Boot Monitor


Current Settings:
ViewStation IP Address:1.1.1.1
Gateway: 0.0.0.0
Netmask: 0.0.0.0
HDSware Host Address: 0.0.0.0
HDSware Directory:
Server Code File:
Configuration File: /FXUtilities/hds-fx/config/config.fx
Default File Protocol: TFTP
Ethernet Type: Thin
```

```
Command Summary:
ip xxx.xxx.xxx.xxx - set ViewStation IP address
gw xxx.xxx.xxx.xxx - set ethernet gateway address
nm xxx.xxx.xxx.xxx - set ethernet netmask
hh address - set HDSware host address
hd directory - set HDSware directory
sc filename - set Server Code file
cf filename - set Configuration file
set resource value - set configuration resource
show resource- display configuration resource
tftp or ftp or NFS - use TFTP or anonymous FTP or NFS
for default file accesses
thin or thick - set Thin or Thick ethernet
factory - reset NVM to factory defaults
boot - save settings in NVM and reboot
:
```

You can start your ViewStation with just three entries - **ip**, **hh**, and **hd**; other commands are optional, but may be necessary in your environment. This minimal boot procedure is recommended, since other settings, such as entries in the configuration file, will override these entries and may prevent booting by giving conflicting settings. If you have not modified the configuration file, you can enter the **hh** and **hd** values (this will automatically load the configuration file), and the correct paths to this file will be found.

You enter commands from the keyboard and they are displayed at the ":" prompt at the bottom of the screen. As you enter values, they appear in the "Current Settings:" block at the top so you can see your entries. When you are finished, type "boot" to save the settings in NVM and reboot the ViewStation.

1.6.2.3 Monitor Mode Commands

This is a list of Monitor Mode commands and their operations.

ip *nnn.nnn.nnn.nnn*	This sets the ViewStation IP address
gw *nnn.nnn.nnn.nnn*	This sets the ViewStation gateway address
nm *nnn.nnn.nnn.nnn*	This sets the ViewStation netmask address
hh *nnn.nnn.nnn.nnn*	This sets the HDSware host address
hd <*path*>	This sets the path to the HDSware directory
sc <*filename*>	This sets the server code file as: </hostIP/path/servercodefile>

cf <*filename*>	This sets the configuration file as: </hostIP/path/configfile>
set <*resource value*>	This sets the configuration resource specified
show <*resource value*>	This displays the configuration resource specified
tftp	This selects the TFTP file transfer protocol (default)
ftp	This selects anonymous FTP as the file transfer protocol
nfs	This selects the NFS file transfer protocol
thin	This sets the Ethernet type to "ThinWire" (default)
thick	This sets Ethernet type to "ThickWire" (default = ThinWire)
factory	This resets NVM to Factory Default settings
boot	This saves your settings in NVM; reboots the ViewStation

These commands let you select important ViewStation settings and control the boot process. Note that if you are running on a Token Ring network with a Token Ring interface card installed in the ViewStation, the IP and netmask fields accept Token Ring addresses (such as *ip tr1 nnn.nnn.nnn.nnn*), refer to Section 1.4.3.1 for more details on Token Ring settings.

Other settings can be made in the same way. Monitor Mode is not intended to substitute for Setup Mode, but to provide a way for starting your ViewStation if flashPROM or BOOTP aren't available. Other variables may be necessary for your system, such as a gateway address and netmask setting, or a configuration file location. If you are working with a unit that already has data in NVM (Non-Volatile Memory), you can clear the NVM with the "factory" command.

The "show" and "set" commands let you set any configuration resource, such as:

show HDSsetup.ipaddress

and the ViewStation will report the current HDSsetup.ipaddress setting.

In the same way, you can set any configuration resource, such as:

set HDSsetup.config.file /128.92.1.3/usr/lib/X11/HDSware/hds-fx/config/config.fx

If you are working with a ViewStation that already has data in NVM (Non-Volatile Memory), you can clear the NVM with the "factory" command.

1.6.2.4 Monitor Mode Sample Entries

The syntax of these commands is very important. Spelling and upper/lowercase letters are important.

The **ip** entry is the network address assigned by your system administrator; it must be a unique address and it must be appropriate for your local network. It will have four fields of numbers between 0-255 separated by periods.

ip 128.92.61.144

The **hh** entry is the host address where you loaded the HDSware tape. Its form is the same as the IP field.

hh 128.92.60.100

The **hd** entry is the path on the HDSware host to the top of the HDSware directory tree. This is the directory where you loaded the tape. This might be any directory location, but the path should be root-based and go to the /hds-fx directory level. The path might be:

/home/HDSware or */usr2/HDStape or /usr/lib/X11/HDSware*

The important thing to realize is that the ViewStation will add these entries to each other to create a full path, as follows:

/<hh host>/<hd path>/<HDSware directory tree>

Make sure that the paths are complete and don't overlap or the boot procedure will fail.

The **sc** entry is the server code location. This must be a full path specification.

sc /128.92.60.100/home/HDSware/hds-fx/image/V30-D.hds

Note the leading slash and the slash separators. This is the hh host field, the hd path field, and the HDSware location of the boot file. This is the full specification of the server code file location.

Note that a potential problem remains which might prevent ViewStation booting. Monitor mode may correctly specify a path but the file transfer protocol (TFTP) may not be running on your host or it may have some security restriction that would prevent the file being transfered to the ViewStation for booting. This problem and some suggested solutions are discussed in the *HDS ViewStation Owner's Manual*.

1.6.2.5 Booting ViewStations with 4 MB Memory

If you have a ViewStation with only 4MB of memory installed, your ViewStation may not boot with the default settings but show a dark screen with periodic flashing after the white boot screens. Sometimes it may boot correctly and give you a root window with the Setup icon, but it will have so little memory available that you get memory warning messages when you try to start anything.

To boot the 4 MB ViewStation you must use the server code version V32.hds *(with the appropriate version number, but not the V32-D.hds version)*. The Vnn.hds version does not use downloadable clients. This is a fully-functional server with Telnet but does not offer HDSterm (Telnet with VT320 emulation), local window managers, or other HDSperx local clients. These clients can be run from the host computer rather than from the ViewStation.

1. Turn on the ViewStation and go into Monitor Mode (press "m" when the white screen says "Initializing").

2. If you have HDSware version 3.2, use the configuration file:
./hds-fx/config/config.4m

If you don't have HDSware 3.2, edit your configuration file (usually /FXUtilities/hds-fx/config/config.fx) with this line for the server code file:

HDSsetup.ServerCode.file: /FXUtilities/hds-fx/image/V31.hds *[use the appropriate version number]*

3. Enter these values in Monitor Mode:

factory *[this resets the ViewStation to factory defaults]*

ip nnn.nnn.nnn.nnn *[this sets the ViewStation's IP address which is assigned by your system administrator]*

hh nnn.nnn.nnn.nnn *[this sets the HDSware IP address]*

hd <directory path> *[this sets the HDSware directory path]*

boot *[this saves these values and boots the ViewStation]*

This reads the edited configuration file with the V31.hds image file. If you have HDSware version 3.2, enter the values above and include a configuration file line before the boot command:

cf /FXUtilities/hds-fx/config/config.4m

4. These steps boot the ViewStation with the smaller server code file and you will see the root window with the Setup icon.

1.6.2.6 Booting the ViewStation from a VMS Host

If you want to boot the ViewStation from a VMS host, you must first load the HDSware on the host, as in Section 4.3. You must have HDSware version 3.0.4 or higher (in both server code and bootprom) to use this VMS booting procedure.

Use of NFS file transfer is recommended for VMS systems. This is discussed in Section 4.3.4 of this manual.

When you start the ViewStation, press "m" at the beginning of the white screen to enter Monitor Mode. In Monitor Mode, make these settings:

Set the ViewStation's IP address:	**ip** *nnn.nnn.nnn.nnn* also set the netmask and gateway addresses, if needed
Set the HDSware VMS host address:	**hh** *nnn.nnn.nnn.nnn*
Set the HDSware path, VMS syntax:	**hd** <*directory name*> such as: **hd** HDS$HOME
Set the Configuration file:	**cf** /FXUtilities/[hds-fx.config]config.fx note that "FXUtilities" is an abbreviation for the path: **hh** + **hd**

Set the host type to VMS: **set** HDSsetup.FXUtilities.hosttype: VMS

If needed, you can also set the file transfer protocol to FTP, including the entry of a username and password with the "set" command with the appropriate configuration resource names.

Select the FTP protocol: **set** HDSsetup.FXUtilities.protocol: FTP

Set the username (for FTP): **set** HDSsetup.FXUtilities.username: <username>

Set the password (for FTP): **set** HDSsetup.FXUtilities.password: <password>

When all of these settings are made, you can boot the ViewStation.

Boot the ViewStation: **boot**

This boots the ViewStation by loading the server code file, whose location is specified in the configuration file, by means of the file transfer protocol you specified. Both of these conditions must be met - that is, the file must be in the location entered (the configuration file knows the correct location unless the HDSware tree has been modified), and the file transfer protocol (TFTP, FTP, or NFS) must be running and have access to that location.

1.6.3 Starting with BOOTP

BOOTP is a network service program (refer to Section 3.2 of this manual and RFC 951, RFC 1048, and RFC 1084 for specific details) running on your host computer that listens for startup or boot requests. The ViewStation will respond to bootp messages from either an Ethernet or a Token Ring network. When the host receives a boot request, it looks in the bootp table (a file called */etc/bootptab*) for information about the machine making the request.

The fields in this file have an abbreviation for the entry name (such as **ha** for hardware address, and **ip** for IP address, etc.) and an associated value. These values are sent with the bootp response and received by the ViewStation to start its boot process. Check the syntax for these files on your particular system.

1.6.3.1 BOOTP Command Fields

The BOOTP response is limited to 1024 bytes. The ViewStation uses the following Bootp fields:

:ha= hardware address

example: :ha=008096006a5c:\

:ip= IP address

example: :ip=128.91.6.15:\

:hd= home directory

example: :hd=/usr/HDSware/hds-fx/image:\

:bf= bootfile < *filename (limited to 128 chars)* >

example: :bf=V303-D.hds:\

:gw=	gateway address example: :gw=128.91.2.5:\
:sm=	subnet mask example: :sm=255.255.0.0:\
:hn=	ViewStation's name (from Comm. menu in Setup)
:ds=	domain nameserver (entry is one or two addresses separated by a space) example: :ds=128.91.3.12 128.91.3.1:\ more than two entries are ignored; if only one is used, secondary is zero
:T128=	(vendor tag) configuration password example: :T128="ApollO":\
:T129=	(vendor tag) configuration filename example: :T129="config.fx":\
:T130=	(vendor tag) configuration directory example: :T130="/HDSware/hds-fx/config":\

BOOTP may be used to load the configuration file (containing Setup Mode values) or load only the ViewStation's IP address and boot file entry (containing the server code) or both. Do not confuse these functions.

These entries and values, along with any other BOOTP fields your system may use, are retrieved from the bootp table and sent to the ViewStation in the bootp response. The configuration path and filename are also sent if the vendor tag fields are set. This configuration file contains individual resources and values for Setup Mode fields, which are given in detail in a later section.

When you turn on the ViewStation, it sends the BOOTP request automatically. BOOTP then sets the IP address for the ViewStation and loads the software file. You have a root window with the Setup icon in the lower right corner.

1.6.3.2 Problems with BOOTP Starting

If there is a problem with Bootp, try Monitor Mode, or look for these things:

- Verify that BOOTP is running on your system.
- Verify that your entries in the bootptab file are correct; verify the file syntax and the paths and filenames.
- Verify that the file transfer protocol, TFTP by default, is running so the file can be transferred correctly.
- If you are running TFTP in secure mode, remember that this changes the root directory to */tftpboot* and this must be reflected in your hd and T130 entries, which should then start with */tftpboot*.
- Note that there is a 64 byte limit (actually 60 bytes) on "vendor" fields, which includes T128, T129, T130, gw, sm, and ds.

If you have problems or questions about starting BOOTP, refer to Section 3.2.

1.6.3.3 Sample *bootptab* File

This file would normally be located in */etc/bootptab*. Note that lines marked # are comment lines. Note also that the passwords shown here are in clear text, since the bootp table can be read-protected on the system. (A configuration file specified by BOOTP cannot be read-protected, since it must be readable by TFTP, and so passwords occuring there must be encrypted. Configuration files loaded directly by the ViewStation can be read-protected, and so can use clear-text passwords.) The configuration file is loaded from the host which sent the bootp response. This security issue is discussed in Section 1.11.1.5 of this manual.

This sample file contains a section for global settings, which would apply to everything on the system and / or also a subnet entry, which would apply to smaller divisions of the entire system. The entries for individual machines are entered last. Refer to the RFC documents and your system documents for the specific details of *booptab* file syntax. This is sometimes a difficult problem; for instance, some systems require the boot file name to have a leading slash, others refuse to read it with the leading slash, and some allow :hd and :bf on the same line, others do not, etc.

```
# First, we define a system entry (such as subnet2) which
# specifies the things every host on that subnet uses.

subnet2.default:\

:hd=/home/HDSware/hds-fx/image:\

:bf=V30-D.hds:\

:T128="Galahad":\

:T129="test.3.cfg":\

:T130="/home/hdssun1/boot":

# Now come the individual ViewStations

marketing#1:\

:tc=subnet2.default:\

:ht=ethernet:\

:ha=008096000167:\

:ip=128.91.5.68:

marketing#2:\

:tc=subnet2.default:\

:hd=:bf=/home/HDSware/hds-fx/image/V30.hds:\

:ht=ethernet:\
```

```
:ha=00809600015d:\
:ip=128.91.5.13:
```

This sample file has sample entries for Bootp. Note that the vendor tag fields are present and that the configuration file called for the marketing#2 unit is test.3.cfg.

When you use BOOTP to specify the download of the boot file (server code), several different combinations of bootp fields can be used to attempt the download process. These combinations are listed below in the order they are tried, with the bootp field names shown in bold:

1. **T130** and **T129**
2. **T130** and hds.setup
3. **hd** and **T129** (if there is no T130 field)
4. path of **hd** and **hds.setup** (if there are no T129 or T130 fields)

1.6.4 Startup Considerations

There are a number of considerations involved with your choice of starting the ViewStation. One consideration is convenience, another is central control and system administration. Some of these questions are discussed below.

1.6.4.1 Local storage Benefits

There are two types of local storage on the HDS ViewStation, ROM storage and hard disk storage, with several size options for each. Its advantage is convenience, its disadvantage is cost, which is relatively small except in large installations.

ROM Storage

The ViewStation's flashPROM storage is typically 2 MB, with 1.75 MB allocated for server code and .25 MB allocated for the bootprom code. The bootprom may also be present as a PLCC chip on the main circuit board. The FlashPROM is reprogrammable so you can always update your HDSware versions.

ROM storage is also available with the PCMCIA board option, with ROM cards in 2.5 MB and 5.0 MB storage capacity available.

Hard Disk Storage

The PCMCIA board can also use hard disks, with both 20 MB and 40 MB capacity available. These disks are shipped with the HDSware already installed. This means that the ViewStation has all its software resident, including server code, local clients, fonts, and so on. This is especially useful for ViewStations without Ethernet connections to a host computer, such as ViewStations using a PPP connection from a remote site. This permits the ViewStation to boot, load its clients, and load fonts without burdening the PPP line.

1.6.4.2 BOOTP Benefits

The BOOTP protocol has a number of benefits, especially in a large installation with central administration. BOOTP offers the advantage of central control, both

for setting the IP addresses and for controlling the configuration through the ViewStation's configuration files. Passwords can also be controlled from BOOTP.

The only disadvantage to BOOTP is the effort required to start and maintain it, which is a small effort for experienced personnel in administering a large system.

1.6.4.3 Monitor Mode Benefits

Monitor mode has the advantage of guaranteed starting in any situation. Its only disadvantages are that it cannot be done remotely, and that it has a limited user interface.

1.6.5 Determining the IP Address

An important consideration about starting the ViewStation is determining its IP address. This is a major concern for system administrators and there are several methods available to control this process.

The ViewStation's Setup Mode, Startup Options menu, selects the method of IP address determination, either Setup, BOOTP, or RARP.

1.6.5.1 Setup Mode IP Address Selection

The simplest way to set the IP address is through Setup Mode in the HDSware menu, or by Monitor Mode. In both these cases the IP address is stored in NVM. This relies on the user obtaining a unique, correct address and entering it manually.

1.6.5.2 IP Addressing with BOOTP

The BOOTP protocol sets the IP address based on entries in its *bootptab* file by matching the hardware address sent by the ViewStation in its bootp request. This permits the system administrator to control IP addresses remotely. Since the hardware address for each ViewStation is built into its serial number (see Section 1.1.12.1 of this manual), the system administrator could add entries to the bootptab file simply by looking at the shipping documents, and the IP address would be correctly assigned even if the ViewStation in question was started at a remote location.

1.6.5.3 IP Addressing with RARP

The RARP (Reverse Address Resolution Protocol) is another mechanism to set the IP address. RARP doesn't have the capability to specify a boot file. (If you want to use the download option, you should use the BOOTP daemon for both IP address resolution and the boot file location. If you just want to download the server code, you can use the Startup Options menu directly without using BOOTP at all.)

The RARP (Reverse Address Resolution Protocol) daemon needs two files, one to match the hardware address to a name and another to match the name to an assigned IP address. (In the example below, these files are named */etc/ethers* and */etc/hosts* respectively. These file names may be different on your system.)

When you turn the ViewStation on, it sends a broadcast message containing its hardware address out on the network. The RARP daemon receives this message,

checks the */etc/ethers* file to match that hardware address to a name, then checks that name in the */etc/hosts* file to get the IP address to return to the broadcast source. Correct operation of the RARP protocol obviously requires that the information for the ViewStation be entered correctly, particularly that the names used are identical (most systems don't have a rigid syntax for the names, but whatever is used must be used consistently). Changes to these files are registered only when the RARP daemon starts, so you will have to shut it down and restart it whenever you change the files.

1.6.5.4 Sun NIS IP Addressing

Recent versions of the SUN operating system use NIS, Network Information Service (formerly called SUN Yellow Pages), instead of RARP. Refer to the SUN documentation for details on NIS operation, but note that the */etc/ethers* and */etc/hosts* files are replaced by NIS maps. These maps can be created automatically from the RARP files with SUN conversion routines, usually found in */var/yp*.

1.6.5.5 RARP Protocol Example

This example deals with one particular system of network operation. Each system is different, with different host configurations, network arrangements, and operating system variables. Check your system carefully to see what its names, configurations, and requirements are. Use the **man** command to look at your online documentation to see what the arguments for the commands are and the syntax for the required files. Refer also to the section of your documents called "Networking Services", or something like that, for additional information.

The RARP protocol requires two files, one to match the hardware address to a name, called */etc/ethers*, and one to match the name with an IP address, called */etc/hosts*. These two files are shown below:

The ethernet file is */etc/ethers*. Its form is as follows:

<Ethernet hardware address> <white space> <host name>

or, specifically:

00:80:96:00:44:a5 hds12

The hosts file is */etc/hosts*. Its form is as follows:

<IP address> <white space> <host name>

or, specifically:

128.91.4.12 hds12

The host name, for this example, has no definite syntax, but must be identical to the entry in the */etc/ethers* file. The syntax and punctuation of the addresses may also vary from system to system.

The RARP daemon must also be configured in a similar way, with files such as a *rarp.rc* and / or an *rarp.ifconfig*. The RARP daemon, in this example, is found in */usr/etc/rarpd*. These names and locations depend on your system. Changes to

these files are only registered when the daemon starts, so you will have to kill the daemon and restart it whenever you make changes.

1.6.6 Network-based ViewStation Operation

The ViewStation can load its server code from ROM (or hard disk) or it can start by loading its server code from the network, either from a BOOTP-specified location or from a location specified internally. This internal specification can either be from Monitor Mode, or set in the Setup Mode, Startup Options menu.

In its factory configuration the HDS ViewStation uses its download (network-based) operating mode. This download operation has two basic parts:

- Determine the ViewStation's IP address from the network (using the BOOTP or RARP techniques described above)
- Load the server code (using the BOOTP protocol to identify the server's software file or from the internal path and filename specification). The actual loading is accomplished by a TFTP file transfer.

Optionally, you can also load the ViewStation's configuration from a remote configuration file. These three elements are independent of each other, though they are often used together.

These settings are all selected on the Setup Mode Startup Options menu, which is described in detail in Section 1.5.6.6.

HDS ViewStation: Remote Configuration

This chapter discusses remote configuration of the ViewStation. IP address assignment, loading the server code, and loading configuration files are discussed.

Section 1.0 - HDS ViewStation

1.7 Remote Configuration

One of the principal benefits of X terminals is the ability to configure and administer them remotely. There are a number of considerations.

- Determining the IP address for each ViewStation. There are several methods, which are discussed in Section 1.6.5. These are local setting, BOOTP, and RARP (or NIS).
- Loading the ViewStation's Server file. There are several methods, which are discussed in Section 1.6. These are local storage, BOOTP, Monitor Mode, and direct network loading.
- Loading the ViewStation's Configuration file. This can also involve questions of system and user modifiable settings. This file can be loaded locally or with BOOTP.
- Loading the ViewStation's Client Configuration file for its local clients.
- Passwords and system security.
- Remote monitoring and management (updates, changes, and troubleshooting)

Each of these matters will be discussed in turn.

1.7.1 Determining the IP Address

This is discussed in Section 1.6.5.

1.7.2 Loading the ViewStation's Server File

This is the subject of Section 1.6, the preceding chapter.

1.7.3 Loading the ViewStation's Configuration File

The configuration file is a resource file containing all the selections in Setup Mode and more. This file is located in */FXUtilities/hds-fx/config/config.fx* and is loaded automatically from that location with the HDSware automatic configuration feature of the ViewStation. It contains most important settings and paths, including HDSware locations specified with the HDSware or FXUtilities mount point.

There are configuration files designed for specific configuration choices, such as config.fx.dsk for ViewStations with a hard disk drive, config.4m for ViewStations booting with 4 MB of memory, and config.ext for ViewStations loading the X11 extensions.

The Setup Mode Startup Options menu contains a field to specify a configuration file for ViewStation loading; it can also be loaded with BOOTP.

1.7.3.1 FXUtilities and HDSware Abbreviation

The HDSware menu in Setup Mode (or **hh** and **hd** in Monitor Mode) specify the host and directory path for the HDSware software. This information is combined and used as the "HDSware" or "FXUtilities" abbreviation. Thus, FXUtilities with the internal HDSware path give a full specification for any path or file.

The Configuration file has entries for fontpaths in the form (for the config file):

HDSsetup.config.file: /FXUtilities/hds-fx/config/config.fx

Another useful abbreviation recognized by the ViewStation is the "BOOTPhost", which uses the root partition as its path.

1.7.4 Configuration File Syntax

There is a sample configuration file, *./hds-fx/config/config.syntax.fx*, in the HDSware distribution. This sample file gives the resource syntax and permitted values for all the configuration options.

Note: if there is no value entered for a resource, or the value entered is prohibited (out of range or wrong type), the default value will be used. Default values are shown in bold type.

Note: there must be no white space after the resource entry.

```
This file shows the syntax for the ViewStation's configura-
tion files.

Note that [ YES | NO ] denotes a choice which must be speci-
fied as either YES or NO if this resource name is given. Do
NOT write the enclosing square_brackets [ and ] or the
vertical_bar |. The entry DOMAIN_NAME must be either a
numeric IP_ADDRESS nnn.nnn.nnn.nnn or an alphanumeric NAME
which can be resolved by a domain name server.

Field Name Value and Default Assignment

HDSsetup.hostname: DOMAIN_NAME

HDSsetup.ipaddress: nnn.nnn.nnn.nnn
```

```
HDSsetup.ipaddress.resolve: [ BOOTP | RARP | SETUP ]

HDSsetup.ethertype: [ THIN | THICK ]

HDSsetup.network.netmask: nnn.nnn.nnn.nnn
HDSsetup.network.gateway: nnn.nnn.nnn.nnn
HDSsetup.network.broadcast: nnn.nnn.nnn.nnn
HDSsetup.network.timeout: [ 5 to 120 ] (default = 15 sec-
onds)
HDSsetup.password: <encrypted value when BOOTP password
given>

HDSsetup.pcmcia.internal.tr.ipaddress: nnn.nnn.nnn.nnn
HDSsetup.pcmcia.internal.tr.netmask: nnn.nnn.nnn.nnn
HDSsetup.pcmcia.internal.tr.mtu: [ 516 | 1500 | 2052 | 4472
| 8144 | 11407 | 17800 ] (default = 2052)
HDSsetup.pcmcia.external.tr.ipaddress: nnn.nnn.nnn.nnn
HDSsetup.pcmcia.external.tr.netmask: nnn.nnn.nnn.nnn
HDSsetup.pcmcia.external.tr.mtu: [ 516 | 1500 | 2052 | 4472
| 8144 | 11407 | 17800 ] (default = 2052)

HDSsetup.boot.download: [ NETWORK | ROM ]
HDSsetup.boot.bootp.protocol: [ TFTP | FTP | NFS ]
HDSsetup.boot.bootp.username:
HDSsetup.boot.bootp.password: <encrypted value when BOOTP
password given>
HDSsetup.boot.bootp.account:

HDSsetup.ServerCode.file:
HDSsetup.config.file: (default = /FXUtilities/hds-fx/config/
config.fx)
HDSsetup.syslog.config.file:

HDSsetup.rhosts.file:
```

HDSsetup.serial1.ipaddress: nnn.nnn.nnn.nnn

HDSsetup.serial1.add.network.route: [**NO** | YES]

HDSsetup.serial1.set.default.gateway: [NO | **YES**]

HDSsetup.remote.serial1.host: DOMAIN_NAME

HDSsetup.remote.serial1.netmask: nnn.nnn.nnn.nnn

HDSsetup.serial2.ipaddress: nnn.nnn.nnn.nnn

HDSsetup.serial2.add.network.route: [**NO** | YES]

HDSsetup.serial2.set.default.gateway: [NO | **YES**]

HDSsetup.remote.serial2.host: DOMAIN_NAME

HDSsetup.remote.serial2.netmask: nnn.nnn.nnn.nnn

HDSsetup.ppp.serial1.messagelevel: [**LOW** | MEDIUM | HIGH]

HDSsetup.ppp.serial1.headercomp: [**ENABLED** | DISABLED]

HDSsetup.ppp.serial1.protocomp: [**ENABLED** | DISABLED]

HDSsetup.ppp.serial1.linkmonitor: [**ENABLED** | DISABLED]

HDSsetup.ppp.serial1.linkmonitor.interval: [5 to 300] (default = 10)

HDSsetup.ppp.serial1.addressnegotiation: [**ENABLED** | DISABLED]

HDSsetup.ppp.serial1.mru: [128 to 1500] (default = 1500)

HDSsetup.ppp.serial1.allowpap: [**NO** | YES]

HDSsetup.ppp.serial1.local.name: DOMAIN_NAME

HDSsetup.ppp.serial1.local.password: <encrypted value when BOOTP password given>

HDSsetup.ppp.serial2.messagelevel: [**LOW** | MEDIUM | HIGH]

HDSsetup.ppp.serial2.headercomp: [**ENABLED** | DISABLED]

HDSsetup.ppp.serial2.protocomp: [**ENABLED** | DISABLED]

HDSsetup.ppp.serial2.linkmonitor: [**ENABLED** | DISABLED]

HDSsetup.ppp.serial2.linkmonitor.interval: [5 to 300] (default = 10)

HDSsetup.ppp.serial2.addressnegotiation: [**ENABLED** | DISABLED]

HDSsetup.ppp.serial2.mru: [128 to 1500] (default = 1500)

HDSsetup.ppp.serial2.allowpap: [**NO** | YES]

HDSsetup.ppp.serial2.local.name: DOMAIN_NAME

HDSsetup.ppp.serial2.local.password: <encrypted value when BOOTP password given>

HDSsetup.port.serial1.usage: [**UNSPECIFIED** | SLIP | CSLIP | PPP | CONSOLE]

HDSsetup.port.serial1.baud: [150 | 300 | 1200 | 2400 | 4800 | **9600** | 19200 | 38400 | 57600]

HDSsetup.port.serial1.parity: [**None** | Even | Odd]

HDSsetup.port.serial1.databits: [7 | **8**]

HDSsetup.port.serial1.stopbits: [**1** | 2]

HDSsetup.port.serial1.flow: [**None** | XON/XOFF | SendXOFF | RecvXOFF | CTS/RTS]

HDSsetup.port.serial2.usage: [**UNSPECIFIED** | SLIP | CSLIP | PPP | CONSOLE]

HDSsetup.port.serial2.baud: [150 | 300 | 1200 | 2400 | 4800 | **9600** | 19200 | 38400 | 57600]

HDSsetup.port.serial2.parity: [**None** | Even | Odd]

HDSsetup.port.serial2.databits: [7 | **8**]

HDSsetup.port.serial2.stopbits: [**1** | 2]

HDSsetup.port.serial2.flow: [**None** | XON/XOFF | SendXOFF | RecvXOFF | CTS/RTS]

HDSsetup.domain_server.primary: nnn.nnn.nnn.nnn

HDSsetup.domain_server.secondary: nnn.nnn.nnn.nnn

HDSsetup.domain_server.suffix:

HDSsetup.xdmcp.access: [**OFF** | Broadcast | Direct | Indirect | Hosts]

HDSsetup.xdmcp.action: [Stop | **Persist**]

```
HDSsetup.xdmcp.host: DOMAIN_NAME

HDSsetup.xdmcp.host.file:

HDSsetup.xdmcp.keepalive.minimum: [ 0 to 10000 ] (default =
1 minute)

HDSsetup.xdmcp.keepalive.timeout: [ 3 to 3600 ] (default =
20 seconds)

HDSsetup.xdmcp.chooser: [ YES | NO ]

HDSsetup.xdmcp.chooser.AllowDirect: [ NO | YES ]

HDSsetup.xdmcp.default.AccessControl: [ ENABLED | DISABLED ]

HDSsetup.xdmcp.server_reset: [ YES | NO ]


HDSsetup.password.is_global: [ NO | YES ]

HDSsetup.hide_icon: [ NO | YES ]


HDSsetup.telnet.primary.host: DOMAIN_NAME

HDSsetup.telnet.autostart: [ NO | YES ]


HDSsetup.Mwm.autostart: [ NO | YES ]

HDSsetup.Olwm.autostart: [ NO | YES ]

HDSsetup.console.autostart: [ NO | YES ]

HDSsetup.memory.autostart: [ NO | YES ]


HDSsetup.snmp.host: DOMAIN_NAME

HDSsetup.snmp.traps: [ NO | YES ]


HDSsetup.RGB.file:


HDSsetup.FXUtilities.address: DOMAIN_NAME

HDSsetup.FXUtilities.hosttype: [ UNIX | VMS ]

HDSsetup.FXUtilities.protocol: [ TFTP | FTP | NFS ]

HDSsetup.FXUtilities.username:

HDSsetup.FXUtilities.password: <encrypted value when BOOTP
password given>
```

```
HDSsetup.FXUtilities.account:

HDSsetup.FXUtilities.directory:


HDSsetup.host1.hostname:

HDSsetup.host1.address: DOMAIN_NAME

HDSsetup.host1.hosttype: [ UNIX | VMS ]

HDSsetup.host1.protocol: [ TFTP | FTP | NFS ]

HDSsetup.host1.username:

HDSsetup.host1.password: <encrypted value when BOOTP pass-
word given>

HDSsetup.host1.account:

HDSsetup.host1.local.directory:

HDSsetup.host1.remote.directory:


HDSsetup.host2.hostname:

HDSsetup.host2.address: DOMAIN_NAME

HDSsetup.host2.hosttype: [ UNIX | VMS ]

HDSsetup.host2.protocol: [ TFTP | FTP | NFS ]

HDSsetup.host2.username:

HDSsetup.host2.password: <encrypted value when BOOTP pass-
word given>

HDSsetup.host2.account:

HDSsetup.host2.local.directory:

HDSsetup.host2.remote.directory:


HDSsetup.host3.hostname:

HDSsetup.host3.address: DOMAIN_NAME

HDSsetup.host3.hosttype: [ UNIX | VMS ]

HDSsetup.host3.protocol: [ TFTP | FTP | NFS ]

HDSsetup.host3.username:

HDSsetup.host3.password: <encrypted value when BOOTP pass-
word given>

HDSsetup.host3.account:
```

```
HDSsetup.host3.local.directory:

HDSsetup.host3.remote.directory:


HDSsetup.host4.hostname:

HDSsetup.host4.address: DOMAIN_NAME

HDSsetup.host4.hosttype: [ UNIX | VMS ]

HDSsetup.host4.protocol: [ TFTP | FTP | NFS ]

HDSsetup.host4.username:

HDSsetup.host4.password: <encrypted value when BOOTP pass-
word given>

HDSsetup.host4.account:

HDSsetup.host4.local.directory:

HDSsetup.host4.remote.directory:


HDSsetup.host5.hostname:

HDSsetup.host5.address: DOMAIN_NAME

HDSsetup.host5.hosttype: [ UNIX | VMS ]

HDSsetup.host5.protocol: [ TFTP | FTP | NFS ]

HDSsetup.host5.username:

HDSsetup.host5.password: <encrypted value when BOOTP pass-
word given>

HDSsetup.host5.account:

HDSsetup.host5.local.directory:

HDSsetup.host5.remote.directory:


HDSsetup.client.config.path:

HDSsetup.startup.command.file:


HDSsetup.server_reset.command.file:


HDSsetup.KeycodeTranslation.path:
```

HDSsetup.screen.width: nnnn (pixels value)

HDSsetup.screen.height: nnnn (pixels value)

HDSsetup.screen.depth: n (bits value)

HDSsetup.keyboard.kbtype: [**Default** | Swiss-German | Swiss-French | Swedish | French | Japanese | French-Canadian | French-Canadian-compose]

HDSsetup.keyboard.click: [0 | **25** | 50 | 100] (default = 25 percent)

HDSsetup.bell.volume: [0 | 25 | **50** | 100] (default = 50 percent)

HDSsetup.bell.pitch: [100 to 10000] (default = 600 Hz)

HDSsetup.bell.duration: [10 to 1000] (default = 100 millisecs)

HDSsetup.screensaver.timeout: [0 to 32767] (default = 300 secs)

HDSsetup.screensaver.interval: [0 to 300] (default = 30 secs)

HDSsetup.screensaver.method: [**Energy Star** | Blanking | XLogo]

HDSsetup.backingstore.enable: [**YES** | NO]

HDSsetup.backingstore.saveunder: [**YES** | NO]

HDSsetup.backingstore.default: [**NEVER** | MAPPED]

HDSsetup.default.file.protocol: [**TFTP** | FTP | NFS]

HDSsetup.PixelZeroWhite: [**NO** | YES]

HDSsetup.memwarn: nnn (Kbytes value; default = 300)

HDSsetup.server_reset: [**NO** | YES]

HDSsetup.keyboard_cursor_mode: [**DISABLED** | ENABLED]

```
HDSsetup.SaveToNVM: [ NO | YES ]

HDSsetup.net_type: [ Auto Sense | Ethernet | IEEE 802.3 ]

HDSsetup.NFS.block.size: [ 128 to 8192 ] (default = 8192
bytes)

HDSsetup.syslog.buffer.size: [ 8 to 64 ] (default = 12 kb)

HDSsetup.hostresolver.host: nnn.nnn.nnn.nnn

HDSsetup.hostresolver.file:

HDSsetup.font.cache.size: nnn (Kbytes value; default = 1024)

HDSsetup.timeserver.enable: [ YES | NO ]

HDSsetup.timeserver.host: DOMAIN_NAME

HDSsetup.default.AccessControl: [ DISABLED | ENABLED ]

HDSsetup.audio.in.device: [ Unspecified | HDS ViewCam | HDS
Headset | CallPort Spkr/Mic | HDS Microphone | HDS Tuner |
VCR | Line Level Audio ]

HDSsetup.audio.out.device: [ HDS Internal Speaker | External
Speakers | HDS Headset | CallPort Spkr/Mic ]

HDSsetup.audio.in.volume: [ 1 to 100 ]

HDSsetup.audio.out.volume: [ 0 to 100 ]

HDSsetup.internal.flow.timeout: [ 0 to 32767 ] (default = 60
secs)
```

This file changes with HDSware versions. Check current HDSware for new ViewStation configuration resources.

1.7.4.1 Generating an Individual Configuration File

Your configuration file should start from *./hds-fx/config/config.syntax.fx* because it has some useful default settings, such as the FXUtilities paths. You can extract parts of this syntax file and select the appropriate values to construct your config.fx file. Your file should contain only settings different from the defaults.

There is an easy way to generate a configuration file. The Console Window has a command:

nvm dump

that reports your current Setup Mode settings in resource form. These entries can be made part of your configuration file since the syntax is correct. All you need to do is set up one ViewStation with your desired settings using the Setup Mode menus, then do nvm dump in the Console Window, and then cut-and-paste into your config.fx file.

This is a sample nvm dump. Note that the header information and the individual IP address should not become part of the config.fx file.

```
HDS> nvm dump

# NVM header:

# size = 8

# flags = 0

# checksum = af44

# version = 2

# revision = 4

# length = 790

HDSsetup.ipaddress: 128.91.6.10

HDSsetup.ipaddress.resolve: Setup

HDSsetup.network.netmask: 255.255.0.0

HDSsetup.network.gateway: 128.91.2.5

HDSsetup.network.broadcast: 128.92.255.255

HDSsetup.ServerCode.file: /FXUtilities/hds-fx/image/V30-
D.hds

HDSsetup.config.file: None

HDSsetup.syslog.config.file: /FXUtilities/hds-fx/config/sys-
log.conf

HDSsetup.domain_server.primary: 128.91.3.1

HDSsetup.domain_server.suffix: hds.com

HDSsetup.telnet.primary.host: hds486
```

```
HDSsetup.telnet.autostart: YES

HDSsetup.Olwm.autostart: YES

HDSsetup.default.font.path: /FXUtilities/hds-fx/fonts/75dpi/
,/FXUtilities/hds-fx/fonts/100dpi/,/FXUtilities/hds-fx/
fonts/misc/,/FXUtilities/hds-fx/fonts/Xol,/FXUtilities/hds-
fx/fonts/wp5.1/

HDSsetup.FXUtilities.address: 128.92.3.12

HDSsetup.FXUtilities.directory: /usr2

HDSsetup.host1.hostname: hdssun

HDSsetup.host1.address: 128.92.3.12

HDSsetup.host2.hostname: hdssun1

HDSsetup.host2.address: 128.91.3.1

HDSsetup.host3.hostname: hdssun

HDSsetup.host3.address: 128.92.3.12

HDSsetup.host4.hostname: hds486

HDSsetup.host4.address: 128.91.2.5

HDSsetup.client.config.path: /FXUtilities/hds-fx/config/cli-
ents.config

HDSsetup.KeycodeTranslation.path: /FXUtilities/hds-fx/config

HDSsetup.screen.width: 1280

HDSsetup.screen.height: 1024

HDSsetup.screen.depth: 8

HDSsetup.backingstore.enable: NO

HDSsetup.backingstore.saveunder: NO

HDSsetup.keyboard_cursor_mode: Enabled

# NVM dump complete
```

1.7.5 Using Include Statements in a Configuration File

The ViewStation's configuration file can have some "include" statements attached to it. This is a convenient way to control settings and variations in a complex environment. It allows you to have one configuration file with common settings and individual configuration files with unique entries.

You can use this feature by adding an include statement at the end of a configuration file. Typically, the basic configuration file would contain settings that are common to everyone in the environment and would be attached as an include file after the unique settings configuration file.

The include statement gives a filename for a second configuration file that contained settings common to a subnet or to a group of users. These settings would be added to the primary configuration file containing the unique information. This allows you to customize some things, such as window managers, gateway computer addresses, fontpaths, and so on, that would apply to some users only. Note that the ViewStation looks for the include file in the same directory as the original configuration file (you could also specify a full file path if you wanted).

Suppose you wanted separate configurations for MWM and OLWM users. You would have three configuration files, one for MWM, one for OLWM, and a common configuration file for all users: Note that the name of the file in the include statement must have double quotes ("filename").

```
# config.mwm

# This is the configuration file for MWM users

HDSsetup.Mwm.autostart: YES

#include "config.fx.general"


# config.olwm

# This is the configuration file for OLWM users

HDSsetup.Olwm.autostart: YES

HDSsetup.default.font.path: /FXUtilities/hds-fx/Xol

#include "config.fx.general"


# config.fx.general

# This is the configuration file with common settings

HDSsetup.FXUtilities.address: 128.91.3.1

HDSsetup.FXUtilities.directory: /usr/home/HDSware

HDSsetup.domain_server.primary: 128.91.3.1

HDSsetup.domain_server.suffix: hds.com

HDSsetup.client.config.path: /FXUtilities/hds-fx/config/cli-
ents.config
```

In this case, users select their window manager and its associated files by specifying one configuration file or the other. An MWM user would enter his configuration file as */FXUtilities/hds-fx/config/config.mwm*.

1.7.6 ViewStation Hard Disk Configuration File

ViewStations that have an internal hard disk use a different configuration file, called *./hds-fx/config/config.fx.dsk*. This file is different in several important ways. It is shown below.

If you have a disk drive installed on your ViewStation, the factory installation provides different configuration files so the paths all point to the local disk drive rather than to a directory on a remote host. These configuration files are called "config.fx.dsk", which distinguishes them from the normal "config.fx" files.

1.7.6.1 Configuration File config.fx.dsk

The "config.fx.dsk" file is used for ViewStation configuration and specifies the paths to find HDSware. The factory installation sets this config.fx.dsk file to be used automatically. If you use your own configuration files, you should use this configuration file (or the relevant parts of it) for booting, either from Monitor Mode, or Bootp.

```
/hds-fx/config/config.fx.dsk

#

# This is the configuration file used by default when the

# FXUtilities host address and directory path have

# been set to the local ViewStation disk.

# Setup entries for HDSware are then

# HDSware host: <local ViewStation IP> and

# HDSware Directory path: </dev/flash>.

#

# Note the use of the Internet loopback address

# for this local ViewStation.

# Note also the local directory specification: /dev/flash.

#

HDSsetup.FXUtilities.address: 127.0.0.1

HDSsetup.FXUtilities.directory: /dev/flash

HDSsetup.default.font.path:/FXUtilities/hds-fx/fonts/75dpi/
,/FXUtilities/hds-fx/fonts/100dpi/,/FXUtilities/hds-fx/
fonts/misc/,/FXUtilities/hds-fx/fonts/Xol/

HDSsetup.ServerCode.file: /FXUtilities/hds-fx/image_u/V303-
D.uhds

HDSsetup.client.config.path: /FXUtilities/hds-fx/config/cli-
ents.config.dsk
```

1.7.6.2 Configuration File clients.config.dsk

The "clients.config.dsk" file is used to set the path and configuration for the ViewStation's local clients. This file also specifies these paths as the local hard disk drive. This file is specified in the "config.fx.dsk" and is used automatically when you start the ViewStation with the hard disk drive. If you use your own configuration files, you should use this file (or the relevant parts of it). The file shown here is only a section of the file as it appears on the disk, but contains sufficient information for you to see the changes necessary.

```
/hds-fx/config/clients.config.dsk
#
# This is the configuration file for ViewStation local clients.
# The config.fx.dsk file should point to this file in "client.config.path:".
# This contains common resources for most clients.
#
*defaultEnv: XFILESEARCHPATH=/FXUtilities/hds-fx/lib/%N%S:/
HDS$$ROM/%N%S:/FXUtilities/hds-fx/lib/app-defaults/%N%S\n\
     XBMLANGPATH=/HDS$$ROM/%B
#
# Mwm - Motif Window Manager.
hds.client.mwm.path: /FXUtilities/hds-fx/bin/mwm
hds.client.mwm.defaultEnv: XFILESEARCHPATH=/HDS$$ROM/%N%S\n\
    XBMLANGPATH=/HDS$$ROM/%B\n\
    MWMSHELL=/bin/sh
hds.client.mwm.transparentMode: yes
hds.client.mwm.minAvailMem: 570k
hds.client.mwm.critical: yes
#
# HDSCalc Calculator
hds.client.hdscalc.menuName: HDS calc
hds.client.hdscalc.path: /FXUtilities/hds-fx/bin/hdscalc
#hds.client.hdscalc.defaultEnv: XFILESEARCHPATH=/FXUtilities/hds-fx/lib/
app-defaults/%N
hds.client.hdscalc.minAvailMem: 460k
#
# Xdmcp chooser
hds.client.chooser.path:/FXUtilities/hds-fx/bin/chooser
#
```

```
# other entries in this file have been deleted from this printed copy.
# refer to the file itself for a complete listing.
#
```

1.7.7 Configuration File for 4 MB ViewStations

There is a special configuration file available for ViewStations running with 4 MB of memory. This configuration file "config.fx.4m" contains an entry for the server code version Vnn.hds rather than the standard Vnn-D.hds server version with downloadable clients. This server version cannot download local clients.

1.7.8 Remote Configuration Security Considerations

There are a number of concerns with system security. These include setting and changing a password, sending the password as part of a configuration file, etc. Other concerns of password management, either by the user or the system administrator, are also relevant to remote management of ViewStations. These things are discussed in detail in Section 1.11.

1.7.9 Remote Management of ViewStations

When the ViewStations are set up, whether locally or remotely, there is sometimes a need to manage them. This might involve changing settings, updating versions, or problem diagnosis.

1.7.9.1 Changing ViewStation Settings

You can change settings on the ViewStation either by using its Setup Mode from a remote machine, described in Section 1.5.1.1, or by modifying the config.fx file and loading it on reboot. The configuration file can specify that its contents be saved in NVM on the ViewStation.

1.7.9.2 Changing the ViewStation Password

The system administrator can change the password on an individual ViewStation by using a remote login to the Console Window and then using the password command. Note that this command is password protected.

1.7.9.3 Changing Server Code Versions

You can update the server code (and / or the bootprom if it is loaded on the Flash-PROM card) by using a remote login to the Console Window and then using the "flash" command. This is discussed in detail in Section 1.2.7.

1.7.9.4 Problem Diagnosis from a Remote Machine

Problem diagnosis is made easy via the Console Window, which provides an interface to the ViewStation's internal architecture and an error log for the ViewStation. The error log can be automatically redirected to your host's syslog file. This is discussed in Section 1.2.3.1.

HDS ViewStation: Keyboards and their Operations

This chapter discusses ViewStation keyboards and their operation. Some customization is also discussed.

Section 1.0 - HDS ViewStation

1.8 Keyboards and their Operations

Keyboards are the primary user interface to the desktop device. Users have many requirements and preferences, so keyboards and their operation and configuration are important considerations.

1.8.1 HDS ViewStation Keyboards

The HDS ViewStation supports a number of keyboards, shown below.

- IBM 101+ style keyboard (AT type) - this is the standard keyboard
- DEC VT420 type
- IBM 102 key multinational keyboard
- Lexmark 122 key IBM 3270 keyboard
- Sun Type 4 style or Type 5 style (using either Serial Port 1 with special modifications for older ViewStations) or with a Sun keyboard connector on FXD and Ultra ViewStations
- Multinational keyboards of various configurations are also supported.

All keyboards use a five-pin DIN male connector with an IBM interface. Complete key remappings and configurations are selectable from software.

Other keyboards that support the IBM AT keyboard interface will probably work with the ViewStation, but are not supported without HDS's testing and approval. They may also require special user-defined keymap tables.

1.8.1.1 Keyboard Connections

The standard keyboard connector is a five-pin DIN male connector with an IBM AT interface, see Section 1.1.10.2.

Sun keyboards use a separate connector on the back panel.

1.8.2 Keystroke Functions on the ViewStation

As a convenience to users, the HDS ViewStation uses a number of special keystrokes for various functions. These are listed below, along with their variations on optional keyboards. These key definitions apply only in the clients indicated and only under the correct circumstances.

1.8.2.1 Keyboard Cursor Mode

The ViewStation can use its keyboard cursor keys to simulate the mouse movement. This is called "keyboard cursor mode" and is used in environments where a mouse is not practical. Keyboard cursor mode is activated by an entry in the configuration file, config.fx:

```
HDSsetup.keyboard_cursor_mode: [ ENABLED | DISABLED ]
```

When keyboard cursor mode is enabled, the cursor keypad functions as a mouse. The arrow keys move the cursor (note the increasing acceleration of the mouse cursor when the key is held down), and the lower three keys of the editing keypad function as mouse buttons.

When keyboard cursor mode has been enabled in the configuration file, you can toggle the action by using the Alt-Ctrl_R keys. These keys toggle keyboard cursor mode On and Off; the Scroll Lock LED lights when it is enabled.

1.8.2.2 Setup Mode Keystrokes

Setup Mode Editing Keys

The editing keypads on the keyboards are active in Setup Mode on most keyboards.

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Cursor keys	yes	yes	yes	yes
Insert function	Insert	InsertHere	Ins	Insert
Delete function	Delete	none	Del	Del
Home / PageUp	Home / PgUp	Prev	none	Home / PageUp
End / PageDown	End / PageDn	Next	none	End / PageDown

Setup Mode Hidden Icon Hotkey

If you have the Setup Mode resource set to "Hide Icon", the Setup Mode icon doesn't appear on the screen unless you hit the Setup Mode hotkey. This is a definable key sequence which you can set with the hotkey resource in the Setup client's defaultArgs table in the *./hds-fx/config/clients.config* file. In this example the new hotkey would be Shift-Alt-F1:

hds.client.setup.defaultArgs: -xrm '*hotkey: Shift Alt<Key>F1'

The default assignments for the hotkeys are as follows:

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Setup Mode Hotkey	Alt-SysRq	Alt-Help	Alt-PrSc	Alt-PrintScreen

1.8.2.3 Monitor Mode Keystrokes

When you first turn on the ViewStation, the white screen you see is called the Boot Monitor. If you want to go directly to Monitor Mode to make your settings, you can press the "m" key while the "Initializing" message is displayed (3 seconds); otherwise you have to wait until the boot process finishes in order to get into Monitor Mode.

There are two other keystrokes that can be useful, a "Pause" function to pause (and resume) the boot process, and a "Debug message" function to display more detailed messages (two levels of debug messages, activated with one or two keypresses).

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Pause (Resume)	Pause	F1 (Hold)	Pause	Pause
Debug messages (1x or 2x)	PrntScrn	Help	PrSc	PrintScreen

1.8.2.4 HDSterm Environment Variable Keystrokes

You have to set your environment variable when you login into a host computer so the computer can send the application client windows to the proper display. The ViewStation's Telnet clients (HDSterm and tty Telnet) have special key definitions to make this setting easy for you. These definitions automatically insert the IP address of your ViewStation.

There are settings for several different environments.

Function	IBM kybd	DEC kybd	Sun4 kybd	Sun5 kybd
Env. variable (C shell)	Alt-F12	Alt-F12	Alt-Again	Alt-Again
produces: "setenv DISPLAY <*IPaddress*>:0"				
Env. variable (Bourne shell)	Alt-F11	Alt-F11	Alt-Stop	Alt-Stop
produces: "DISPLAY=<*IPaddress*>:0;export DISPLAY"				
Env. variable (VMS hosts)	Alt-F10	Alt-F10	Alt-F10	Alt-F10
produces: "set display / create / server=0 / trans=tcpip / node=<*IPaddress*>"				

1.8.3 Keyboard Operations

The keyboard is sensed automatically when it is connected. If it is unplugged and then reattached, the ViewStation automatically re-initializes the keyboard. You can also use the "keyboard" command from the Console window to reset the keyboard.

As part of its initialization, the ViewStation queries the keyboard for its type and the keyboard replies with an identifier code. The ViewStation matches this code to a table of keycodes appropriate to that keyboard.

This keycode (sent from the keyboard to the ViewStation) is matched to a key symbol (a character or function) in a table called the xmodmap table. This table is visible to the user by the xmodmap command:

xmodmap -pk (this prints the keymap table)

xmodmap -pm (this prints the modifier table)

xmodmap -pp (this prints the pointer table)

Refer to Section 2.6.13 for more details on xmodmap.

1.8.3.1 Keycode Modification Tables

This xmodmap table of keycodes can be modified by the user if needed, but this is usually not recommended. Modifications to the keymap can be made in the keycode translation file in *./hds-fx/config/<keycode file>*.

This keyboard translation mapping is for sites using keycodes in their xmodmap files (this is not recommended - keysym to keysym remappings are preferred and do not require keycode remapping). In order to modify the keycodes for any keyboard, set this resource in the ViewStation's configuration file:

HDSsetup.KeycodeTranslation.path: /FXUtilities/hds-fx/config

This specifies the directory for the KeycodeTranslation file, whose name is the hexadecimal keyboard identifier.

For example:

IBM-101+ keyboard identifier is AB83

Sun4-style keyboard identifier is AB04

The keyboard identifier is a property of the root window, so if you don't know the identifier, you can look at the properties of the root window for its value (use the xprop command and click on the root window).

1.8.3.2 Key Mappings

Xmodmap is an X client on your host computer (it is also a ViewStation local client) that reports your keyboard mappings. Refer to the xmodmap manual pages for more examples and command syntax.

This is a sample of an xmodmap -pk print. It is a section of the file indicating the kind of thing you'll see in the file.

```
203 0x0033 (3) 0x0023 (numbersign)
204 0x0065 (e) 0x0045 (E)
205 0x0064 (d) 0x0044 (D)
206 0x0063 (c) 0x0043 (C)
207
208 0x0034 (4) 0x0024 (dollar)
209 0x0072 (r) 0x0052 (R)
210 0x0066 (f) 0x0046 (F)
211 0x0076 (v) 0x0056 (V)
```

```
212 0x0020 (space)
213
214 0x0035 (5) 0x0025 (percent)
215 0x0074 (t) 0x0054 (T)
216 0x0067 (g) 0x0047 (G)
217 0x0062 (b) 0x0042 (B)
218
219 0x0036 (6) 0x005e (asciicircum)
220 0x0079 (y) 0x0059 (Y)
221 0x0068 (h) 0x0048 (H)
222 0x006e (n) 0x004e (N)
```

From this excerpt you can see that keycode 222 refers to keysyms "n" unshifted and "N" shifted. The columns indicate the keysym with the shift key. Other modifiers, if defined, are used the same way.

1.8.3.3 Modifier Keymappings

This is the modifier mapping for the ViewStation.

```
xmodmap: up to 2 keys per modifier, (keycodes in parentheses):
shift Shift_R (0xab), Shift_L (0xae)
lock Caps_Lock (0xb0)
control Control_L (0xaf), Control_R (0xb3)
mod1 Alt_L (0xac), Alt_R (0xb2)
mod2
mod3
mod4
mod5 Num_Lock (0x84)
```

This modifier map reports the current modifier settings.

1.8.3.4 ViewStation Pointer Mappings

This is the ViewStation's pointer mapping.

```
There are 3 pointer buttons defined.
 Physical   Button
 Button     Code
 1          1
 2          2
 3          3
```

1.8.4 Xev Key Monitoring

You can use the Xev (X Event) utility program (this is a standard X client) to see what keycodes and keysyms are actually being transmitted. When you start Xev, place the cursor in the event box, and then press a key, or move the mouse, etc. You will see the event logged on your screen with a lot of information about the event. This example shows the keycode, the modifiers' state, and the keysym. This helps you analyze what is being sent and how the assignments are made.

This is a sample Xev event that reports the sequence <Press Shift_L>, <Press a>, <Release a>, <Release Shift_L>, which produces "A":

```
KeyPress event, serial 15, synthetic NO, window 0x3800001,
 root 0x29, subw 0x3800002, time 95819540, (47,49),
root:(112,135), state 0x0, keycode 174 (keysym 0xffe1,
Shift_L), same_screen YES, XLookupString gives 0 characters:
""
KeyPress event, serial 15, synthetic NO, window 0x3800001,
 root 0x29, subw 0x3800002, time 95821980, (47,49),
root:(112,135), state 0x1, keycode 194 (keysym 0x41, A),
same_screen YES, XLookupString gives 1 characters: "A"
KeyRelease event, serial 15, synthetic NO, window 0x3800001,
 root 0x29, subw 0x3800002, time 95822180, (47,49),
root:(112,135), state 0x1, keycode 194 (keysym 0x41, A),
same_screen YES, XLookupString gives 1 characters: "A"
KeyRelease event, serial 15, synthetic NO, window 0x3800001,
 root 0x29, subw 0x3800002, time 95823630, (47,49),
root:(112,135), state 0x1, keycode 174 (keysym 0xffe1,
Shift_L), same_screen YES, XLookupString gives 0 characters:
""
```

Note the keycode and keysym information. The "state" refers to the state of the modifier keys (Shift = 1, Ctrl = 2, Alt = 4, and so on). The XLookupString interprets the keysym for you.

1.8.5 Key Handling in a Client/Server Environment

One objective of Client / Server architecture is to handle configurations such as keyboard layouts or keypress assignments separately, either on the server or on the client, so communication between the two is independent of the actual configuration of either. What this means for keyboards and keymaps is that the X terminal server handles everything about the keyboard configuration (hardware connections, keyboard layout, etc.) and produces a resulting key function called a "keysym". This keysym is a neutral function, such as "A" or "Backspace" or "F4". The client application receives this keysym and interprets it according to its own configuration (F4 may mean "Save and Exit" or "Next Page" or something else).

In this way a keysym, such as "F24", may be generated by any keyboard configuration (such as different keyboard layouts, multinational languages, use of Shift

or other modifier keys, etc.) on the X terminal. This keyboard configuration is handled by a server file called xmodmap.

At the same time, the "F24" keysym may be interpreted by clients in any way they wish (such as "Home the cursor" in one application, "Insert marker text" in another, and "<string = your email address>" in another). This keysym interpretation is handled by a client resource called an event translation. This design lets any user produce an "F24" keysym, with whatever keyboard configuration, and have F24 produce the expected behavior on any application.

For the user wanting to modify the keymapping, there are two avenues of approach:

- Change xmodmap (the keyboard configuration file to produce the keysym on the X terminal), or
- Change the event translation file (to interpret the keysym in the client application)

There are some considerations that will influence your choice.

Changing xmodmap (the Keyboard Mapping on the ViewStation)

You can change the ViewStation's keymapping (both keyboard and mouse), but these changes will apply to everything you do. This may be desired in the case of switching mouse buttons for a left-handed user, or using a multinational keyboard. However, for most cases, this change is too broad and may introduce unexpected behavior and problems.

Changing the event translation (the Keysym Interpretation on the Client)

In most cases, this is the desired action since it applies to a specific application and key behavior. This change requires modifying the event translation resource for the client in your .Xdefaults file.

Both of these methods are described below, with examples.

1.8.5.1 Keymap Changes with Xmodmap

The xmodmap utility allows you to specify definitions and functions for each key on the keyboard. These are global changes.

Mouse Button Changes

For example, you can change the function of the mouse keys to make it easier for left-handed users to operate the mouse. Normally the most frequently used mouse key is the left button, which corresponds to the index finger of a right-handed user. To reverse the button order (to use the index finger of a left-handed person) use this entry in the xmodmap file:

```
xmodmap -e "pointer = 3 2 1"
```

Modifier and Lock Key Changes

Another commonly requested change is to swap the CapsLock key with the Left Ctrl key (note that some keyboards have a DIP switch to make this change). This example is taken from the manual pages for xmodmap. Note that the Lock and

Control functions are removed, then keysyms assigned, then the Lock and Control functions are added.

```
! Xmodmap data file which swaps the CapsLock and Left Ctrl keys.
! Usage: xmodmap    .xmodmap-swap-keys
!
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
```

Keysym Changes

Other changes often used are assigning the Backspace function to the Delete key, which is done as follows:

```
xmodmap -e "keysym BackSpace = Delete"
```

Note that you could also make this change in your xterm resource file, in the Event Translation table, or you could make it with a Unix stty command in your .login file.

Note that this entry changes one keysym to another. This is the recommended procedure. Users sometimes modify the modmap table that links a keycode to a keysym. This procedure is NOT recommended because it links the new keysym to a particular keycode (hardware type). If you use that keymap with another keyboard, it may not work. The keysym / keysym translation will always work.

These commands can be used on an individual xmodmap command line or grouped in a file which is then read by xmodmap. Xmodmap is typically read from an .xsession file or some other startup file. Note that the xmodmap client is available as a ViewStation local client. You could use the local xmodmap in a server-reset file (described in Section 1.2.8.1) to make your keyboard modifications even before you log in. This is an advantage to users with special keyboard requirements because even the username and password will use the modified keymapping (this is not true of host-based xmodmap).

Be careful with your changes so you are certain the specified key function behaves the way you want in all environments. These keyboard settings need to be considered carefully and depend on the client and client window and their definitions. Another consideration is the timing of the xmodmap execution. Since it is a global change, you probably want it to cover all clients at all times. This makes an xdm .xsession file or a .login file or a ViewStation server-reset file a good place to run xmodmap.

The xmodmap utility is very powerful. You can construct an xmodmap file for each user, or for the whole system as needed. Be careful with the syntax and the

labels associated with each key and function. There are commands in xmodmap to print the current keyboard mapping and also help commands. Refer to other X manuals for more details and examples.

Note also that these keyboard settings stay in place until the ViewStation is reset. This can be a problem if the ViewStation is used by different users with different customized xmodmap files. If user A uses xmodmap to change the Caps Lock and the CTRL keys and user B uses it to change only the Delete key, user B will find user A's settings still in place when he starts working on the keyboard. By selecting the "Reset server when last client exits" field in the User Preferences menu of Setup Mode, the xmodmap settings of one user are cleared when his session ends.

Multinational Keyboards and Mappings

Multinational keyboards present some additional functions and considerations. The ViewStation's Setup Mode, User Preferences menu, offers a number of multinational keyboard types that the ViewStation supports. These keyboard selections select particular internal keymaps for the ViewStation.

Since these keymaps need to be in place very early in the ViewStation's startup cycle (such as entering the user's name in an Xdm login banner), the keyboard type should be set in the configuration file. Additionally, since xmodmap is a ViewStation local client, it can be run from a server-reset file to provide additional keyboard customization.

Some multinational keyboards use four key legends per keytop. The upper and lower are shifted / unshifted values; the right and left pairs are selected with the "mode-switch" key (typically Alt_R). Note that this mode-switch function must be defined and then the key defined as a modifier key. The mode-switch key is assigned to keycode 151 (Alt_R). You must then make this a modifier function:

```
mod1 Alt_R(0xb2) mode-switch(0xnn)
```

This is all done automatically for the ViewStation, but you can make additional customizations in this way.

1.8.5.2 Keymap Changes with Event Translations

The Event Translation changes are localized, rather than global, changes. They apply to one client or one window only. For instance, an email window might have the F1 key function as a "Send" command, while your word processor might have F1 function as "Save". These definitions are all defined as resources of the client itself. They are entered in the client's resource file *./app-defaults*, or in your resource file *.Xresources* or *.Xdefaults*. It doesn't matter where the resource is defined (except for its global or local applicability), but you must be sure the resource is always read when the client starts.

There is a special kind of resource is called an Event Translation. This is quite a simple concept, despite its name. It simply means that one event (such as a key press or a button click) is translated into another. The most obvious example is that of a definition programmed onto a function key. In the X environment this process is handled as an "event translation", which means that pressing a function

key such as F10 (an event), generates a specific, defined response, which would be the "translation". These event translations are defined in a table, which is used as a resource for each individual client.

A translation table is handled as if it were a single resource; the "override" indicates that only the following definitions override the default definitions. A sample translation table might be:

```
XTerm*VT100.Translations:  #override\
<Key>F1:string ("lpr -Pps") string (0x0d)\n\
<Key>F2:string ("xwd | xpr -device ps | lpr") string
(0x0d)\n\
<Key>F3:string ("ls -C") string (0x0d)\n\
<Key>F4:string ("Mary Smith, Comptroller, XYZ Corporation")
```

Note these syntax requirements:

- The resource needs the "#override\" entry to show they are new translations for these keys.
- Strings are enclosed in parentheses; strings with spaces or special characters also need double quotation marks as delimiters.
- Commands that require a carriage return for execution have the hex value for CR entered as a string at the end of the entry.
- The table must be a single entry, so the `\n\` must appear at the end of each line except the last.
- This table is a single resource in your .Xresources file.

This example shows a number of applications for event translations. Note that the resource specification is for "Xterm*VT100" so it applies to all xterm clients that are started. It would not apply to other applications.

It shows some useful definitions, such as F1 doing a printer command, and F3 doing a list command. F4 prints a simple string for a frequently used phrase. F2 shows a more complicated command for a screen dump to a postscript printer.

Using resources and event translations is discussed in a number of X reference books. Refer also to the man pages for xrdb (and your clients), and also to Section 3.4 of this manual.

1.8.5.3 HDSterm Event Translations

HDSterm keymappings are described in Sections 2.2.9ff of the *HDS ViewStation System Administrator's Guide*. Refer to those sections for details and examples.

There are two special things to consider in modifying HDSterm keymappings:

- HDSterm uses sets of keys with separate translations for each set
- HDSterm has some keys defined as Motif special keys

The default mapping is a text file which you can see (or cut and paste) from the ViewStation's Console window with these commands:

```
cd HDS$$ROM
```

```
more HDSTerm
```

This is the HDSterm *app-defaults* file. It has the HDSTerm resources (note the uppercase T in the resource name) and then the keymap definitions. You can put these modified resources in your *.Xdefaults* file in your home directory, or any other location you read resources.

1.8.5.4 HDSterm Keymap Groups

Unlike many X Windows applications, the keyboard translations for HDSterm are not applied as a single default resource, which you modify by applying your own "HDSTerm*translations" resource. Instead, the translations are dynamically applied by HDSterm at run time, using several sets of translations. These sets are stored as application sub-resources, using the instance name:

*term.vt220-auxTrans-<set name>

The following sets of translations are used (see full listings in Section 2.2.9.1 or in the file itself):

Key-set Name	Translation Application
appKeypad	Map numeric keypad keys to DEC VT220/320 "application mode" strings
numKeypad	Map numeric keypad keys to DEC VT220/320 "numeric mode" strings
appCKeys	Map cursor keys to DEC VT220/320 "application mode" strings
normCKeys	Map cursor keys to DEC VT220/320 "normal mode" strings
fkeys	Map function keys to the strings generated by XTerm
ibmFkeys	Map function keys F1 to F4 to the DEC VT220/320 PF1 to PF4 strings
miscFkeys	Map "editing" keys to their closest equivalent to DEC VT220/320 editing keys, **based on their key legends**
ibmMiscFkeys	Map "editing" keys on IBM keyboard **based on their positional equivalent** on a DEC VT220/320 keyboard
telnetFkeys	Map Alt-F10 to Alt-F12 to generate complete shell commands to set the DISPLAY environment variable to the ViewStation's IP address:0

The way to understand these translations is to follow the use of the action procedure "key-input". If key-input has a parameter, it is taken to be an ASCII string which should be transmitted when the specified input event occurs. If key-input is used without a parameter, the specified event sends the normal ASCII binding returned by Xlib's XLookupString() function.

1.8.5.5 Customizing HDSterm Keymaps

To customize the keyboard translations, you must first find the keyset containing the keys you wish to change. Keys that belong to one of the special sets can be

redefined as part of the set. Keys that do not belong to a special set can be redefined with normal key override translations, the "*screen.translations" resource. Full key listings are shown in Section 2.2.9.1 of the *HDS ViewStation System Administrator's Guide.*

Ordinary Keys - For ordinary keys, you just redefine them. For instance, the Backspace key on an IBM keyboard, designated as osfBackSpace, often needs to be remapped to the Delete key for DEC applications. Since this key doesn't appear in any keyset group, you can use a normal screen translation, such as the following (but note the use of the osfBackSpace name for the keysym):

```
HDSTerm*screen.translations:  #override \
<Key>osfBackSpace:  key-input("\177")
```

Keyset Keys - If the key you want to redefine is in one of these keyset groups, you must use the entire keyset in your translation. This is necessary because the new keyset resource **completely replaces** the built-in default for that resource. Therefore, you may not use an "#override" directive to list only those keys you want changed. You must include all keys which have translations defined in the keyset. For ordinary keys, you may use the "#override" directive on the *screen.translations resource, as these are applied at widget initialization, as in the example above.

1.8.5.6 Customizing Example

If you want Shift-F9 to generate "cd $HOME" and F10 to function as a normal function key instead of the osfMenuBar, you would have to replace the entire "*term.vt220-auxTrans-fkeys" resource.

In the example below, this definition is added for the Shift-F9 key. Note that the unshifted F9 must now be explicitly prefaced by ~Shift (this means "not Shift") and the beginning colon removed (the colon used with other keys means "ignore modifier keys"). The F10 key first must have the accelerator function disabled by entering a null value, and is then remapped in the normal way. Note the use of the osf key binding names instead of the normal keysyms.

```
! this sets the osfMenuBar key (F10) to a null value
HDSTerm*menuBar.menuAccelerator: ""
! this redefines the 'fkey' group for HDSTerm keys
HDSTerm*term.vt220-auxTrans-fkeys: \
 ~Shift <Key>osfHelp: key-input("Ctrl-[[11~")\n \
 Shift <Key>osfHelp: key-input("Ctrl-[[25~")\n \
 :<Key>F2: key-input("Ctrl-[[12~")\n \
 :<Key>F3: key-input("Ctrl-[[13~")\n \
 :<Key>osfMenu: key-input("Ctrl-[[14~")\n \
 :<Key>F5: key-input("Ctrl-[Ctrl-[[15~")\n \
 :<Key>F6: key-input("Ctrl-[[17~")\n \
 :<Key>F7: key-input("Ctrl-[[18~")\n \
```

```
:<Key>F8: key-input("Ctrl-[[19~")\n \
~Shift <Key>F9: key-input("Ctrl-[[20~")\n \
Shift <Key>F9: key-input("cd $HOME")\n \
~Shift <Key>osfMenuBar: key-input("Ctrl-[[21~")\n \
Shift <Key>osfMenuBar: key-input("Ctrl-[[22~")\n \
:<Key>F11: key-input("Ctrl-[[23~")\n \
:<Key>F12: key-input("Ctrl-[[24~")\n \
:<Key>F13: key-input("Ctrl-[[25~")\n \
:<Key>F14: key-input("Ctrl-[[26~")\n \
:<Key>F15: key-input("Ctrl-[[28~")\n \
:<Key>Help: key-input("Ctrl-[[28~")\n \
:<Key>F16: key-input("Ctrl-[[29~")\n \
:<Key>Menu: key-input("Ctrl-[[29~")\n \
:<Key>F17: key-input("Ctrl-[[31~")\n \
:<Key>F18: key-input("Ctrl-[[32~")\n \
:<Key>F19: key-input("Ctrl-[[33~")\n \
:<Key>F20: key-input("Ctrl-[[34~")\n \
:<Key>KP_F1: key-input("Ctrl-[OP")\n \
:<Key>KP_F2: key-input("Ctrl-[OQ")\n \
:<Key>KP_F3: key-input("Ctrl-[OR")\n \
:<Key>KP_F4: key-input("Ctrl-[OS")\n \
```

1.8.5.7 Special Motif Bindings for HDSterm Keys

Some HDSterm keys have special definitions in the Motif widget set, such as F1, F4, and F10 (see the table in Section 2.2.9.1 for a complete list). If you try to modify these keys you must refer to them by that name in the translation table, as shown in the example above. You should also consider the consequences of these changes on the behavior of Motif window manager.

In the special case of F10, which is an accelerator key, you must also explicitly disable the key before you redefine it in order to disable the accelerator function, which normally overrides the key translation. In the example is given above, the osfMenuBar key is disabled as the first line of the file, then redefined in the body of the translation table.

HDS ViewStation: Font Handling

This chapter discusses the use of fonts on the ViewStation, including loading processes, font utility programs, and handling custom fonts.

Section 1.0 - HDS ViewStation

1.9 Font Handling

Font handling has changed with the release of X11R5. Under X11R4, each server (X terminal or workstation) had to use a set of .snf fonts specific to the server architecture. Separate directories of .snf fonts were maintained for each server type. An snf font file was loaded on request of the server.

A font server program was introduced with X11R5 that used portable, scalable fonts that were converted by the font server to a given server format on demand. This is a more economical method of font management.

1.9.1 Fonts in the X Window System

Many clients in the X Window System use special display fonts for their screen presentation. The fonts may be large or small, use bold or italic forms, or use different typefaces. To accommodate the needs of all the X clients, the X Window System provides a number of typefonts as part of its standard distribution. An X terminal must be able to find and use these fonts to handle X clients successfully.

The X Window System fonts offer different type styles, different sizes of characters, and different screen font resolutions for the characters (for instance, 75 dpi or 100 dpi). In addition, there may be different character sets involved, including multinational and non-Roman alphabets, such as Kanji and Katakana. Further, the X Window System supports a number of clients that use fonts from different sources, such as DEC proprietary fonts, Open Look fonts, or Postscript fonts.

These fonts are made available to the server from a font server or from a remote host. The HDS ViewStation uses either TFTP (Trivial File Transfer Protocol), FTP

(File Transfer Protocol), or NFS (Network File System) to locate and load the fonts dynamically on demand. You can make this protocol selection for each remote host from the Remote Hosts menu in Setup Mode.

1.9.2 Automatic Fontpath Configuration

The HDS ViewStation has features designed to make installation of fonts and fontpaths very simple. The fonts and font utility programs, along with other important files, are contained on the HDSware tape. When you load this tape onto one of your host systems, all that remains for you to do is enter that host address and directory location on the Setup Mode Main Menu as the HDSware host and path.

The ViewStation takes this information and prepends it to the directory structure of the tape and uses that information to locate the fonts. The fontpaths, the server code files, configuration files, etc. are all located automatically, so you don't have to enter them manually. The fonts distributed with the HDSware tape are in the correct .snf format and are ready to use.

The automatic fontpath configuration does not include font servers (which is a host-based service) but lists only the X11R5 font directories in the HDSware distribution.

1.9.3 Font Server Operations

The font server is a host-based X client that compiles fonts into the correct server format on demand. This client is usually started by the system administrator from boot files like */etc/rc.local*. You should be sure the font server is running if you intend to use it for your font access.

1.9.3.1 Putting the Font Server in the Font Path

The ViewStation's font path has to have an entry for the font server. This is typically a simple entry which specifies the hostname and the port number:

tcp/<hostname>:<port>

or tcp/hdssun1:7000

The default port number is usually 7000, but it can be assigned to other ports by your system administrator.

This entry should be included in the ViewStation's font path in the normal way. For example, a fontpath might be:

tcp/hdssun1:7000,/home/HDSware/hds-fx/fonts/Xol

1.9.3.2 Font Server Benefits

With normal font loading, a font file is transferred from the host's font directory into the ViewStation's memory. But with font server operation the ViewStation opens a connection to the font server on the host computer and the fonts are not actually loaded or cached by the ViewStation.

This can result in some saving of memory space on the ViewStation. For instance, a ViewStation starting up with Xdm and then starting HPvue with HPterm uses

220 KB of memory to load fonts (12 fonts) with conventional font loading, but uses no memory with a font server. This case illustrates the savings possible with font server usage. Depending on your use of fonts, this could be a benefit to your memory consumption.

1.9.3.3 Using Alternate Font Servers

If you want to have a backup font server, you can specify alternate font servers in the font server's configuration file (see the font server man pages for syntax). A typical entry would be:

```
alternate-servers=hostA:7000,hostB:7000,hostC:7000
```

Note however that these lists replace each other, so if hostA is unavailable, the alternate list used is from hostB and will not retry hostA unless it appears in the new list. It is therefore recommended that each host font server show all of the other font servers in their lists.

1.9.4 HDS ViewStation Font File Formats

The ViewStation can use .bdf, .pcf, or .snf font files directly. If any of these font formats are available to the ViewStation, it can use them. The .pcf fonts are usually available with an X11R5 font server.

A complete set of compressed .snf font files, in HDS ViewStation format, is available directly from HDS. You may contact your sales representative or distributor for more information about these pre-compiled fonts. If you use these fonts, you can load them onto a host and then set the fontpath automatically by using the HDSware host described above, or you can set the fontpath manually as described below.

1.9.4.1 ViewStation Compressed Font Files

The ViewStation uses font files in either compressed or uncompressed form. The font files are normally supplied as compressed files (with the .Z file extension) on the HDSware tape. You do not need to uncompress these fonts to use them on the ViewStation.

If you are compiling your own fonts, you can use fonts in an uncompressed form.

If you do need to compress the fonts, use 12 bit compression rather than 16 bit compression. The normal 16 bit compression uses a larger, less efficient table, so 12 bit compression is recommended. The command to use 12 bit compression is:

```
compress -b 12
```

1.9.4.2 Making Special .snf Font Files

If you have special application fonts or your own custom fonts (typically in .bdf format), you must use the **bdfto960** compiler supplied by HDS to compile them. If you have .bdf fonts the **bdfto960** compiler is specifically designed to create .snf fonts for the HDS ViewStation architecture and can be used in all situations. You can also use the **bdftosnf** compiler supplied with the MIT X distribution, but this does not work correctly in all situations and may not produce workable .snf font

files (refer to the discussion below). If you don't have .bdf files but only have .snf files, you can use the **hdsfc960** compiler, described in Section 1.3.6.

1.9.4.3 Font File Formats - .bdf and .snf

Font files contained in the MIT X distribution are in .bdf format (Bitmap Distribution Format, a generic, human-readable format). Other systems, such as DECwindows or Open Look, provide fonts in other formats, such as .dwf (DEC Windows Format) which are pre-compiled for the particular machine architecture they use. These fonts must be recompiled into a special .snf format (Server Natural Format) specifically for the HDS ViewStation (or the .snf format for the machine they are running on). Each server type has different .snf format requirements, based on its memory configuration and processor type, bit and byte order, etc. Other servers will use different .snf files, and you must create specific ones for the HDS ViewStation, as shown below.

The first step is to create a special directory for these font files. Then the .bdf fonts you need should be copied into the directory and converted to the .snf format for the ViewStation.

1.9.5 Using HDS Font Utilities

There are a number of font utility programs available from HDS. These utilities include a general **hdsfc960** utility for font management and other more specific programs. These programs are available in executable form for many types of systems, or as source code for all systems. The **hdsfc960** font utilities are available from any HDS office.

The HDS font utility programs create .snf files in HDS ViewStation format from font files of any format. They read the font data from a remote host server display (such as the system console) using normal X protocols and reconstruct the .bdf file, then use their own **bdfto960** compiler to make the .snf font files. These font utility programs can use either generic .bdf files or precompiled font files, such as .dwf or .ps. They can also run on any kind of host architecture and create properly formatted files.

1.9.5.1 HDS Font Utilities

HDS has prepared a set of font utility programs to make it easier for HDS ViewStation users to compile and manage font files. These programs are included on the HDSware tape, along with X11R4 fonts which have been precompiled for use with the HDS ViewStation. This tape is available from any HDS office.

There are several separate font utility programs:

hdsfc960 - The general program to make .snf files from any font source file

bdfto960 - The program to make HDS .snf files from .bdf files

hdsmkdir - The program to create a *fonts.dir* file for a font directory

getbdf - The program to create .bdf files from any font source

These utility programs are available in the HDSware distribution in the directories *./hds-fx/utils*. There is an */execs* directory with executable files for a number of host systems. There is also a */src* directory with source code and sample files for compiling these utility programs for other hosts.

1.9.5.2 bdfto960 Font Utility Program

If you have .bdf files you want to convert, use **bdfto960**. The command line arguments are set by default to the HDS ViewStation format. This compiler is similar to the MIT **bdftosnf** compiler except it works on any type of machine architecture. You can use it in the same way as the MIT compiler.

For instance, if you want to manually convert some .bdf fonts, use this procedure. The HDS **bdfto960** utility has the command line arguments set automatically, but you must enter them manually with the MIT **bdftosnf** command.

Converting .bdf files to .snf files with the MIT **bdftosnf** command is done with the command line specifying the arguments for the HDS ViewStation architecture:

bdftosnf -p4 -u1 -L -l file.bdf **>** file.snf

where:

-p4 sets glyph padding to 4

-u1 sets scanline padding to 1

-L sets Least Significant byte first

-l sets Least Significant bit first

Converting the large numbers of fonts can sometimes be a lengthy process. This little program will help you handle the conversion process by processing all the files in a loop. These same programs can also be used with **bdfto960**, but it is not necessary to enter the arguments on the command line.

For the C Shell: /bin/csh

```
unix% foreach i (*.bdf)
        echo $i
        bdftosnf -p4 -u1 -L -l $i > $i:r.snf
      end
```

For the Bourne Shell: /bin/sh

```
unix$ for i in *.bdf
      {
        echo $i
        bdftosnf -p4 -u1 -L -l $i > `basename $i .bdf`.snf
      }
```

(Note that ` is an accent grave; be careful with spaces in these commands)

1.9.5.3 hdsmkdir Font Utility Program

Creating a *fonts.dir* file with **hdsmkdir** is also a simple operation. It works like the **mkfontdir** command, but it works on any machine architecture and accepts more than one directory name.

1.9.5.4 getbdf Font Utility Program

The **getbdf** program creates a .bdf font file from a file of different format. It reads the font data using various X tools (xlib), and reconstructs the original .bdf file. This utility is especially useful for systems that have only precompiled font files.

1.9.5.5 hdsfc960 Font Utility Program

The **hdsfc960** utility program is more general. It creates HDS .snf font files from any font file type, including precompiled fonts. It uses X tools (xlib) to read the font from the host server display, then creates a .bdf file from that font data using **getbdf**, then creates the .snf file using **bdfto960**, and finally creates a *fonts.dir* file using **hdsmkdir**.

To use **hdsfc960**, you must first create a directory for the HDS .snf fonts. It can be at any location and can have any name, but it must be empty. The **hdsfc960** program converts all the font files in the current fontpath, so you should be careful to limit the fontpath on the target server to those fonts you want to recompile. (The best way to do this is to create a separate directory containing only the fonts you want to recompile, then restrict the fontpath of the host target display to that directory.) The command line is:

hdsfc960 -d <*IP address of target display:display*> <destination directory>

or, for example,

hdsfc960 -d 128.91.6.1:0 /usr/XllR5/HDS/fonts/100dpi

You should enter the target server IP address (the target server is a display - such as the host's console - that can display the fonts) explicitly so you don't convert any fonts you don't want to. The **hdsfc960** program will convert all fonts in the target server's fontpath, so you should use **xlsfonts** beforehand to make sure the font list is correct.

Note that **hdsfc960** uses the other HDS font utility programs, so they must be available and have the correct execute privileges set.

1.9.5.6 Handling .bdf Fonts and Fonts in other Formats

There are large directories of .bdf font files supplied with the standard MIT X distribution. You may not want to use all of these fonts or create .snf files for all of them. This choice depends on the X clients and applications you want to use. All of these fonts must be compiled into .snf format to be used on the ViewStation (or use the pre-compiled HDS .snf fonts from the HDSware tape).

Font Sizes

Other important font considerations include choosing the correct 75dpi (dpi = dots per inch) or 100dpi files for your HDS ViewStation. These sets of files are pro-

vided for monitors with different screen resolutions. (See Section 1.1 for dot resolutions of various FX monitors.) The appearance of a font on the screen is best when the dots per inch resolution on the screen is as closely matched as possible to the number of dots provided for each character in the font file.

(This can become a rather complicated question for users concerned about exact character size. For instance, a 24 point font from the 75dpi directory displayed on a monitor whose resolution is actually 85 dpi will be slightly smaller than 24 points. [The calculation goes like this: Since a point is about 1/72nd of an inch, a 24 point character from the 75dpi directory is about 24 dots high. This means that 24 dots displayed on 85 dpi results in a smaller character - 24/85ths compared with 24/75ths of an inch.] On the other hand, the same 24 point font from the 100dpi directory will be larger than 24 points when displayed on the same monitor [24/85ths compared to 24/100ths of an inch]. Since most applications where this is important, such as WYSIWYG publishing programs, take these factors into consideration with internal mechanisms for their screen displays and printing, most users won't need to be concerned about it. If this is an important consideration, put the preferred font directory, 75dpi or 100dpi, first in your search path. This will insure that when a particular font is requested [if it is available in both directories] it will be loaded from that directory.)

The HDS ViewStation uses either 75dpi or 100dpi font files, depending on the monitor used. Some application programs specify specific dpi values in their font requests and may require loading both 100dpi and 75dpi font directories. The correct font size is calculated by measuring the horizontal display width in inches, then dividing it into the horizontal resolution of the screen to give pixels/inch (refer to Section 1.1 for these values). Then choose the font resolution closest to that value, 75 or 100 dpi. This information is also provided by the X command **xdpyinfo**.

If your .bdf fonts appear as files with a .Z extension, they are compressed files (all files from the MIT distribution are compressed). You can convert these files easily with the uncompress command:

```
uncompress filename.bdf.Z
```

Or, using a wildcard character:

```
uncompress *.bdf.Z
```

The new files appear with the .bdf extension only. They then may be compiled into .snf font files.

Other systems, such as OPEN LOOK (usually *.ff* or *.f3b* font format), Hewlett Packard (in .scf format), or DECwindows (in *.dwf* format), have their own sets of precompiled fonts and font directories necessary for their operation. Some of these fonts may be in a scalable format (which means that they are general formulas for fonts and individual font files, in the desired sizes, must be created before they can be used); scalable fonts often have the *.f3b* extension on their file name, and their extended font name has 0-0's in the size and spacing fields. Other common font formats include *.afb* and *.afm*, which are Adobe formats, bitmap and

metric. On SUN systems, scalable fonts are converted to various size font files with the **makeafb** utility program, and these *.afb* files are converted to *.bdf* files with the **convertfont** utility. Other systems may use different formats and/or different utility conversion programs.

All of these font format types may be converted to HDS .snf format by using the HDS font utility programs, specifically, the **hdsfc960** font compiler. This compiler takes font files of any format (even if they can be read only by the particular server they belong to), reads them with general X routines, and creates HDS format .snf files.

1.9.6 Loading Fonts to the ViewStation

There are a few simple requirements for using fonts. The ViewStation can read .pcf, .bdf, or .snf font formats directly. If you are using a font server, the server takes care of differences in server format, so you don't have to have .snf fonts for the ViewStation.

- Fonts must have the correct format (.pcf, .bdf, or .snf). This was discussed above.
- The fontpath must be correctly given to the ViewStation so it can locate the fonts. If you're using a font server, its path must be included.
- There must be a fonts.dir directory.
- There should be a fonts.alias directory (optional).

1.9.6.1 Setting the Fontpath

The fontpath is entered automatically if you use the automatic HDSware configuration feature (see Section 1.3.4) of the HDS ViewStation. You can also enter the fontpath in Setup Mode on the User Preferences menu (see Section 1.5.6.5).

A typical path might be:

/FXUtilities/hds-fx/fonts/75dpi/,/FXUtilities/hds-fx/fonts/misc

where FXUtilities is the HDSware host and path.

If you have a font server, your path might be:

tcp/hostA:7000

You can use multiple hosts and multiple directories separated by commas. If one host or directory is not reachable, the ViewStation still uses the remaining ones. If the same .snf fonts are located in more than one path, or on more than one host, only the first path correctly reached will be used.

Backup Font Hosts

The font server has an alternate font server list as part of its configuration. This is discussed in Section 1.9.3.2.

If you use a regular fontpath, you can specify that a font host act as a backup, and there is a little trick to make this happen. You specify the same "local mount directory" for each host, then specify the remote directory for each host (on the Remote hosts menu in Setup Mode), so the resulting local font paths are the same. The

server will try each host which matches the local mount directory until it finds one that responds. The Local Mount Directory specified must be the same for each host, and only one font path using that mount point is entered in Setup Mode.

For example, on the Remote host menu:

Remote Host 1:	Host Name: hdssun3
	Network address: 128.91.3.1
	Local Mount Directory: /fonts
	Remote Mount Directory: /usr/lib /X11/fonts/hds-fx/fonts
Remote Host 2:	Host Name: hdssun6
	Network address: 128.92.7.5
	Local Mount Directory: /fonts
	Remote Mount Directory: /home/HDS /hds-fx/fonts

and so on.

If the font path entered in Setup Mode is "/fonts/75dpi,/fonts/misc,/fonts/100dpi", then Remote host 1 will be contacted first for all fonts, with Remote host 2 active as a backup in case host 1 is not available.

1.9.6.2 Verifying the Fontpath Setting

You can use the xset command to set the fontpath. For instance,

```
xset fp= /usr/lib/X11/hds-fx/fonts/100dpi/
```

The command to update the fontpath setting is:

```
xset fp rehash
```

The HDS ViewStation automatically updates the fontpath when you Apply Changes in Setup Mode (note the message at the bottom of the Setup Mode window). You will probably also want to save the fontpath setting in the ViewStation's NVM. If you have changed the fontpath, you should verify that the fontpath is correctly registered or save it in NVM and restart the ViewStation to be certain it is correctly read.

You can check the fontpath with the **xset** query command:

```
xset q
```

which shows many of your current server settings, including the fontpath.

You can also use the **xlsfonts** command to show the available fonts in your fontpath. This is useful because it shows the valid aliases as well as normal fonts.

If you have set the fontpath in Setup Mode, you will notice that the fontpath includes HDS$$ROM in addition to your designated directory. This indicates that the ROM fonts are in the search path. If you wish to append them to your font-

path (a good idea), you can do it with the xset command as follows (the command **xset +fp** puts the new directory at the beginning of the search path and **xset fp+** puts it at the end):

xset fp+ /HDS\$\$ROM/

At this point, your **xset** query should show the correct fontpaths, and your **xlsfonts** query should list all the fonts you have available. You can test the operation of your loaded fonts by running an xterm program specifying one of the particular fonts in your fontpath. Use a command line such as:

xterm -fn timB18

to load the 18 point times bold proportional font. This xterm will look strange because of the proportional font, but the font appearance is distinctive so you'll know the font loaded correctly. If the font doesn't load, a default will often be selected and an error line appears saying the font couldn't be found.

If you specify a font by its entire font name (not its file name), you can use wildcard characters in the name, but you must use single quotation marks around the font name, such as:

xterm -fn '*times-bold-*-180-*'

The wildcard characters are special UNIX characters and the entire name must be quotation marks or the wildcards each preceded by a backslash \, such as:

xterm -fn *times-bold-*-180-*

1.9.6.3 Making a *fonts.dir* File

A *fonts.dir* file is essential for the correct operation of X font handling. This file MUST be present in your fontpath directory. It is a kind of database for fonts that the ViewStation, and all X servers, use to locate fonts. You must create a *fonts.dir* file after all your conversions are complete and the unnecessary .bdf files have been removed. This file is necessary for any type of font loading. Make certain you are in the correct HDS .snf file directory. If a *fonts.dir* file exists, delete it; then just issue this command:

hdsmkdir or **mkfontdir**

This command automatically creates a *fonts.dir* file. Verify that it appears in the directory by using the **ls** command. The format of the *fonts.dir* text file is:

< *number of fonts in the file* >

<filename without path> <white space> <fontname>

<filename without path> <white space> <fontname>

<filename without path> <white space> <fontname>

For example, a typical (but untypically short) *fonts.dir* file would be:

4

courBO10.snf -adobe-courier-bold-o-normal--10-100-75-75-m-6-iso8859-1

courBR12.snf -adobe-courier-bold-r-normal--12-120-75-75-m-70-iso8859-1

charBI10.snf -bitstream-charter-bold-i-normal--10-100-75-75-p-72-iso8859-1

charBI12.snf -bitstream-charter-bold-i-normal--12-120-75-75-p-74-iso8859-1

1.9.6.4 Making a *fonts.alias* File

Another commonly used font file is optional, but is sometimes used to substitute different names for specific fonts. This is the *fonts.alias* file. The ViewStation looks for this file and uses it if it is present.

The format of the *font.alias* text file is:

<alias> <white space> <fontname>

<alias> <white space> <fontname>

<alias> <white space> <fontname>

For example, a *fonts.alias* file might be:

-courier-bold-r-normal--*-120-* 9x15

-courier-bold-i-normal--*-120-* 9x15

FILE_NAMES_ALIASES

This means that if a client requests the Courier Bold Italic 12 font, the ViewStation checks the *fonts.alias* file and substitutes the 9x15 font.

The string FILE_NAMES_ALIASES is a special keyword which creates an alias for each font file listed. The alias is formed by deleting the file extension from the font file name. Thus the alias for 6x10.snf is just 6x10, which can be used when the client calls that font, as in **xterm -fn 6x10**.

Create this file with a text editor, such as *vi*, and save it as "fonts.alias" in the HDS font directory. Be careful with spelling in this file. If a name is misspelled, the font won't be aliased correctly. If the keyword FILE_NAMES_ALIASES is misspelled, this function won't operate.

HDS ViewStation: PC Card Devices

This chapter describes the ViewStation's PC Card device support and its operation.

Section 1.0 - HDS ViewStation

1.10 PC CardDevices

The PC Card, formerly PCMCIA (Personal Computer Memory Card Interface Association), interface is a hardware and software specification for add-on devices. This interface is supported on the HDS ViewStation for expansion devices.

The HDS ViewStation currently supports these PC Card devices:

Memory cards, 2.5 and 5.0 MB

Hard disk drives, 20 and 40 MB

Token Ring network card

HDS Audio card

New PC Card devices are being added to the ViewStation's capability.

1.10.1 ViewStation PC Card Adapter Card

The HDS ViewStation has an adapter card for PC Card devices. This is a daughter board which can be added to the main circuit board. It supports two PC Card devices, one internal and one external, as well as an internal FlashPROM card. Each device is independent and equivalent.

The PC Card Adapter card can be added to the ViewStation by plugging it into the main circuit board connector. The ViewStation's back panel has an opening for the PC Card connector but may have a temporary covering installed.

1.10.2 ViewStation FlashPROM

This is s special HDS FlashPROM memory card that uses a special connector on the PC Card adapter card. This permits both PC Card slots to be used for PC Card devices..

If you have a PC Card adapter card with the HDS FlashPROM memory card installed, you cannot use a hard disk drive in a PC Card slot without first removing the FlashPROM card and installling a PLCC boot prom.

1.10.2.1 Loading New Server Code into flashPROM

If you have a new version of the ViewStation server code and want to load it into the flashPROMs, follow this procedure:

- Load the server code file - Load the new server code from the HDSware tape into a directory on your boot host (it must be a .hds file). The server code is in a directory called *./hds-fx/image*, with several versions having names like this:

  /hds-fx/image/Vnn-D.hds

  /hds-fx/image/Vnn.hds and so on.

  The file names indicate the version level of the code (Vnn) and whether this server code file includes the capability to download clients (-D), such as window managers and other HDSperx clients.

- Open a Console Window - Open a Console session (from Setup Mode or a remotely via rlogin). Note that flashPROM commands are password protected. If a password is set, you must enter it in the Console session before you can use the flashPROM commands. Entering the password changes the Console session prompt from hds> to HDS>.
- Enter this command from the HDS> prompt:

  flash server <*full path/filename*> (this must be a .hds file)

  flash server /128.91.3.1/home/HDSware/hds-fx/image/V3.0-D.hds

  The Console window will report the progress of the load, including the erasing of the FlashPROMs and loading the new server code file.

- If you want to burn a new bootprom, you can use this command:

  flash boot <*path/filename*> (this must be a .bin file)

  Note that some ViewStations orignally ordered without FlashPROM have the bootprom in a PLCC ROM on the main circuit board rather than on the FlashPROM board. If you have the PLCC bootprom configuration, the bootprom cannot be reprogrammed.

The new server code you selected will be loaded into the flashPROM. Messages reporting the erase procedure and the load progress appear in the Console session.

You could separately flash the bootprom file and/or the server code file into the storage device, depending on your preference. The separate flash boot and flash server loading for the generates a "ROM" boot for the ViewStation. Loading the

server code as an HDSware file (*./hds-fx/image/Vnn-D.hds*) on the PC Card device generates a "Network" boot, even though the network in this case is entirely internal to the ViewStation.

The system administrator, using remote login to the Console session, can burn flashPROMs for all the ViewStations on the network. Note that these procedures are all password protected, so unauthorized users cannot use these commands.

1.10.3 ViewStation ROM cards

The ViewStation can use a number of PC Card ROM cards. Two sizes are supported, 2.5 MB and 5.0 MB.

These devices can be used for storing the ViewStation's server code and any additional files you might need, such as the window manager, other local clients, selected fonts, etc. In the factory configuration, the ROM cards come with only the ViewStation's server code loaded.

You can program the ROM cards (and the hard disks) with the Flash command, described below.

1.10.4 Hard Disk Drive Storage Devices

The HDS ViewStation supports PC Card hard disk drives. Two sizes are supported, 20 MB and 40 MB.

These hard disk drives can be used for storing the ViewStation's server code and other ViewStation files, such as window managers, local clients, fonts, etc. In the factory configuration, the hard disk drives come with the HDSware software loaded. The entire HDSware distribution does not quite fit on the 20 MB disk. On the 20 MB disk, the */utils* and */contrib* directories are not loaded (since these contain utility programs that run on the host, this is a reasonable omission).

You can load your own files onto the hard disk drive with the Flash command, described below.

If you have a hard disk installed, you must use the hard disk for booting - you cannot use an HDS FlashPROM card. The PC Card hard disk (with a PLCC boot prom installed on the main board) replaces the FlashPROM card.

1.10.4.1 Adding a PC Card Hard Disk Drive

If you are adding a PC Card device (such as a disk drive) to an existing ViewStation, the procedure is simple. Follow these steps:

- Check the boot prom version, it must be version 2.4 or greater, to recognize the PC Card device.
- Install the PC Card adapter card. Open the ViewStation base and use the long connector on the right rear edge of the circuit board. Make sure it is correctly seated and that the pins are correctly aligned. Remove the coverings for the backpanel openings for the PC Card and / or Audio connectors.
- If you already have a PC Card adapter card, you may have to install the PC Card device itself. It simply plugs into a socket on the adapter card. Be careful

with this installation. The pins are small and tightly spaced so be sure to insert the device carefully.

- Boot the ViewStation from the network, using Monitor Mode. You should load HDSware version 2.2 or higher.
- Use the *config.fx.dsk* and *clients.config.dsk* files or make the appropriate changes to your own *config.fx* and *clients.config* entries for the files you are putting on the device. These modified files should be part of your new flash file system.
- Use the fdisk command to format the hard disk.
- Use the flash files command to load your files onto the disk.

At this point, your files are loaded and your configuration files specify the correct paths so your own ViewStation (or other ViewStations) can find them. All you have to do is turn on the ViewStation.

1.10.4.2 Using the Hard Disk Drive

The HDS ViewStation's hard disk drive is especially convenient in some system environments, such as remote locations on a PPP link, or isolated environments. The hard disk can be used for booting the individual ViewStation where it is installed and also any other ViewStations. It can serve as the HDSware file server for fonts, local clients, etc.

For Internal Booting and File Service

With the internal hard disk drive, the ViewStation boots and loads its files from the disk. It accesses the disk as if it were a network device, but uses the IP loopback address (127.0.0.1) and the directory path of */dev/flash* as its HDSware location. Local clients, fonts, etc. are read directly from the disk.

These settings are made at the factory, so all the user has to do is turn on the ViewStation and all the software is immediately available. ViewStations with a hard disk installed should use the special configuration file *./hds-fx/config/config.fx.dsk*.

For External Booting and File Service

Other ViewStations can use this HDSware location by specifying the HDSware location as the disk-equipped ViewStation's IP address as the HDSware host and */dev/flash* as the HDSware directory.

This permits booting, client loading, font loading, etc. to run from a ViewStation instead of a host computer.

1.10.4.3 Using the fdisk Command

The fdisk command is a Console window command. It is password protected. You can use it to format or erase ViewStation FlashPROM and PC Card devices. It is available from any Console window, either local on the ViewStation, or from a remote Console window from another device.

The fdisk command must be used for disks being programmed for the first time. ViewStations equipped with ROM or hard disks from the factory do not need to use the fdisk command.

1.10.4.4 Using the flash Command

The flash command is a Console window command. It is password protected. You can use it to program ViewStation FlashPROM and PC Card devices. It is available from any Console window, either local on the ViewStation, or from a remote Console window from another device.

There are three parts of the flash command:

flash boot <*filename*>

flash server <*filename*>

flash files <*filename*>

1.10.4.5 The flash boot Command

The flash boot command programs the bootprom on the FlashPROM card.

1.10.4.6 The flash server Command

The flash server command programs the server code on the FlashPROM card. It can also be used to program a ROM card (though probably not a hard disk).

The flash server command should be used first because it allocates a reserved block at the top of the available memory. Subsequent loading of the ROM card by using the flash files command will not overwrite the reserved server block but use only the available memory remaining. The reverse procedure will not work, since the flash server command will overwrite any flash file load.

Note that all the files in your flash file system are erased when you load a new version of server code using the "flash server" command. This means that you must reprogram the flash file system using "flash files" each time you use the "flash server" command.

1.10.4.7 The flash files Command

The flash files command loads a special file into the available flash or disk memory. This special file must be prepared with the "flashfilesys" script. The procedure is outlined below.

There are several arguments for the flash files command:

flash files [-rs] [-n] [filename]

-rs Remove Flash server code image

-n Display number of bytes available for Flash filesystem

To load files onto your hard disk, follow this procedure:

- Check the available memory with the -n command argument:
  flash files -n
- Remove the server code file (if you've loaded it in a reserved section) with the -rs command argument:
  flash files -rs
- Select the files to load.

If you want to load the HDSware software, there is an input file already made for you. This file is *./hds-fx/utils/src/flashfilesys/disk.files*. Check it to see that the files included there are the ones you want.

The HDSware distribution on CD-ROM includes a .ffs file constructed from standard HDS files.

If you want to use your own files, create a text file listing the files you want to load. Make sure the disk has sufficient space for the files you choose. The file should look like this:

```
#/home/hdssun1/mikeh/config/config.test     config.fx
/home/hdssun1/mikeh/bin/hdsxclock.Z         /bin/hdsxclock.Z
#/home/hdssun1/mikeh/clients/xfed.Z         /bin/xfed.Z
#/home/hdssun1/mikeh/lib/app-defaults/Xfed  /lib/app-defaults/Xfed
/usr/lib/X11/i960/fonts/misc                /fonts/misc
/usr/lib/X11/i960/fonts/lotus               /fonts/lotus
                                            .       .
```

Each line of the file specifies the real file name and its corresponding flash file system file name. When a directory is specified, all files within a directory are included. Subdirectories are not included. Entries that begin with a '#' character are ignored. Blank lines are allowed. In this example, a configuration file is stored as "config.fx" and two font directories are stored as "/fonts/misc" and "/fonts/lotus". The ViewStation checks for (and decompresses) compressed files with the extension .Z, so you can use compressed files in your flash file system.

Note also that this sample file contains a client that is part of HDSperx on the ViewStation (hdsxclock) which has its resources set in the local configuration files, *clients.config*. There is also another executable file (xfed - X font editor) that has its resources included in */dev/flash* as a separate file. You would have to modify the xfed resources to specify this new resource file location as the ViewStation Flash File system.

If you want to add or change files to your Flash File system, you must change the list and go through the procedure again.

- Create the output file with flashfilesys.

  There is a shell script called "flashfilesys" in the HDSware directory under *./hds-fx/utils/src*. The flashfilesys script organizes the files you've selected and creates a header for the files that includes the file names and their total length. This information is used to test whether there is sufficient space for storing it in the flash file system and that the files are readable, etc.

  The flashfilesys script is an executable file that runs on your host computer.

  This shell script takes two arguments, the name of the input file you generated in the previous step, and the name of the output file for the flash file system file.

```
flashfilesys <inputfile.txt> <outputfile> or

flashfilesys <FX_disk.file> <V30.ffs>
```

This .ffs output file is the file actually loaded into the flash file system.

- Load the file into the Flash File System

  Store the output file in the Flash File system with the "flash files" command in the Console shell window. All of the flash file commands are protected by the Setup Mode password. This command is:

  flash files <outputfile> or

  flash files </128.91.3.1/HDSware/hds-fx/utils/src/flashfilesys/V30.ffs>

Note that this is a large file (approx. 60 MB for the HDSware distribution, less if you don't load the documentation, special fonts, etc.) and that TFTP file transfer protocol has a 16 MB limitation. You must use FTP or NFS to transfer this file to the ViewStation's hard disk.

1.10.5 Token Ring Interface Card

The HDS ViewStation supports Token Ring communications with its PC Card adapter card and a Token Ring "HDS PC Card Token-Ring 16/4 Adapter" card. This is an available option for all ViewStation models.

The ViewStation's Token Ring communication complies with IEEE 802.5 standard and uses IEEE 802.2 packet types. The Token Ring communications option requires HDSware version 3.2 or higher. [Token Ring communications was also implemented with HDSware v3.0.2, but required a special bootprom. Note that if you are using the v3.0.2-E bootprom with 4Mb capacity, you must have a jumper set (position JP7 on the ViewStation FX; position JP6 on ViewStation Ultra) to indicate the higher bootprom capacity.]

The ViewStation can connect to Token Ring, Ethernet, and PPP networks simultaneously. It can have up to four IP addresses (one for each network) and can act as a network router, including passing information from one network to another. Refer to Section 1.4 formore complete discussion.

ViewStation local clients, such as the IBM 3270 terminal emulator, add powerful capabilities to the Token Ring communications feature. This terminal emulation permits direct connection to IBM mainframe systems on Token Ring networks (at the same time keeping other ViewStation connections and emulations open).

An additional ViewStation feature, the HDS Developer's Kit, allows ViewStation customers to write and compile their own application programs to run on the ViewStation. This includes OSF/Motif applications which could be used on IBM mainframe computers without having either X or Motif running on the mainframe.

1.10.5.1 Token Ring Specifications

ViewStation Hardware Interface

The HDS ViewStation uses "tr1" to designate the token ring interface to the PC Card internal slot and "tr2" to designate the PC Card external slot.

Token Ring communications can use either shielded twisted pair connectors (called Type 1 cabling) or unshielded twisted pair connectors (called Type 3 cabling). Cable adapters (DB9 or RJ45) are supplied with the Adapter card.

Network cable quality often determines whether 4 or 16 Mb ring speed is appropriate. Token Ring speed is jumper selectable for either 4 Mbps or 16 Mbps. This is controlled by jumpers on the JP9 block for the ViewStation FX and on the JP11 block on the ViewStation Ultra. Jumper pins 5-6 select the ring speed for the internal PC Card slot and jumper pins 7-8 select the ring speed for the external PC Card slot. Ring speed 4 Mbps is selected when the jumper is in place, and ring speed 16 Mbps is selected when the jumper is off. This configuration should be selected with the initial order so the correct configuration is installed, tested, and shipped.

ViewStation Software Interface

The hardware interface is used in the boot monitor to set the Token Ring IP address and netmask as follows (tr1 designates the internal PC Card slot and tr2 designates the external slot):

```
ip trx nnn.nnn.nnn.nnn

nm trx nnn.nnn.nnn.nnn
```

In the configuration file, the resources for configuring the Token Ring IP address, netmask, and MTU (Maximum Transmission Unit, default is 2052) are these:

```
HDSsetup.pcmcia.internal.tr.ipaddress: nnn.nnn.nnn.nnn

HDSsetup.pcmcia.internal.tr.netmask: nnn.nnn.nnn.nnn

HDSsetup.pcmcia.internal.tr.mtu: [ 516 | 1500 | 2052 | 4472
| 8144 | 11407 | 17800 ]

HDSsetup.pcmcia.external.tr.ipaddress: nnn.nnn.nnn.nnn

HDSsetup.pcmcia.external.tr.netmask: nnn.nnn.nnn.nnn

HDSsetup.pcmcia.external.tr.mtu: [ 516 | 1500 | 2052 | 4472
| 8144 | 11407 | 17800 ]
```

These settings are also available in the ViewStation's Setup Mode client.

ViewStation Token Ring communications supports source routing, so it will work on bridged networks. XDMCP and BOOTP are supported on the Token Ring network for both direct and broadcast access.

1.10.5.2 ViewStation Token Ring Controls

From the ViewStation's Console window, the "srt" shell command displays, adds, and deletes source route entries for token ring communications.

```
srt         displays all source route entries

srt -s      add/modify a source route entry
```

```
srt -d      delete a source route entry
```

If the Token Ring adapter becomes unplugged from the Token Ring network, the user can issue a command to re-insert the adapter onto the network:

```
ifconfig trX up
```

(where trX = tr1 for the internal slot and tr2 for the external slot)

This reconnects the token ring adapter.

The "token" shell command displays token ring interface statistics.

1.10.6 Multimedia Devices

The HDS ViewStation has a PC Card card that supplies audio capability. This card inserts into the PC Card adapter card. Note that connectors for Audio In and Audio Out have to be fit into the back panel.

A complete discussion of ViewStation Multimedia operation is in Section 2.7.

1.10.6.1 Configuring the ViewStation for Multimedia

Verify that the ViewStation has an audio card on the PC Card adapter and that the audio card is functioning normally. ViewStations which have the audio card installed have two stereo audio connectors on the back panel labeled input and output. The Multimedia ViewStation also has an internal speaker installed. Selection of the correct audio input and output devices is available in Setup Mode (User Preferences menu).

Multimedia support requires software version V3.0 or higher. You must run this code to access the Multimedia card. You must also use the new entries showing the Multimedia Audio clients in the ViewStation Client Configuration File, */hds-fx/config/clients.config*, in the HDSware 3.0 distribution.

Locally, the ViewStation supports a */dev/audio* device which is compatible with the Sun audio device interface. To determine if a particular ViewStation supports audio, execute the command

ls /dev/audio

in a ViewStation Console window. If the result is '/dev/audio not found', then the ViewStation either does not have an audio card, the audio card may be improperly installed, or an older software version may be running which does not support the audio card.

Attach speakers or headphones to the stereo jack labeled 'output' on the back panel of the ViewStation. This is all the hardware configuration required for HDSperx Multimedia support.

1.10.6.2 ViewStation Multimedia Audio Specifications

The HDS ViewStation Multimedia unit has both input and output jacks on the back panel, using stereo 3.5mm connectors.

Input jack specifications:

There are several types of input, selectable either from the Audio Devices menu (User Preference manu of Setup Mode) or directly from the HDS audio client.

These settings select the correct input levels for the audio signal. If you are using your own devices, choose the setting that gives the best audio response.

- HDS ViewCam - this is a video camera sold by HDS
- HDS Headset - this is a headset / microphone sold by HDS
- CallPort Speaker / Microphone - this is a integrated microphone and speaker that uses echo suppression for optimum response in noisy environments
- HDS Microphone - this is a microphone sold by HDS
- HDS Tuner - this is the TV / cable receiving device sold by HDS
- VCR - this sets an input level appropriate to most video cassette recorders
- Line Level Audio - this is a normal line level input setting
- Unspecified - this is for other input levels

Output jack specifications:

- HDS Internal Speaker - this sets the level for the ViewStation's internal speaker (2 W output at 8 Ohms. Harmonic distortion is 0.055% with power at 1 W)
- External speakers - this sets the level for self-powered external speakers
- HDS Headset - this sets the level for the HDS Headset (0.75 W at 1 V (peak-to-peak) when driving loads of 25-32 Ohms. Harmonic distortion is approx. 0.005% with 32 Ohm load)
- CallPort Speaker / Microphone - this sets the level for the CallPort integrated speaker / microphone with echo suppression

These signals can be either stereo or mono and have 8 or 16-bit audio sampling rates from 5 to 48 KHz. These settings, as well as signal inputs, are selected from the Setup Mode or from the options in the **HDS audio** client menu. This client is available from the HDSperx menu.

1.10.6.3 ViewStation Multimedia Audio Capabilities

The ViewStation Multimedia implementation gives you easy control by means of an audio manager client, **hdsaudio**, running locally on the ViewStation, with a Motif interface for user selections. Other local clients, such as **hdsplay**, permit use of sound files from the network. The ViewStation can mix its input with these network audio sources. **NetAudio** is also supported. The ViewStation's Audio clients are described in Section 2.7.

1.10.7 Other PC Card Devices

New items are being added to the ViewStation's PC Card support all the time.

HDS ViewStation: Security Considerations

This chapter discusses the ViewStation's internal security, security for remote shell processes, and some system security techniques like host access. There is also a summary of the ViewStation's optional CMW security.

Section 1.0 - HDS ViewStation

1.11 HDS ViewStation Security Considerations

There are a number of important issues regarding security on an X terminal. Since the HDS ViewStation has a Unix micro-kernel as its system architecture, considerations about ViewStation access and client operation also arise.

1.11.1 ViewStation Passwords

The HDS ViewStation has a password that can be set in the User Preferences menu of Setup Mode. The password must be entered any time a protected field is edited. Only one entry of the password is needed for each Setup Mode or Console Window session. When Setup Mode or the Console Window is exited, the password must be entered again to edit protected fields.

A password entered in a configuration file, or from a bootp response is compared to the password stored in the ViewStation's non-volatile memory. The passwords must match or the operation fails.

1.11.1.1 Changing the ViewStation Password

The ViewStation's password can be changed locally by editing the field in Setup Mode (User Preference menu), or remotely by using the password command.

Using a remote login to the ViewStation, the Console Window lets you enter the password command with the change option:

```
password change
```

which then lets you change the ViewStation's internal password. This operation is password protected.

1.11.1.2 Setup Mode Password Protection

The password protects sensitive fields in Setup Mode, such as IP information, fontpaths, etc. These fields cannot be edited except by entering the password.

Further protection is available by using resources in the configuration file. The resources:

HDSsetup.password.is_global: yes

HDSsetup.hide_icon: yes

let you make all of the fields in Setup Mode and all local clients password protected and then hide the Setup Mode icon (the icon can be restored by hitting the *Alt-SysRq* key). Refer also to Section 1.5.1 of this manual.

Further, entries in the configuration file override entries in Setup Mode and NVM. In this way, the system administrator can make settings for the ViewStation that the user cannot change.

1.11.1.3 Console Window Password Protection

Commands in the Console Window that control the ViewStation's internal operation are divided into two groups, with sensitive commands being password protected.

This is the summary of Console Window commands.

```
Summary of valid commands:
arp             cat             cd              date
env             echo            ether           exit
hdsmemory       help            hwconfig        list
log             logpg           ls              more
netstat         nvm             opendev         password
ports           ps              pwd             rehash
setenv          unsetenv        version
Summary of privileged commands:
db              dw              fdisk           flash
kill            reboot          restart
```

You must enter the password to use any privileged command, including password change.

1.11.1.4 HDSperx Password Protection

The HDS ViewStation's local clients can be individually password protected by means of client attributes in the clients.config file. These clients include the HDSperx clients and also many shell commands.

Each client has an entry in the clients.config file. The "passwdRequired" attribute can be added to these entries to make the client password protected.

If the passwdRequired attribute is set to true, the client can only be run if the ViewStation user has been able to present the ViewStation's password before

attempting to start the client. The password is entered by using the 'password' command in the HDS Console Window.

Almost all Console shell commands can be password protected, according to the following list.

You must set passwdRequired for these commands:

arp	cat	date	ether
hdsmemory	help	hwconfig	list
log	logpg	ls	more
netstat	nvm	opendev	ports
ps	pwd	rehash	version

The passwdRequired resource is set by default for these commands:

db	dw	fdisk	flash
kill	reboot	restart	

These commands cannot be used with the passwdRequired resource since they are integral parts of the shell itself:

cd	echo	env	exit
password	setenv	unsetenv	

You can set this resource just like with any other HDSperx client, such as:

```
hds.client.list.passwdRequired: true
hds.client.ps.passwdRequired: true
hds.client.netstat.passwdRequired: true
hds.client.opendev.passwdRequired: true
```

1.11.1.5 Password Security

The ViewStation's password is not displayed on any screen. It must be entered before it is changed, either locally or remotely. The password can be encrypted if it needs to be transmitted over the network; the ViewStation will read and decrypt the password before it does its password verification.

If the system administrator sets a password in the configuration file and then sets that file to be saved in NVM, the user cannot change the password unless he knows it and has access to Setup Mode and the configuration file. The configuration password and the NVM (Setup Mode) password must always match. If the configuration password is loaded first, it is used as the current Setup Mode password; if the NVM password is loaded first, the configuration file (with all its settings) is discarded if its password doesn't match the NVM password. Since many Setup Mode fields, such as fontpaths, IP addresses, privileged Console window commands, etc. are protected by this Setup Mode password, important ViewStation settings are secure.

Handling the password on the network system needs some care.

1.11.1.6 Passwords in Bootp

If you use Bootp with your first time startup to load the configuration file, you must use TFTP as the file transfer protocol (or anonymous FTP with its username and password to load the file), and since TFTP requires a world-readable file, the password in the configuration file must be encrypted to preserve its integrity. The password in the bootp table itself can appear in clear text since this file belongs to the root user and can be hidden or read-protected. In this way, the level of system security of the root user is extended to each ViewStation.

If you use the ViewStation to load the configuration file directly , you can use the FTP or NFS protocols for the file transfer. You can also read-protect the file, so you can use the password in clear text.

1.11.1.7 Password Encryption with hdscrypt

The ViewStation has an decryption routine built into its server (based on a public-domain routine called "crypt" which is included with MicroEMACS written and copyrighted by Dana L. Hoggett; this program, renamed "hdscrypt" to avoid confusion with other encryption programs, can be obtained directly from HDS). This encryption program allows you to encrypt your password and then have the ViewStation decrypt it for comparison with the Setup Mode password(s) as needed.

To use Bootp to load a configuration file at initial startup, you must use TFTP (or anonymous FTP with a username and password). The configuration file itself must have read permission for TFTP (and the rest of the world) in order to be moved. If the Setup Mode password is used in the configuration file, it must be encrypted to preserve its integrity.

Using the hdscrypt program is simple. When you run the hdscrypt program, it requests a "key", that is, your user password, which serves as the basis for the encryption scheme. In this example:

hdssun12: hdscrypt

Enter Key **`rembrandt`**

`rembrandt`

Encrypts to: [IXWnI‘&}z]

the Key entered is "rembrandt", which is the user's password. You then re-enter the password and it is encrypted against this Key. The encrypted password is then displayed in brackets. You can encrypt other passwords (such as the FTP password) using this same key, but note that all these passwords must be used in the same configuration file, since the decryption is done against the same key.

This password (key) is also entered in the T128 field in the bootptab file as plain text. The encrypted password "IXWnl'&}z" is the entry you make in the configuration file.

When the configuration file is loaded into the ViewStation, the ViewStation server code (which also runs the hdscrypt program) examines the entry and decrypts it into "rembrandt", which it then compares to the password entered by the user in

its Setup Mode or its NVM. The NVM password and the bootptab passwords must be the same, and only the encrypted password appears in the configuration file. Since the bootp protocol and bootptab are controlled exclusively by the superuser, they may be protected or hidden.

Note that special characters, like spaces, can be used in encrypted passwords if they are enclosed in double quotes "". This ViewStation feature is present in boot prom versions 3.0 and higher.

With HDSware 3.0.3 and higher (you must use both boot prom code and server code at the V3.0.3 level), the ViewStation accepts any password in the configuration file as either plain text or encrypted text.

1.11.2 ViewStation rhosts Security

The ViewStation uses an internal rhosts file to configure the rsh and rlogin daemons and control access to the ViewStation. By default, this security is not used. This feature is available with HDSware version 3.2 and later.

The ViewStation's rhost security is set up by entering an rhosts file name in Setup Mode on the Remote Hosts menu or with the resource:

```
HDSsetup.rhosts.file: <filename>
```

in the configuration file. If no file is specified or the file is missing, rhosts security is not used. If the file is present but empty, no access to the ViewStation is allowed.

1.11.2.1 Setting Up an rhosts File on the ViewStation

An rhosts file is specific for an individual ViewStation. It can have any name or be in any location, but typical filenames might include the ViewStation name, such as "gregVS.rhosts" and might be located in the HDSware configuration directory (*/HDSware/hds-fx/config/*). These files should be writeable by root only.

The file contains records in this format (with white space separation):

remote_host_name remote_user_name

Each record should be on a separate line. The record gives permission for a host and user to use rsh and rlogin for communication to the ViewStation. The remote_host field accepts a DNS name, an IP address, a list of hosts (comma separated), or a "*" wildcard for all hosts. If the DNS name does not include an "." in its entry, the DNS suffix will be appended to the name when it is added to the database. The remote_user field accepts any valid username or "*" for any user.

1.11.2.2 ViewStation Remote Access Reporting

The ViewStation reports each host and user access attempting to connect to it with messages in its Console Window. Some sample messages are:

```
rshd_io: accepted rsh by gregh@hdssun1.hds.com
rshd_io: accepted rlogin by gregh@hdssun1.hds.com
rshd_io: rejected rsh by gregh@hdssun5.hds.com
rshd_io: rejected rlogin by gregh@hdssun5.hds.com
```

1.11.3 Permission for Remote Shell Commands

The process of starting the local window manager from a remote host uses the **rsh** (or **rcmd** on SCO hosts or **remsh** on HP hosts) command. This process establishes privileges for the ViewStation to use the remote host's permissions to start other remote processes on other network devices. These permissions are governed by the *hosts.equiv* and *.rhosts* files, discussed below.

With these things in place, the ViewStation is a trusted machine, that is, if a remote shell command comes from a particular host to the ViewStation, it will permit remote shell processes only back to that host or hosts specified in the permission files. The initial user logon from the ViewStation to the host sets the username, password, environment, and permissions. These permissions are extended to the ViewStation by the remote shell process, and any further extension of those privileges is governed by the *hosts.equiv* file. In this way system security is preserved at the same level for the ViewStation as an X terminal and any host or workstation on the network. The ViewStation is no more vulnerable (or any less vulnerable) than any other device on the network.

1.11.3.1 Using the */etc/hosts.equiv* File

The most common security permission is set in the */etc/hosts.equiv* file, which is a simple list of hostnames which have permissions equivalent to the host. Generally, you must add the ViewStation's hostname to this file on the remote host you will be using.

Syntax for this *hosts.equiv* file varies with different operating systems. Adding names to this file is not done automatically (some operating systems, like Sun OS, do permit the addition of a plus sign "+" to this file, which indicates permission for automatic addition of host names), but in general, you must explicitly add the ViewStation's name to this file. The following examples are suggestive of the contents of this file. Check your own man pages for correct syntax on your system.

```
# hosts.equiv syntax examples
#
# host1 userA              this allows userA access to host1
# + host1                  this allows all users access to host1
# host1 -userA             this denies userA access to host1
# - host1                  this denies all users access to host1
# -@groupA                 this denies groupA users access to hosts
# +@groupA  +@groupB       this allows groupB users access to
                           groupA hosts
```

1.11.3.2 Using the .rhosts File on the Host

Another location that may have permissions set, or prohibited, is the *.rhosts* file in the user's home directory. This file allows the user to specify permissions and

equivalency of his user name for other hosts. You may also have to modify this file. The syntax for this file is similar but varies with different operating systems.

The ViewStation's individual rhosts file is discussed in Section 1.11.2.

1.11.4 Host Access Control Security

The host-based security described above can also be extended to a user-based security procedure, called MIT-MAGIC-COOKIE-1, which is supported by the HDS ViewStation. This security procedure extends the security of the user's account and password to all clients on the system. The procedure is simple and almost invisible to the user.

Note that for some systems (such as OpenWindows), this security is on by default.

1.11.4.1 Host Access Control from Setup Mode

A new resource has been added to permit the ViewStation to use Access Control security when you are not using Xdmcp. It uses the first remote client to connect as its Access Control.

The resource is in the config.syntax.fx file. The default is disabled.

```
HDSsetup.default.AccessControl: [ DISABLED | ENABLED ]
```

1.11.4.2 Using Magic Cookie Security

When a session is started with Xdm, a special code (the Magic Cookie) is written to the .Xauthority file in the user's home directory. This .Xauthority file has restricted permissions (-rw-------) so only the original user can read it. Before any new clients can start, they must read this file to obtain the Magic Cookie to be able to start their process. If they aren't the original user, they can't read the file and access is denied. The key to Magic Cookie security is the integrity of the user's account and password. If anyone can read the user's .Xauthority file, then this security mechanism is useless.

1.11.4.3 Starting Magic Cookie Security in Xdm

This security is started in the Xdm configuration file, xdm-config, with two resources:

DisplayManager._0.authorize: true

DisplayManager*authorize: true

The first resource turns on authorization for the local display and the second turns it on for for all other displays. You can disable authorization by setting these to "false" and restarting Xdm. This is the proper way to disable authorization for your Xdm process.

There is also a way to disable it by allowing authorization for all users. This is discussed in the sections below. More details on Xdm operation and setup are found in Section 3.3.2.

1.11.4.4 Magic Cookie Security on the ViewStation

The HDS ViewStation controls its response to Magic Cookie requests by the "Host Access Control" field in the XDMCP menu of Setup Mode. This permits you to enable or disable this security feature on the ViewStation. Note that this is turned ON in its default configuration.

1.11.4.5 Problems with Magic Cookie Authentication

Sometimes a problem arises with Magic Cookie authentication. Its symptom is that the user can log on with Xdm and use processes on the Xdm host but cannot start processes on other hosts. There are two difficulties observed.

- Authentication is On by default with R5; users may not be aware of the authentication process. Typically, the problem doesn't show itself until the user tries to log onto a host other than the Xdm host and cannot do so.
- A faulty "Magic Cookie" is created by some Xdm clients

In both of these cases, there are ways to disable or to extend the authentication so your system can be accessible to all users. These are important considerations and need careful security control.

Extending User Authentication

Most systems have an MIT resource for the login client that permits disabling the access control. This resource is available in both R4 and R5 versions. In your */usr/lib/X11/xdm/Xresources* file, set the resource:

```
xlogin*allowAccess: true
```

This allows access to the session for all users, thus effectively disabling Magic Cookie authentication.

You can also set this resource with the key sequence *Ctrl*-+ (some systems have not implemented this key translation to set the allowAccess resource from the keyboard). You can use the *Ctrl*-+ key sequence at any time, including when the Xdm login window is present. When you press the *Ctrl*-+ sequence, you will notice the login banner change to "This is an unsecure session", which means that the authentication feature is disabled. Note that *Ctrl*-+ toggles the allowAccess resource, so if it was initially Off, this can turn it On. In this way, the user can select the authentication he wants for the system.

Extending Magic Cookie Authentication

We have observed that some versions of Xdm (a host-based client) do not work correctly and create an invalid Magic-Cookie. This means that the user can only gain access to the original Xdm host. Other hosts will have access denied and will report authentication and permission errors. This condition arises because the host's Xdm process generates an incorrect Cookie (typically all F's).

If you have this problem, you can explicitly extend authentication to one or more hosts. This is the intended method of operation, so user security is controlled by the Magic Cookie. In a practical sense, extending authentication to all hosts is the same as disabling the authentication.

1.11.4.6 Using the ViewStation's xhost Client

If you have an R5 Xdm client with an Xsetup file, you can solve authentication problems by using the **xhost** client, a local process running on the HDS ViewStation. This solution has the HDS ViewStation grant access to other hosts by means of the **xhost** program. Because the ViewStation runs this **xhost** program at Xdm startup, it sets this permission as Xdm starts up.

In the Xdm directory on your host machine, edit the Xsetup file (on a Sun system - */usr/openwin/lib/xdm/Xsetup*; on an HP system - */usr/lib/X11/vue/vuelogin* / *Xsetup*; on SCO systems - */usr/lib/X11/scologin* / *Xsetup*; etc.). This Xsetup file may not exist and you may have to create it. If it is not present, you must also enter this Xsetup file name in the xdm configuration file, xdm-config. Refer to Section 3.3.2.

This sample script checks for HDS ViewStations among displays requesting Xdm management, then sets "xhost +", which extends access permission to all hosts.

```
vendr=`xdpyinfo | grep Vendor | grep HDS`
if [ -n "$vendr" ]
then
# this assumes that the ViewStation display is registered in
# the /etc/hosts file or with NIS on the host
                disp=`echo $DISPLAY | sed 's/:/ /' | awk '{
print $1 }'`
                rsh $disp xhost +
fi
```

Be careful with the syntax in this file; make adjustments appropriate for your host machine (such as "remsh" for HP hosts).

The ViewStation's initial login to the Xdm host (which creates the Cookie authorization) then has permissions extended to all other hosts.

1.11.5 CMW Security

The HDS ViewStation supports Compartmentalized Workstation (CMW) security, in which applications and processes are controlled by a "Trusted Client", typically a window manager. This trusted client maintains labels containing permission levels for all windows, X atoms, and pieces of data for all clients and servers. These labels are examined with each transaction and a detailed audit trail is maintained. This CMW security meets U.S. Government B1 level security, using the NSA rating procedure.

1.11.5.1 CMW Security Features

This section summarizes the compartmentalized workstation secure window interface for the HDS ViewStation terminal. This includes modifications to the server software based on X11R4. These comprise the TCB (Trusted Computing Base). There are two main features of CMW security:

- Trusted X server, including all server operations, with the Motif window manager.
- Trusted Motif window manager, including label display and trusted paths.

Four specific security categories are addressed:

1) Associating labels with all window system objects. This includes a set of client idenfications and labels, as well as labels for Atom Names, Properties, Colormaps, Pixmaps, and Drawables. For any Drawable (a window or a pixmap), the trusted X server uses three labels:

Sensitivity label (normally set equal to the parenting client sensitivity, but modifiable)

Information label (dynamically modified with window operations)

Input information label (which determines the acceptable input events sent to the window)

2) Access control and operating restrictions for trusted clients. There are two specific controls applied;

Certain window objects are accessible only to their single creating client

Shared objects are controlled by explicit, mandatory control checking.

3) Protocol extensions and modifications for the trusted window manager. The window manager operations are modified in five ways:

Cut/paste operations are governed by authentication clients; data transferred is label verified.

Authentication procedures are given to the trusted window manager, including starting new sessions.

Support for window-based trusted paths, including reserved screen areas.

Support for label modification (clear or reset operations) processes.

Support for inactive session or timeout processes from a configurable logoff event.

4) Auditing events and security concerns. The trusted X server and trusted window manager create and maintain audit records for these events:

Creation and destruction of window system objects

Explicit changes to window labels

Implicit changes to information labels resulting from floats

Access denials resulting from access control checking

Access denials resulting from lack of privilege

Client use of privilege

Server reinitializations

HDS ViewStation: Floppy Disk Drive

This chapter discusses connecting and using a 3 1/2" floppy disk drive with the HDS ViewStation.

Section 1.0 - HDS ViewStation

1.12 ViewStation Floppy Drive Feature

With HDSware release Version 3.1, a 3.5" external floppy disk drive can be used with HDS ViewStation models. The HDS floppy disk drive supports reading, writing, and formatting high density (1.44MB) and double density (720KB) MS-DOS disk formats. Only one floppy disk drive is supported at one time.

Note that older ViewStations must have ECO 375A installed in order to connect a floppy disk drive.

1.12.1 Hardware configuration

The HDS floppy disk drive comes with three separate parts; a floppy drive base unit, a parallel port cable, and an external wall mount power supply. Use these steps to connect the floppy disk drive to the ViewStation:

1. Connect the parallel cable's male connector to the parallel port on the View-Station's backpanel.

2. Connect the other end of the parallel cable to the floppy drive base unit's male connector.

3. Plug the power supply's connector into the side of the floppy drive base unit.

4. Plug the power supply into an electrical outlet. The green LED next to the power connector should light on the floppy drive base unit.

1.12.2 Using the Floppy Drive with a Printer

A parallel port printer can still be used with the floppy disk drive present. Connect the printer's parallel port cable to the floppy base unit's female connector (in a daisy chain).

Printing from the ViewStation works the same way as when the printer was directly connected to the ViewStation. The printer is accessed via device */dev/pp0*. You may notice that the data transfer rate to the printer is reduced when both the floppy drive and the printer are simultaneously in use, since the parallel port is shared between the printer and floppy drive.

1.12.3 Software configuration

The ViewStation's floppy disk drive is auto-detected during startup; there is no ViewStation configuration needed for it. Further, the floppy disk's filesystem is automatically mounted and exported.

The Console window command "hwconfig" shows the presence of the floppy drive:

```
Parallel port device 1: 1.44MB Floppy Disk Drive
```

The floppy device driver is installed as */dev/fd1* in the ViewStation's local filesystem. This floppy device driver file name is used when formatting a disk. Formatting disks is discussed in a later section.

1.12.4 Changing Floppy Disks

Floppy disks can be changed whenever the floppy disk drive light is off. When a new floppy disk is inserted, its density (high or double) is automatically determined. If the floppy disk is not formatted, the Console message log will contain the following message:

```
HDS> Unformatted floppy disk?
```

When NFS is used to access the floppy and the floppy disk is changed, stale file errors may occur. This is a normal condition that occurs as a result of the NFS client referencing files that no longer exist. If the current working directory was within the floppy filesystem, change it to be the floppy mount point (e.g. '*cd /tmp/vsfloppy*' if the floppy was mounted using the example below).

1.12.5 Floppy Disk Drive Access

You can read or write to the floppy disk drive locally using shell commands in the Console window, or remotely from a host computer using **rcp** (remote copy), or by using NFS. The NFS method is most useful, and is set by default.

To access the floppy files remotely using NFS, the ViewStation's clients configuration database requires an entry for **nfsd**, the NFS server daemon. The *clients.config* file distributed with HDSware Version 3.1 contains the necessary **nfsd** entry, which is:

```
 hds.osExtension.nfsd.path: /FXUtilities/hds-fx/extensions/
nfsd
```

Note:

This nfsd entry in the clients.config file is On by default.

The ViewStation boot code does not support the floppy disk, so it cannot be used for booting.

1.12.6 Local Floppy Filesystem Access

Files on the floppy appear in the */floppy1* directory in the ViewStation's local filesystem. The ViewStation's **ls**, **cat**, and **more** shell commands can be used to examine floppy files in the Console window. For example:

```
HDS> cd /floppy1

/floppy1

HDS> ls

total 132

1 test* 131 termcap*

HDS> cat test

This is a small file on the HDS floppy.

HDS>
```

1.12.7 Floppy Drive RCP (Remote Copy) Access

The ViewStation accepts remote copy requests to its floppy drive issued by remote hosts. The remote copy command does not work from the HDS shell.

The **rcp** remote copy command supports the standard Unix filename wildcard matching for '*' and '?'. Metacharacters intended to be interpreted on the ViewStation should be quoted using \, ", or '.

This is an **rcp** example, issued from a workstation (hdssun1), to copy floppy files starting with the letter 't' from ViewStation named "mshvs":

```
hdssun1> rcp mshvs:/floppy1/t\* .
```

1.12.8 Floppy Drive NFS (Network File Service) Access

To access the ViewStation's floppy files remotely using NFS, you must mount the ViewStation's */floppy1* directory using the '**mount**' command.

The following command mounts the floppy drive attached to the ViewStation called "mshvs" to a file location called */tmp/vsfloppy*: (check the syntax on your NFS system)

```
mount -o rw,intr,wsize=6144,timeo=1200 mshvs:/floppy1 /tmp/
vsfloppy
```

Note:

1. The */tmp/vsfloppy* directory must exist before issuing the **mount** command.

2. The timeout argument, timeo=1200, is important since the floppy drive is a slow I/O device. With the timeout argument, the NFS client waits longer

before resending requests, which allows the floppy drive time to complete the requests.

3. The write size argument, wsize=6144, is important since it reduces the size of the NFS write requests sent to the ViewStation. If too many large requests are sent too quickly, the ViewStation will lose the requests. The NFS client may think the NFS server is having problems and display a message like:

```
NFS server <ViewStation> not responding
```

There will be a noticeable pause before the NFS client retries, at which time you may see a message like:

```
NFS server <ViewStation> ok
```

4. The floppy filesystem should be unmounted before powering down the ViewStation, such as:

umount /tmp/vsfloppy

Otherwise, processes on the workstation may get "stuck" when performing operations that want access to the floppy filesystem mount, such as 'df'.

1.12.9 Floppy Drive File Ownership

Since MS-DOS filesystems do not store file owner or permissions mode information, the ViewStation assigns the same user ID, group ID, and permissions mode for all floppy files.

By default, floppy files are exported with uid=65534, gid=65534, and permissions mode 0777 (read, write, execute permissions for user, group, and others). Note that the mode, 0777, is interpreted as an octal number since it begins with a zero. Files that are read-only report their permissions mode without write permissions. If the floppy disk is write protected, all files will report their permissions mode without write permissions.

The default mount point, file owner, and permissions mode can be changed by adding a mount command to the ViewStation's Startup command file. The default mount command is:

```
mount -t rmfs -o synchro-
nous,exported,uid=65534,gid=65534,mode=0777 /dev/fd1 /
floppy1
```

1.12.10 Floppy Drive Formatting disks

The **fdformat** utility is provided as a ViewStation Console window shell command to format disks. This is the **fdformat** manual page:

NAME

fdformat - format floppy diskettes

SYNOPSIS

fdformat [-l] [-n] [-v label] device

DESCRIPTION

fdformat is a utility for formatting floppy diskettes. All new blank diskettes must be formatted before they can be used. fdformat formats and verifies each track on the diskette, and terminates if it finds any bad sectors. All existing data on the diskette, if any, is destroyed by formatting.

By default, fdformat formats a high density diskette with a capacity of 1.44 megabytes. Use the -l option to format low density diskettes (720K capacity). Note: it is not possible to put a low density format onto a high density floppy diskette. High density diskettes can be recognized by the high density detect hole in the lower right corner of the diskette, opposite the write protect hole in the lower left corner.

This installs a MS-DOS file system on the diskette after the format is done.

OPTIONS

-v Put a MS-DOS label on the disk after formatting it.

-l Format a low density (720K) diskette.

-n Skip verifying the floppy diskette after formatting.

NOTES

The MS-DOS file system installed by fdformat does not have the necessary system files to make it bootable on a PC.

FILES

/dev/fd1

/dev/fd2

etc.

BUGS

Bad sector mapping is not currently supported on floppy diskettes. Therefore, a diskette is unusable if fdformat finds an error (bad sector).

This command formats a high density disk with a volume label of 'TEST':

```
HDS> fdformat -v test /dev/fd1
```

1.12.11 Using the ViewStation's Floppy Disk with SoftWindows from Insignia Solutions

Application programs like SoftWindows from Insignia Solutions run on your host computer and can read/write MS-DOS disks using a DOS utility, FSA, File Sharing Architecture. These application programs reference the UNIX file system as if it were a DOS file system. For example, a typical application would set DOS drive E: as a UNIX directory.

Follow these steps to use the ViewStation's floppy disk drive from SoftWindows:

1. Make sure your ViewStation's fontpath includes the SoftWindows fonts. If you need to add these fonts, use this command line from an xterm window on your HDSware host:

```
xset +fp /FXUtilities/hds-fx/fonts/softpc
```

where FXUtilities is the host IP address and path to the HDSware software. You can verify this setting with the "xset q" command, which reports the current font-path.

2. Make sure your username environment variables include the SoftWindows environment. You should have an entry in your .profile file for the SoftWindows environment:

```
setenv $SWINHOME /usr/lib/SoftWindows
```

or the directory appropriate to your system. If this variable is not set, SoftWindows will use a pop-up box and offer to set it for you. SoftWindows also checks for its system configuration file and will set it automatically for you if you wish.

If you use Xdm, the .profile file may not be read. In this case, you should add a line to your .xsession or .xinitrc file to read the .profile file.

3. Mount the ViewStation's floppy disk on your host computer's file system with NFS. You must be the root user to do this.

For example, you could mount the floppy disk drive on a ViewStation named **mikevs** at file location */export/home/mike/mkfloppy* with this command:

```
mount -o rw,intr,wsize=6144,timeo=1200 mikevs:/floppy1
/export/home/mike/mkfloppy
```

4. Set up your FSA (File Sharing Architecture, an MS-DOS utility) link to the floppy disk drive. From your SoftWindows Desktop, choose **Open Drive ...** from the **Disk Drives ...** cascade menu on the **Options** menu.

This menu lets you define a UNIX directory as your PC Drive E:

Click on the **FSA Root Directory Name ...** and then enter the mounted file location for the floppy disk drive: (for this example)

```
/export/home/mike/mkfloppy
```

Alternately, you could do Step 4 from your MS-DOS configuration, as below.

4a. From the MS-DOS prompt, you can use the NET USE command:

```
DOS:\> NET USE X:dir
```

or, in this example:

```
DOS:\> NET USE E:/export/home/mike/mkfloppy
```

You may want to add this line to your AUTOEXEC.BAT script to make this configuration automatically when you start your PC.

5. At this point, you're finished. You can read and write to the ViewStation's floppy disk as if it were your PC drive E:. For example, a file BUDGET.DOC on the floppy disk would appear to the PC application as E:BUDGET.DOC

1.12.12 Using the ViewStation's Floppy Disk from WABI 2.0

To use the ViewStation's floppy disk drive on a WABI 2.0 system, follow these steps:

1. Mount the ViewStation's floppy disk on your host computer's file system with NFS. For example, you could mount the floppy disk drive on a ViewStation named **mikevs** at file location */export/home/mike/mkfloppy* with this command:

```
mount -o rw,intr,wsize=6144,timeo=1200 mikevs:/floppy1
/export/home/mike/mkfloppy
```

2. From within your WABI window:

a. In the Program Manager, open the Configuration Manager.

b. Now open the Drives icon.

c. Select an unassigned drive letter in the Drives panel, for this example choose Drive E.

d. In the Available Directory Paths panel, navigate through the list of directories and files to reach the file system you want:

```
/export/home/mike/mkfloppy
```

in this example.

e. Choose Connect. The path displayed is assigned to the floppy disk drive, and appears next to the drive letter in the Drives panel.

f. Choose OK to save the assignment.

3. At this point, the ViewStation's floppy disk drive is selectable from within the File Manager as Drive E.

HDS ViewStation Client Operations

This chapter discusses the operation of the HDS ViewStation's local clients, called HDSperx.

2.0 Section 2.0 - HDS ViewStation Local Clients

2.1 HDS ViewStation Client Operations

2.1.1 Using the HDSperx Clients

The HDS ViewStation has a powerful RISC processor and an internal Unix architecture that permits the ViewStation X terminal to run a number of important clients on its own display. These clients include the window managers, HDSterm, Setup Mode, Console window, and others.

In addition to these clients, the ViewStation can run other clients, such as a calculators and clocks. These are normal X clients that you may commonly run on your host for display on your X terminal. The HDS ViewStation lets you run these clients locally. This gives you the benefits of easy setup and fast response as well as the savings in both host CPU time and host memory, and the reduction of network traffic by having them run on the X terminal.

These HDSperx clients are included on the HDSware tape and are automatically accessible to the ViewStation when you use the HDSware start up feature. The HDSperx clients are available from the root menu of your window manager. All of this setup and configuration is automatic. The details are discussed below.

2.1.2 HDSperx Clients

There are a number of HDSperx client application programs. These are listed below with a brief description of their operation and features. They appear automatically on the root menu of your local window manager when you use the HDSware start up feature.

You can add or delete HDSperx clients from the HDSperx menu by adding or deleting them from the *./hds-fx/config/clients.config* file. This process is discussed in detail below. For instance, you may choose not to load the tetris and maze clients, or you may want to show only one clock instead of three.

You can also modify the client attributes and extend password protection to any or all of these clients, as discussed below.

2.1.2.1 HDSperx Clients Menu

This list shows the clients normally included in the HDSperx menu. This list is subject to change.

HDS audio - a client to control the ViewStation's audio feature

HDS calc - a TI-30 style calculator with all the normal arithmetic and trigonometric functions

HDS calc (HP style) - a HP 10C style calculator using Reverse Polish Notation

HDS clock (analog) - an analog clock with a square face

HDS clock (digital) - a digital clock

HDS clock (transparent) - an analog clock with a round face

HDS console - a Console shell window for the ViewStation

HDS memory - the Available Memory client to report ViewStation memory status

HDS saver - a screen saver and screen lock, with selectable programs for the screen display

HDS serial - an HDSterm window for serial connections

HDS telnet - an HDSterm window for Telnet connections

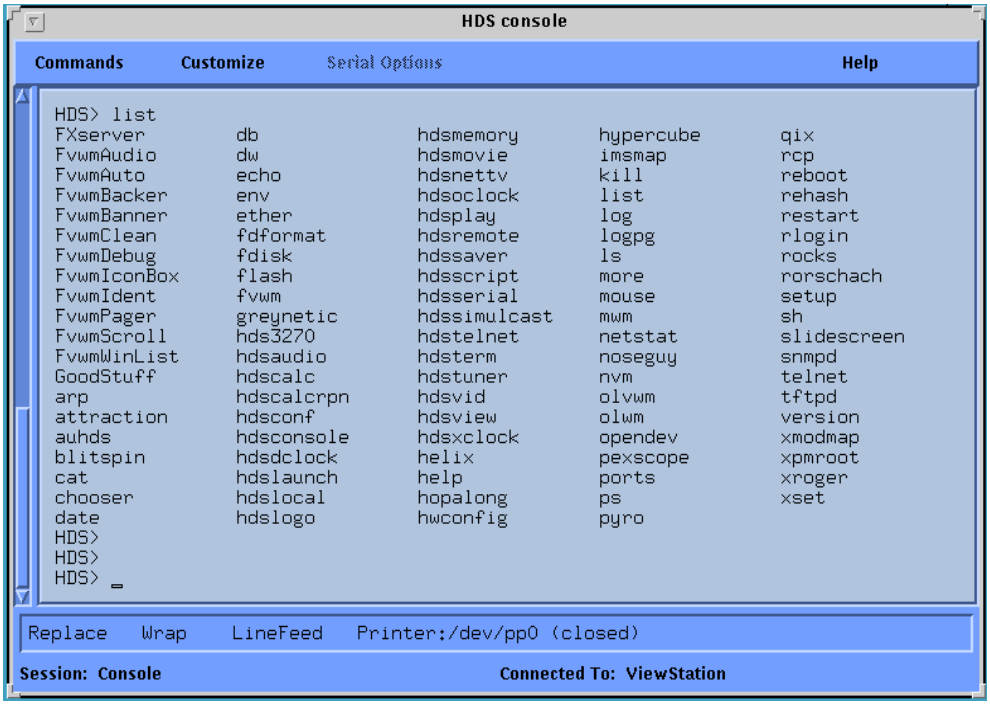
HDS video - a client to control the ViewStation's video feature

HDS view - a client to display Postscript files

Clients with the "menuName" attribute in the clients.config file appear on the HDSperx menu. Names commented out of the clients.config file, such as tetris or maze, do not appear on the menu or on the list command.

2.1.2.2 HDSperx Client List

There are many more HDSperx clients, including some clients for Console Window operations. This list is available with the "list" command in the Console Window. It is subject to change.



2.1.3 HDSware Installation of HDSperx

The HDSware installation feature of the ViewStation is a simple way to utilize the HDSperx clients and menus. When you use HDSware to get started (specifying the HDSware host and path), the ViewStation automatically locates the *./hds-fx/config/clients.config* file and the HDSperx clients themselves in the HDSware directory, *./hds-fx/bin*, and automatically adds them to the window manager's menu file so they are easily accessible from a submenu on your local window manager's root menu. You must restart the ViewStation after you make the HDSware entries so the configuration file will be read; you probably also want to save these values in NVM.

2.1.3.1 Using the HDSperx Clients

When you use your window manager's root menu (move the pointer to the root window and hold down the left mouse button), you will see HDSperx as its first menu entry. When you move to this entry, another menu appears that contains a list of the HDSperx clients. Select the client you want and release the button. That's all it takes. The new HDSperx client starts and you can use it by moving the pointer to it.

2.1.4 Customizing HDSperx Clients

The HDSperx client configuration file automatically lists the available downloadable clients and their attributes. If you set up the ViewStation with the HDSware configuration feature, you do not have to be concerned with these files, unless you want some form of customization for these clients.

The list of clients.config file can be set in the HDS Setup client, under the "Setup Options" menu and the "Startup Options" submenu. This list can also be set with the "HDSsetup.client.config.path" entry in the main ViewStation configuration file, config.fx.

You can add or delete clients from this list (and from the root menu list) by adding or deleting clients (and the X11 extensions provided as downloadable clients) from this file. The easiest way to do this is simply insert a "#" comment character at the beginning of the "menuName" resource line for that client so it is ignored when the file is read.

2.1.4.1 Format of Client Configuration Files

The format of the clients.config file is based on the format used for standard X resource databases. However, they are NOT X resources in the normal sense; they are used only by the ViewStation's clients.config mechanism. These entries should not be placed in normal X resource files or set to be read by **xrdb** since they would have no effect. Similarly, your Xresources should not appear in the clients.config file.

Entries in the clients.config files are self-explanatory and appear in the form:

hds.client.<client_name>.<attribute_name>: <attribute_value>

Comment lines should begin with a '#' or '!' character. Note also that C-preprocessor style comments will not work in these files.

Refer to the *./hds-fx/config/clients.syntax.config* file for samples and more details on client attributes.

2.1.4.2 A clients.config Sample File

This set of entries describes the clients ico and HDS xclock. The first entry determines the name that will be used automatically for the clients in the local window manager menus. The second entry specifies the full remote host path to the client's downloadable code. The third entry specifies the default arguments which will be used by the client if no other arguments are specified. Other resources are described below.

A section of the clients.config file is shown below:

```
#
#   ico
hds.client.ico.menuName:  Ico
hds.client.ico.path: /FXUtilities/hds-fx/bin/ico
```

```
hds.client.ico.defaultArgs:  -p8
#
#   xclock
#
hds.client.hdsxclock.menuName: HDS Clock (Analog)
hds.client.hdsxclock.path: /FXUtilities/hds-fx/bin/hdsxclock
hds.client.hdsxclock.defaultArgs: -name HDSxclock -u 1
```

and so on. You can look at the complete file in /FXUtilities/hds-fx/config/clients.config.

2.1.4.3 Pre-loading Motif Libraries

The HDSperx clients, as well as HDSterm, use the Motif libraries Xm and Xt in their operation. These libraries can be pre-loaded when the ViewStation starts up so the clients are ready when you select them.

There are resources in the *./hds-fx/config/clients.syntax.config* file to control this pre-loading feature:

```
hds.library.motif.preload: yes (default value is NO)
```

In addition, individual clients have options to control their own operation and code utilization:

```
hds.client.<name>.shareCode: [ no | yes ]
hds.client.<name>.shareFree: [ no | yes ]
hds.library.<name>.shareFree: [ no | yes ]
```

These resources indicate whether the client named should share its code with other instances of the same client, or whether a separate copy of the code should be loaded. They can also specify whether the code should be freed when the last instance of the client exits. The same free characteristic applies to the libraries loaded for the clients.

2.1.5 HDSperx Client Attributes

There are a number of Client Attributes that you may want to modify. The attributes which can be set in this file include:

critical

This optional attribute specifies that the client is critical to the operation of the ViewStation and should not be suspended during low-memory conditions. This attribute should be used only when absolutely necessary.

Default value: false

Example:

hds.client.mwm.critical: true

defaultArgs

This optional attribute specifies the default arguments to use when starting the client, if no other arguments are specified. Arguments for each client are different and are described in the man pages for each client.

Default value: none.

Example:

hds.client.hdsxclock.defaultArgs: -bg red -name HDSxclock

defaultEnv

This optional attribute specifies the default environment to use when staring the client. Environment variables specified in this way are overridden by those set directly by users. The value consists of a list of variable specifications separated by the symbol \n. Each variable name may be preceded by whitespace.

Default value: none.

Example:

```
hds.client.hdsxclock.defaultEnv: XFILESEARCHPATH=/HDS$$ROM\n\
                                 XBMLANGPATH=/HDS$$ROM/%B
```

menuName

This optional attribute specifies the menu entry to be used for the client in the automatic local window manager menu. If there is no value given for this attribute, then the client will not show up in the window manager menu.

Default value: none.

Example:

hds.client.hdstelnet.menuName: HDS Telnet

minAvailMem

This optional attribute specifies the minimum amount of available ViewStation memory usually required by the client. If the ViewStation does not have enough memory available, then the client cannot be started locally.

The value is measured in bytes. The suffixes "kb", "mb", "k", and "m" are supported for convenience. The case of these suffixes is not important.

Default value: none.

Example:

hds.client.hdsxclock.minAvailMem: 250kb

passwdRequired

If "Yes", this optional attribute indicates that the client should only be run if the ViewStation user has been able to present the ViewStation's Configuration Password before attempting to start the client. The password is entered by using the 'password' command in the HDS Console.

Default value: No

Almost all Console shell commands can be password protected, according to the following list.

You must set passwdRequired for these commands:

```
arp          ether        help
list         log          logpg
ls           memory       netstat
nvm          opendev      ports
ps           pwd          rehash
             version
```

The passwdRequired resource is set by default for these commands:

```
db           dw           flash
reboot       restart      kill
```

These commands cannot be used with the passwdRequired resource since they are integral parts of the shell itself:

```
cd           echo         env
exit         password     setenv
unsetenv
```

You can set this resource just like with any other HDSperx client, such as:

```
hds.client.list.passwdRequired: true
hds.client.ps.passwdRequired: true
hds.client.netstat.passwdRequired: true
hds.client.opendev.passwdRequired: true
```

Another way of extending password protection to local clients is to use the configuration file resource:

```
HDSsetup.password_is_global: yes
```

which protects all Setup Mode entries and all local clients.

shareCode

This optional attribute specifies that the client's executable code should be shared among multiple instances of the client. Each instance of the client has its own stack and set of variables, but the executable code is shared.

Default value: no.

Example:

hds.client.mwm.shareCode: yes

shareFree

This optional attribute applies to clients that have the shareCode attribute set to yes. This attribute determines whether or not the client should be freed when the last instance of the client exits. When this attribute is set to yes, the client will be freed when the last client instance exits. When this attribute is set to no, the client will remain in memory and will not be freed.

Default value: no.

Example:

hds.client.mwm.shareFree: yes

preload

This optional attribute applies only to libraries and indicates the library should be loaded at startup. Libraries are normally loaded when a client is run that uses them. The "preload" attribute will be ignored if the library has the "shareFree" attribute set to yes or if the library is missing or has an empty "path" attribute.

Default value: no.

Example:

hds.library.motif.preload: yes

path

The "path" attribute is REQUIRED for a downloadable client description. This attribute lists the full path which the ViewStation can use to access the executable code for a client.

Example:

hds.client.xclock.path: /FXUtilities/hds-fx/bin/hdsxclock

Note: The same executable code can be referenced by several different client names. This allows users to rename or alias a single command to have multiple client names.

Example:

```
hds.client.xcalc.path: /FXUtilities/hds-fx/bin/xcalc

hds.client.xcalcrpn.path: /FXUtilities/hds-fx/bin/xcalc

hds.client.xcalcrpn.defaultArgs: -rpn
```

Other highly-specialized attributes are provided, but should NOT generally be useful.

forceLocal

This attribute controls whether the downloadable version of the client will be run in preference to the remote host version. It is only relevant when 'transparent-Mode' is active for the program which is launching the client, as it is for a local window manager. If 'forceLocal' is yes, then the local client will be run, if possi-

ble. If 'forceLocal' is false, then the Remote Shell protocol will be used to attempt to run the command remotely.

Usually forceLocal is 'yes', since the user usually would prefer to run a local version of a command to a remote one.

Default: yes

stackSize

This attribute allows the stack size for the client to be increased. (Note: The HDS Console 'ps' command shows the stack utilization for running clients.)

transparentMode

This attribute is intended to be used by clients (such as window managers) which are intended to be run locally but to act as though they were running on a remote host. It is only applicable when the client is started from a remote host via the Remote Shell protocol. In that case, if transparentMode is true, the local client will access the remote user's home environment and use the remote host's files as though the client were running on the remote host. Also, all commands started by the client will be run on the remote host (unless they are downloadable clients with the 'forceLocal' attribute set to 'yes').

If transparentMode is false, then the local client will use the ViewStation's filesystem (where filenames must be entered relative to local mount points). The local client would also not use the Remote Shell protocol to run commands on the remote host.

Default: no

2.1.6 X11 Server Extensions

Extensions Integrated into the HDSware Server

The HDS ViewStation server software contains a number of built-in X11 server extensions. These are software modules that allow the server to do special functions which are not necessary for the standard X11 server functionality, but operate as optional extensions. One example of an extension of this type is the Shape extension, which allows the server to draw a round window, such as the window used for the "oclock" client application.

The X utility program, xdpyinfo, shows what extensions are available for the active display. Check the HDSware version for the extensions currently supported.

2.1.6.1 Downloadable X11 Server Extensions

There are some additional extensions that are not built in but are available for the server to download as needed from the HDSware location on the network. There are currently four extensions of this type included on the HDSware tape, the XTrap extension, the Multi-Buffering extension, the shared memory extension, and the PEX 3-D drawing extension.

2.1.6.2 Loading X11 Server Extensions

These server extensions can be used by loading them with the clients.config file (in the default configuration, these clients are commented out of this file). You can include them in this file by removing the comment characters as you would one of the HDSperx clients and the extension will be loaded with the server code. Since they are built into the server code, they are actually loaded on reboot.

You should note that these extensions are installed as an integral part of the server code, not as independent clients. For this reason, you should always be careful to match the server code version with the extension files on the same HDSware tape. If you mix different file versions, they won't work. This is not generally a problem, since the extension files are included on the same tape; you just have to be careful not to mix tape sources.

2.1.6.3 XTrap Extension

The XTrap extension is used to test the operation of the X server and X software, typically with third party software testing application programs. It is often used with scripts that capture keypresses and mouse movements, or replays scripts to simulate sessions with user and application interaction. The XTest extension is part of the XTrap extension, though it is often installed separately. Refer to the X11 manual pages for a more complete specification.

2.1.6.4 Multi-Buffering Extension

The Multi-Buffering extension is used to enhance the display of graphics images. The displayed image is swapped for another buffered image which was drawn while it was off-screen, thus giving a fast replacement with the actual drawing invisible to the viewer. Refer to the X11 man pages for more detail.

2.1.6.5 Shared Memory Extension

The Shared Memory extension (MIT-SHM) is used to provide local downloaded clients with a mechanism to use high performance direct shared memory access for X pixmaps. This extension is loaded by default (refer to the clients.config file).

2.1.6.6 PEX-3D Extension

The PEX-3D extension is a 3D Protocol extension to the X Window System. It adds over 200 X protocol requests for defining and displaying 3D pictures.

Details on the PEX-3D Extension are found in Section 2.6.15.

HDSterm Terminal Emulation Client

This section describes HDSterm, the ViewStation's local terminal emulation client, including its properties and customizing options.

Section 2.0 - HDS ViewStation Local Clients

2.2 HDSterm Terminal Emulation Client

HDSterm is the ViewStation's terminal emulation client. It provides DEC VT320 terminal emulation in a graphical user interface window. HDSterm offers substantial enhancements of the VT320 terminal emulation, including local printer support and DEC color command support.

There are also other ViewStation terminal emulations discussed in Section 2.3.

2.2.1 Using the HDSterm Client

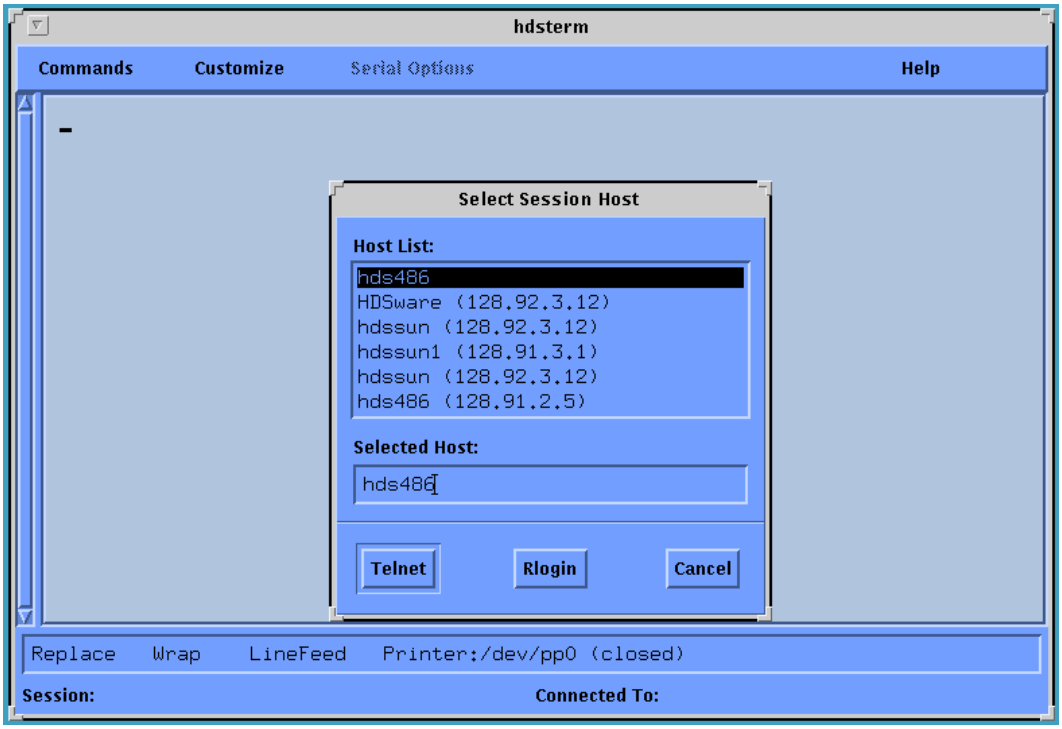
The HDS ViewStation client HDSterm serves as a Telnet window, a Serial Terminal window, and a Console window interface.

This terminal emulation gives you additional capability in the window, including VT320 capabilities such as Numeric and Application keypad functions, cursor keys for single line scrolling (and Shift-cursor keys for scrolling screen-by-screen), cursor selection, and so on. A complete list of these features appears later in this chapter. Many of the features can be selected by the pull-down menus, or are selectable with DEC VT320 command sequences, also listed later in this section.

The HDSterm window appears when you select either the Telnet Terminal Window or the Serial Terminal Window or the Console Window from the Setup Mode Main Menu. For Telnet, you have a dialog window offering you a choice of connecting with any of the hosts in your Remote hosts list, with the Default Telnet host highlighted for easy selection. For the Serial Terminal window, you are prompted for the serial port you want to use. The Console window appears with current output of diagnostic messages.

2.2.1.1 HDSterm Window

The HDSterm window with the Telnet selection menu is shown below.



The HDSterm pull-down menus contain options for operating and configuring the window. HDSterm is loaded automatically if you start the ViewStation with server code Vnn-D.hds.

2.2.1.2 Using the tty Telnet Client

If you have selected a server code version that doesn't load HDSterm, your Telnet selection will use the ViewStation's tty client with a limited ANSI emulation.

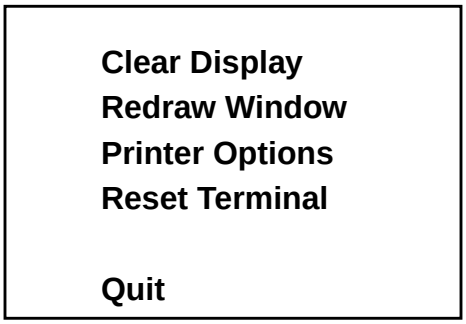
The tty Telnet client is loaded automatically if you start the ViewStation with server code Vnn.hds.

2.2.2 HDSterm Functions

The HDSterm window has a number of functions and settings which are available from pull-down menus.

2.2.2.1 Commands Pull-down Menu

The "Commands" pull-down menu is shown below:



■ Clear Display

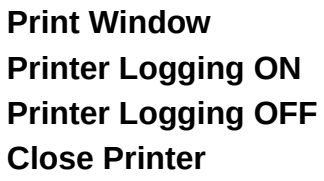
This selection clears the HDSterm window.

■ Redraw Window

This selection redraws the HDSterm window.

■ Printer Options

This selection brings up a sub-menu which has some options for printing. These options are discussed in greater detail in Section 2.2.7.



Clicking on the "Print Window" line sends the current contents of the HDSterm window to the printer. This corresponds to the VT320 command CSI i.

The "Printer Logging" lines refer to the VT320 Autoprint function, corresponding to the CSI ? 4/5 i commands.

The "Close Printer" line empties the printer buffer and closes the connection to the printer.

■ Reset Terminal

This selection resets the HDSterm window.

■ Quit

This selection closes the HDSterm window.

2.2.2.2 Customize Pull-down Menu

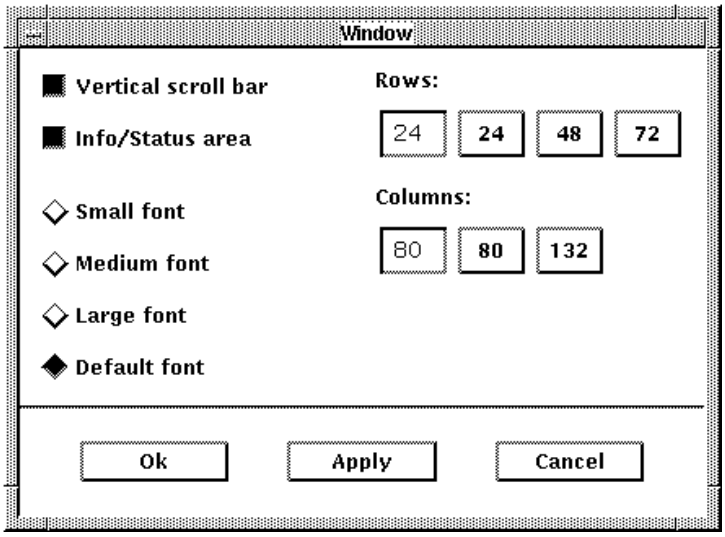
The "Customize" pull-down menu is shown below:



The Customize pull-down menu has three dialog boxes, each with a number of selections. The dialog boxes have three buttons at the bottom to control the settings you have made. The "Ok" button selects your settings and closes the dialog box; the "Apply" button applies your settings immediately so you can see their effect; and the "Cancel" button closes the dialog box.

■ The "Window" Dialog Box

This dialog box selects features about the HDSterm window itself, such as window size and font.



The selections are:

Vertical scroll bar

This selection creates or removes a vertical scroll bar in the HDSTerm window. You can use the scroll bar to move up and down in the window. Utilizing DEC VT320 commands, you can also use the cursor keys for line by line movement or the Shift-cursor keys for screen by screen movement.

Info/Status Area

This selection lets you remove the Information / Status Line from the HDSterm window. Clicking on the button removes this line. The default setting is to display the line. Default settings for the values are shown in bold in the table below.

The Information / Status line displays several of the current HDSterm settings, as follows:

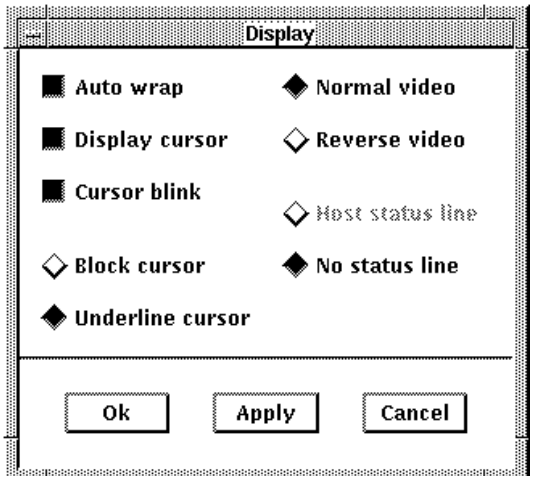
Settings:	**Permitted Values:**
Insert Mode	Insert | **Replace**
Autowrap Mode	**Wrap** | No-wrap
Newline Mode	**LineFeed** | NewLine
Print Device	/dev/*xxx* [**pp0** | tty1 | tty2]
Printer Status	Open | **Closed**
Autoprint Mode	[Logging | ***nothing***]
Printer Controller Mode	[PassThru | ***nothing***]

Small font / Medium font / Large font / Default font

These selections choose the size of the font for the HDSterm window. The actual fonts used for these font selections are set in the HDSterm resource files, which are discussed later.

■ The "Display" Dialog Box

This dialog box selects options for the display within the HDSterm window.



The selections are:

Auto wrap

This setting selects whether Auto wrap is On or Off, that is, whether characters automatically wrap to the next line when they reach the end of a line.

Display cursor

This selects whether the cursor is visible or not.

Cursor blink

This selects whether the cursor, of whatever type, blinks or not.

Block cursor/Underline cursor

This selects whether the window's cursor is a block or an underline cursor.

Normal video/Reverse video

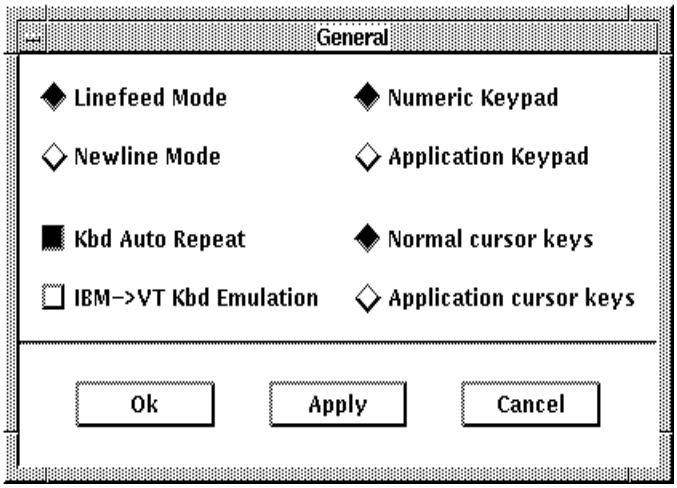
This selects whether the window foreground and background colors are normal or reversed.

Host status line/No status line

This selects whether the window displays a host status line or not. Typically this status line reports error messages or other information coming from the host computer.

■ The "General" Dialog Box

This dialog box selects options for keyboard and text operation for the HDSterm window. These settings are mode switches, either one setting or the other.



Linefeed Mode/Newline Mode

This setting selects whether the Return key sends a linefeed character or a "newline", which is a linefeed and a carriage return. This setting is useful for text editors which may require one or the other for correct operation.

Auto repeat/No Auto repeat

This setting selects whether keys have Auto repeat On or Off.

IBM -> VT Kbd Emulation

This selects whether HDSterm substitutes a VT320 key mapping for users who have an IBM keyboard. This is sometimes useful for editing programs that have

specific DEC keyboard functions. Refer to Section 2.2.8 at the end of this chapter for details about the different mappings.

Numeric Keypad/Application Keypad

This selects whether the Numeric keypad transmits numbers or whether it transmits the special DEC command sequences required in Application mode. Some application programs want the special sequences for correct operation. This can also be set with a DEC escape sequence.

Normal cursor keys/Application cursor keys

This selects whether the cursor keypad transmits cursor control commands or special DEC command sequences required in Application mode. This setting is also available by DEC escape sequence.

2.2.2.3 Serial Options Pull-down Menu

The "Serial Options" pull-down menu is shown below:



The "Serial Options" pull-down menu applies to the Serial Port you selected when you opened the Serial Terminal window, and lets you select the type of connection you want to use, such as SLIP, CSLIP, or PPP. It also offers you a quick way to send a Short Break or a Hangup signal.

For more details and examples of Serial Terminal operation, see Section 1.4.4.

2.2.2.4 Help Pull-down Menu

The "Help" pull-down menu contains one entry:

About HDSterm

This menu brings up a brief message about HDSterm containing references to the location of manual pages and other reference materials that apply to HDSterm.

2.2.3 HDSterm Operation

The HDSterm local client is usually run from menus and button selections, either from Setup Mode to start it in one of its interfaces, such as Telnet or Serial Terminal or Console windows, or from its own menus.

2.2.3.1 HDSterm Manual Starting

You can also start it directly as a client program. From an HDS shell, you can use the command line:

hdsterm -e telnet -T "Telnet"

which starts the HDSterm client as a Telnet window. (The -e command executes a command and the -T command sets the name in the title bar.) This is the command you execute when you click on the Telnet button in the main menu of Setup Mode.

If you want to select a particular host for the Telnet session, use the following command line. If you need to specify a specific telnet port number (default is port 13), you can do it on the command line as the third argument, which is shown as the second example.

```
hdsterm -e telnet <hostname> -T "Telnet"

hdsterm -e telnet <hostname> <port #> -T "Telnet"
```

In the same way, you can start other HDSterm clients for other interfaces:

```
hdsterm -e opendev -T "Terminal"

hdsterm -e /bin/sh -CL -T "Console"

hdsterm -e opendev /dev/tty2 -T "Terminal"

hdsterm -e /bin/sh
```

These are the commands for a Serial Terminal window, a Console window, a Serial Terminal window on serial port 2, and an HDSterm window, respectively. Refer to the HDSterm manual pages following for more command line options and details.

2.2.3.2 HDSterm rlogin Capability

HDSterm supports an rlogin capability. In the simplest form, this allows you to start an HDSterm window with an automatic login to a remote host. Note that the correct permissions to do the rlogin must be in place. This is accomplished by the rsh command, which initially establishes the correct environment and permissions for the rlogin capability. For example:

```
hdsterm -e rlogin <hostname>
```

starts an HDSterm window and automatically logs you into the host. You would typically use this command line in your local window manager's menu (the window manager must have been started by a rsh command).

You can also do this from a remote host with the command:

```
rsh <fxname> hdsterm -e rlogin <hostname>
```

You could also use the command:

```
rsh <fxname> hdsterm -e rlogin
```

with no host name specified, which brings up the host chooser menu so you can pick the host you want. This menu has an added "rlogin" button in the bottom section between the "Telnet" and "Cancel" buttons.

2.2.3.3 Auto-Starting an Application in HDSterm

Many ViewStation users would like to customize their environment and make their operations more productive by having an automatic start for their applications in a terminal window. This is possible with the MIT Xterm client, and many ViewStation users would like the same capability in their HDSterm windows.

Xterm Operation

Automatic starting of clients is possible using Xterm and is typically used with a window manager "root menu" to enable the user to kick-off customized Xterms running specific applications.

For instance, a user might want to start the Uniplex application in an Xterm window from the window manager menu. With Xterm, a user would add a line to their .mwmrc RootMenu such as:

```
"Uniplex" f.exec "xterm -e uniplex"
```

When the user clicks on this menu item, a new Xterm window opens and automatically starts a Uniplex process in the window. When the user quits Uniplex, the Xterm automatically quits too.

HDSterm Operation

The principal difference between Xterm and HDSterm is that one is run on the host computer (Xterm) and the other is run on the local display (HDSterm on the ViewStation). The advantages of this local operation are significant, but it does present a limitation in using an autostart operation.

While HDSterm supports the "execute" (-e) command argument to autostart another program to run in the HDSterm's window, it has the same limitation that Xterm has: The autostart program must be a process running on the "local" machine, that is, the machine running the HDSterm or Xterm client. In the case of Xterm, this can be any process running on the host computer. In the case of HDSterm, it can only be a process running on the ViewStation.

This limitation of running clients only on the local machine applies both to Xterm and HDSterm. For instance, if you wanted to run Xterm on host 1 and execute a client on host 2, the autostart command would fail. You might think that you could get around this limitation of Xterm by typing:

```
xterm -e rsh <anotherhost> uniplex
```

but this process would fail too.

Daemon Operations

The problem with this attempt is that the remote shell process (rsh) connects to a remote shell daemon (rshd) on the remote host, which in turn launches the Uniplex program. But the rsh daemon, unlike the rlogin daemon or the telnet daemon (other processes to do this operation), does not create a pseudo-tty for communi-

cation with the Uniplex application. Most character-based applications that run in a terminal emulation window, including Uniplex, vi, etc., do not operate properly if their input and output is to a network socket rather than to a tty connection.

Unfortunately, neither the rlogin daemon nor the telnet daemon have provision in their operation protocol to launch another task once they set up the pseudo-tty communication. Both of these daemons execute the "/bin/login" process, which records the login system files and then starts up the user's default shell.

An HDSterm solution

A simple solution exists which makes use of HDSterm's customizable "answerback" message. The general idea is to request the answerback message from HDSterm as part of your startup shell script, and then to enter your application program command line as the content of the answerback message. In this way, the application program is started by your startup shell script.

Additional customization, described below, allows you to use this facility from your window manager's menu, and lets you have several applications available to you with this feature.

Follow these steps:

Step 1. Add the following entry at end of your default shell startup script (.cshrc or .profile in your home directory):

```
echo "^E"
```

This "^E" (Control-E) is the standard ENQ or "Enquiry" ASCII control code. Most VTxxx terminals, including HDSterm, respond to this control code by sending a user-specified answerback message (note that currently the standard MIT Xterm ignores the ENQ code). The answerback message is used infrequently and the default setting for most VTxxx terminals is an empty answerback string.

By putting the echo command at the end of the user's startup script, the echo is executed just as the login shell is ready to accept user input. If the answerback message consists of one or more shell commands followed by a newline, the answerback message will provide the commands to be executed by the shell.

Step 2. Make the command line for the answerback message:

If you wanted to start the Uniplex application from HDSterm, use this command line:

```
setenv TERM vt100; uniplex; exit\n
```

Since HDSterm's answerback message is specified by an X resource, it is possible to customize the message for any given instance of the HDSterm client.

For example, the answerback message shown above does three things.

- It sets the TERM environment variable for the window. Note that the rlogin daemon automatically sets the TERM environment variable to the "vt220", since HDSterm specifies this to rlogind. For various reasons, you may wish to override the TERM variable with a different terminal type for the process group running this specific application.

- It starts Uniplex
- It sets the "exit" command, which causes the window to close when you leave Uniplex.

Test the command line for correct operation before you use it.

Step 3. Modify the HDSterm resource to set the new answerback message:

HDSterm's resource which sets the answerback message also includes a number of other items which must remain exactly as in the HDSTerm app-defaults file. You can find this file by opening a Console window and entering these commands:

```
HDS> cd HDS$$ROM

HDS> more HDSTerm
```

This is part of the HDSTerm app-defaults file (the relevant section is at the end of the file).

The following entry is the resource as given in the HDSTerm app-defaults file:

```
*term.vt220-rops: \

1 <%gA%c> \

2 <\\E[%gy%d;%gx%dS> \

3 <\\E[%{gy,1,+,d};%{gx,1,+,d}R> \

6 <%"Welcome to HDSterm"%s> \

7 <\\E[0n> \

8 <\\E[?62;1;2;6;7;8;9c> \

9 <\\E[%{gx,2,+,d};1;1;112;112;1;0;x> \

100 <%{ \

 77 C 75 C 0 pP \

 "ascii" p0 "dec" p1 "graphics" p2 "uk" p3 \

 0 p5, 1 p6 \

 "ascii" pa 32 pb 33 C \

 "dec" pa 160 pb 33 C \

 0 pF \

 0 gT = ? "composeKeys" pa 32 C \

 1 pK 1 pC 1 pX 1 pT ; \

 1 gK = ? "numKeypad" pa 32 C 0 pK ; \

 1 gC = ? "normCKeys" pa 32 C 0 pC ; \

 1 gX = ? "fkeys" pa 32 C \
```

```
"miscFkeys" pa 32 C 0 pX ; \
34 C gy 1 - pa 0 pb 21 C \
53 C 0 px 0 py 1 C \
0 pa 60 C \
0 pb 61 C \
}>
```

The line which specifies the answerback message is the fifth line:

```
6 <%"Welcome to HDSterm"%s> \
```

The procedure to customize the message for specific instances of HDSterm is as follows:

1. Choose a unique name for each instance of HDSterm, one for each different application which you want to autostart.

For example, you could use "Uniplex" for an HDSterm which runs Uniplex, and "Mail" for an HDSterm which runs */usr/ucb/mail*, your mail application.

2. Create a separate resource file entry which contains a "vt220-rops" resource for each different instance, using the chosen names as the "application name". You can do this either by copying this part of the app-defaults file, or by using cut-and-paste from the Console window into your .Xdefaults file. You must use this whole segment of the file.

Customize the answerback string in each resource with the necessary command line to launch the desired application. Follow these examples:

```
Uniplex*term.vt220-rops: \
1 <%gA%c> \
2 <\\E[%gy%d;%gx%dS> \
3 <\\E[%{gy,1,+,d};%{gx,1,+,d}R> \
6 <%"setenv TERM vt100; /usr/bin/uniplex; exit\\n"%s> \
7 <\\E[0n> \
8 <\\E[?62;1;2;6;7;8;9c> \
and so on ....
Mail*term.vt220-rops: \
1 <%gA%c> \
2 <\\E[%gy%d;%gx%dS> \
3 <\\E[%{gy,1,+,d};%{gx,1,+,d}R> \
6 <%"/usr/ucb/mail; exit\\n"%s> \
7 <\\E[0n> \
```

```
8 <\\E[?62;1;2;6;7;8;9c> \

and so on ....
```

Note that you must use the entire "vt220-rops:" entry - these are just file fragments.

The syntax for the answerback line is a "6", followed by a space, followed by "<%", followed by the desired string in double quotes, followed by "%s>", a space, and a backslash. There may not be any spaces or other characters following the backslash. In order for the user's shell to execute the answerback string as a command, it must end with a newline. This is specified by adding \\n just before the terminating double quote (two backslashes before the "n").

3. Put these resources in your .Xdefaults file in your home directory. Preload the .Xdefaults resource file shown above into the ViewStation's RESOURCE_MANAGER property by running **xrdb**. This may be done as part of an Xdm .xsession startup script. Loading these resources into the X Server's RESOURCE_MANAGER property allows them to override HDSterm's app-defaults, without requiring HDSterm to load a remote resource file every time it starts.

Step 4. Add the commands to open the HDSterm window to your client launcher.

For example, you could auto-start Uniplex by adding the following line to your window manager menu (such as .mwmrc) , or to your .xsession file (if you're using Xdm), or some other client launcher file:

For an .mwmrc menu entry:

```
"Uniplex Window" f.exec "hdsterm -name Uniplex -e rlogin
myhost"

"Mail Window" f.exec "hdsterm -name Mail -e rlogin myhost"
```

For an .xsession file:

```
 rsh <ViewStation name> "hdsterm -name Uniplex -e rlogin
<hostname> &"
```

These command lines do a remote login to the host specified and execute the user's startup shell script (which contains the "echo "^E" line at the end).

The "name" argument finds the appropriately renamed *term resource and uses the answerback message defined in the resource entry.

Note that this requires that HDSterm was started remotely, or from a remotely started window manager, and that the */etc/hosts.equiv* and / or *$HOME/.rhosts* files are set up properly. These are prerequisites to correct operation of the remote shell command. It also requires that the user be logged in (username and password) before the command is sent.

If this is correctly configured and run, the rlogin request to the <remote host> will pass the user's name and password, and will then immediately execute the user's startup shell script. The shell script in turn sends the ENQ code to HDSterm,

which then sends the answerback string (your command line) to be executed in the new shell.

If you will be logging in from something other than HDSterm, Xterm, or a VTxxx terminal, you may wish to execute the echo command conditionally based on the appropriate TERM environment variable.

2.2.4 HDSterm Manual Page

This note contains a printed copy of the HDSterm manual page. This page is available on the HDSware tape as an nroff file, labeled hdsterm.1 in the *./hds-fx/man* directory. This means that you can add this nroff file to your other man pages and have help for the HDSterm client available with your normal man utility. If you want to see the file as text output, you can use the command:

```
nroff -man hdsterm.1
```

The HDSterm man page contains all the command line arguments and resources for HDSterm.

= =

HDSTERM(1) USER COMMANDS HDSTERM(1)

NAME

HDSterm - MOTIF based VT320 compatible terminal emulator for X

SYNOPSIS

hdsterm [-*toolkitoption* ...] [-option ...]

DESCRIPTION

The *HDSterm* program is a MOTIF based terminal emulator for the X Window System. It provides DEC VT320 compatible terminals for programs that can't use the window system directly.

OPTIONS

The *HDSterm* terminal emulator accepts all of the standard Goatlike command line options as well as the following (if the option begins with a '+' instead of a '-', the option is restored to its default value):

-help

This causes *HDSterm* to print out a verbose message describing its options.

-cfg *color*

This option specifies the foreground color to use for text cursor. The default is to use the same foreground color that is used for text.

-cbg *color*

This option specifies the background color to use for text cursor. The default is to use the same foreground color that is used for text.

-e *program [arguments ...]*

This option specifies the program (and its command line arguments) to be run in the HDSterm window. **This must be the last option on the command line.**

-fb *font*

This option specifies a font to be used when displaying bold text. This font must be the same height and width as the normal font. If only one of the normal or bold fonts is specified, it will be used as the normal font and the bold font.

-fs *font*

This option specifies a font to be designated as the small font, which is selected via the "Customize Window" menu.

-fm *font*

This option specifies a font to be designated as the medium font, which is selected via the "Customize Window" menu.

-fl *font*

This option specifies a font to be designated as the large font, which is selected via the "Customize Window" menu.

-fsb *font*

This option specifies a font to be designated as the small bold font, which is used for bold text when the small font is selected via the "Customize Window" menu. This font must be the same height and width as the normal small font.

-fmb *font*

This option specifies a font to be designated as the medium bold font, which is used for bold text when the medium font is selected via the "Customize Window" menu. This font must be the same height and width as the normal medium font.

-flb *font*

This option specifies a font to be designated as the large bold font, which is used for bold text when the large font is selected via the "Customize Window" menu. This font must be the same height and width as the normal large font.

-mclick *milliseconds*

This option specifies the maximum time between multi-click selections.

-pcol *color*

This option specifies the color to be used for the pointer cursor. The default is to use the foreground color.

-pcur *cursor*

This option specifies the cursor shape to be used for the pointer cursor. The default is the "xterm'"text cursor.

-vsbar

This option indicates that a scrollbar should be displayed so that the lines which are saved after scrolling off the top can be viewed. This option may be turned on and off from the "Customize Window" menu.

+vsbar

This option indicates that a vertical scrollbar should not be displayed.

-nostatus

This option indicates that the info / status area at the bottom of the main window should not be displayed.

-sl *number*

This option specifies the number of lines to save that have been scrolled off the top of the screen. The default is 64.

-T *string*

This option specifies the title for *HDSterm*'s window. It is equivalent to -title.

-n *string*

This option specifies the icon name for *HDSterm*'s window. It is shorthand for specifying the "*icon-Name" resource. Note that this is not the same as the tool-kit option -name (see below). The default icon name is hdsterm.

-mw *number*

This option specifies the number of pixels between the characters and the left and right window borders.

-mh *number*

This option specifies the number of pixels between the characters and the top and bottom window borders.

-ul *number*

This option specifies the width in pixels of the underline for characters written with the underline attribute.

-ch *number*

This option specifies the height of the text cursor in pixels.

-cw *number*

This option specifies the width of the text cursor in pixels.

-cbl

This option indicates that the text cursor should not blink.

+cbl

This option indicates that the text cursor should blink.

-cbival *number*

This option specifies the text cursor blink interval, in milliseconds. The default is 400 ms.

-tbival *number*

This option specifies the blink interval for text written with the blinking attribute. The default is 500 ms.

-bfocus

This option indicates that the text cursor should stop blinking when the *HDSterm* window loses input focus. This is the default.

+bfocus

This option indicates that the text cursor should continue to blink when the *HDSterm* window loses input focus.

-wrap

This option indicates that text should wrap to column one of the next line after reaching the last column of the previous line. This is the default.

+wrap

This option disables wrapping of text after reaching the last column of the previous line. Additional characters overwrite the character in the last column.

-insert

This option turns off insert mode, and enables replace mode. This is the default.

+insert

This option turns on insert mode, and disables replace mode.

-bell *number*

This option specifies the bell volume, as a percentage from 0 to 100.

-tab *number*

This option specifies that tabstops should be evenly placed at every *number* character positions. The default is 8.

-VT

This option turns on IBM->VT keyboard emulation mode. This is useful when using an IBM style keyboard with applications which use PFx function keys and / or "editing" keys. This option causes function keys F1-F4 to generate the strings normally generated by the PF1-PF4 keys on a VT320 terminal keyboard, and remaps the editing keys to their DEC keyboard equivalents based on position rather than the legends printed on the key tops.

The following standard X Toolkit command line arguments are commonly used with *HDSterm*:

-bg *color*

This option specifies the color to use for the background of the window. The default is "lightsteelblue."

-bd *color*

This option specifies the color to use for the border of the window. The default is ''black.''

-fg *color*

This option specifies the color to use for displaying text. The default is ''black''.

-fn *font*

This option specifies the font to be used for displaying normal text. The default is 9x15.

-name *name*

This option specifies the application name under which resources are to be obtained, rather than the default **HDSTerm**. *Name* should not contain ''.'' or ''*'' characters.

-title *string*

This option specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is hdsterm.

-rv

This option indicates that reverse video should be simulated by swapping the foreground and background colors.

-geometry *geometry*

This option specifies the preferred size and position of the window.

-display *display*

This option specifies the X server to contact; see X(1).

-xrm *resourcestring*

This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

-iconic

This option indicates that *HDSterm* should ask the window manager to start it as an icon rather than as the normal window.

RESOURCES

The program understands all of the core X Toolkit resource names and classes. Note that the client name is HDSTerm (note uppercase "T"). In addition, the following Application Resources are defined:

title (class Title)

Specifies a string that may be used by the window manager when displaying this application.

iconName (class IconName)

This option specifies the icon name for HDSTerm's window.

vertScrollBar (class VertScrollBar)

Setting this option True indicates that a scrollbar should be displayed so that the lines which are saved after scrolling off the top can be viewed. This option may be turned on and off from the ''Customize Window'' menu.

statusLine (class StatusLine)

Setting this option True indicates that the info / status area should be displayed at the bottom of the main window. This option may be turned on and off from the "Customize Window" menu.

smallFont (class SmallFont)

This option specifies a font to be designated as the *small* font, which is selected via the ''Customize Window'' menu.

mediumFont (class MediumFont)

This option specifies a font to be designated as the *medium* font, which is selected via the ''Customize Window'' menu.

largeFont (class LargeFont)

This option specifies a font to be designated as the *large* font, which is selected via the ''Customize Window'' menu.

smallBoldFont (class SmallBoldFont)

This option specifies a font to be designated as the *small bold* font, which is used for bold text when the small font is selected via the ''Customize Window'' menu. This font must be the same height and width as the normal small font.

mediumBoldFont (class MediumBoldFont)

This option specifies a font to be designated as the *medium bold* font, which is used for bold text when the medium font is selected via the ''Customize Window'' menu. This font must be the same height and width as the normal medium font.

largeBoldFont (class LargeBoldFont)

This option specifies a font to be designated as the *large bold* font, which is used for bold text when the large font is selected via the ''Customize Window'' menu. This font must be the same height and width as the normal large font.

vtEmulation (class VtEmulation)

Setting this option True turns on IBM->VT keyboard emulation mode. See the ''-VT'' command line option above for a description.

HDSterm creates a MOTIF MainWindow widget (''top''), consisting of a MenuBar widget (''MenuBar''), a vertical ScrollBar widget (''vsb''), a message area consisting of a TextField status widget (''status''), and some informational Label widgets. The WorkArea of the MainWindow widget is a custom widget (''term'') which is a subclass of the MOTIF Frame widget. The term widget in turn manages a single custom widget (''screen''), which is a subclass of the Xt Core widget. Most of the MOTIF resources applying to

MainWindow, ScrollBar, TextField, Label, Frame, PushButton, CascadeButton, and other MOTIF widgets used may be modified by the user. For example, setting the *top.scrollBarPlacement resource to bottom_right or top_right will move the vertical scroll bar to the right side of the window.

The following resources are specified as part of the term widget (class Term):

command (class Command)

Specifies the command to be run in the HDSTerm window. The default is the local shell (/bin/sh).

termType (class TermType)

Specifies the terminal emulation to be used. Currently only vt320 is supported.

marginWidth (class MarginWidth)

This option specifies the number of pixels between the characters and the left and right window borders.

marginHeight (class MarginHeight)

This option specifies the number of pixels between the characters and the top and bottom window borders.

The following resources are specified as part of the screen widget (class TermCanvas):

background (class Background)

Specifies the color to use for the background of the window. The default is "lightsteelblue."

foreground (class Foreground)

Specifies the color to use for displaying text in the window. The default is "black."

cursorFg (class CursorFg)

Specifies the color to use for the text cursor. The default is "black."

cursorBg (class CursorBg)

Specifies the color to use for the text cursor background. The default is "white."

reverseVideo (class ReverseVideo)

Specifies whether or not reverse video should be simulated. The default is "false."

underlineWidth (class UnderlineWidth)

Specifies the width in pixels of the underline for characters written with the underline attribute.

lines (class Lines)

Specifies the number of lines of text in the displayed window. This resource also sets the initial scrolling region to include all visible lines.

columns (class Columns)

Specifies the number of text columns in the displayed window.

cursorHeight (class CursorHeight)

Specifies the height of the text cursor in pixels.

cursorWidth (class CursorWidth)

Specifies the width of the text cursor in pixels.

cursorBlinking (class CursorBlinking)

Specifies that the text cursor should blink. The default is True.

blinkInterval (class BlinkInterval)

Specifies the interval in milliseconds for the blinking text cursor.

blinkWOFocus (class BlinkWOFocus)

Specifies that the text cursor should continue to blink when the window loses input focus. The default is False.

textBlinkInterval (class TextBlinkInterval)

Specifies the interval in milliseconds for blinking text.

wrapAround (class WrapAround)

Specifies that text should wrap to the next line after reaching the last column in the previous line. The default is True.

insertMode (class InsertMode)

Specifies that text should be inserted, causing characters to the right of the insert point to be shifted right. The default is False.

bellVolume (class BellVolume)

Specifies the bell volume as a percentage from 0 to 100.

defTabWidth (class DefTabWidth)

Specifies the default spacing of initial tabstops.

multiClickTime (class MultiClickTime)

Specifies the maximum time in milliseconds between multi-click select events. The default is 300 milliseconds.

pointerColor (class PointerColor)

Specifies the foreground color of the pointer.

pointerShape (class PointerShape)

Specifies the name of the shape of the pointer. The default is "xterm."

saveLines (class SaveLines)

Specifies the number of lines to save beyond the top of the screen. The default is the number of text lines or 64, whichever is greater.

lineMode (class LineMode)

When set True, specifies that LF, VT, and FF control characters received should cause the cursor to move to the leftmost column of the next line, and scroll if necessary. Also, pressing the "Enter" key will transmit both CR and LF to the program running in the HDSterm window. The default is False, which specifies that LF, VT, and FF cause the cursor to move to the current column of the next line, and pressing "Enter" will transmit only a CR character.

printDevice (class PrintDevice)

Specifies the device driver for local printing. The default is "/dev/pp0" (the parallel port). Serial ports 1 and 2 use drivers "/dev/tty1" and "/dev/tty2" respectively.

ansiFg_n (class AnsiFg_n)

Specifies the color to be used as the text foreground when the ANSI escape sequence to change foreground color is received. The digit ''*n*'', which must be in the range 0-7, corresponds to the ANSI SGR sequence ''ESC [3*n* m''. The ANSI sequence to clear all attributes, ''ESC [0 m'', will reset both the text foreground and background colors to their original values specified by the screen widget's foreground and background resources.

ansiBg_n (class AnsiBg_n)

Specifies the color to be used as the text background when the ANSI escape sequence to change background color is received. The digit ''*n*'', which must be in the range 0-7, corresponds to the ANSI SGR sequence ''ESC [4*n* m''. The ANSI sequence to clear all attributes, ''ESC [0 m'', will reset both the text foreground and background colors to their original values specified by the screen widget's foreground and background resources.

EMULATIONS

The VT320 emulation is fairly complete, but does not support the host-writable status line. *Termcap(5)* entries that work with *HDSterm* include ''vt320'', ''xterm'', ''vt220'', ''vt102'', ''vt100'' and ''ansi''.

POINTER USAGE

Once the HDSterm window is created, *HDSterm* allows you to select text and copy it within the same or other windows.

The selection functions are invoked when the pointer buttons are used with no modifiers. The assignment of the functions described below to keys and buttons may be changed through the resource database; see **ACTIONS** below.

Pointer button one (usually left) is used to save text into the cut buffer. Move the cursor to the beginning of the text, and then hold the button down while moving the cursor to the end of the region and releasing the button. The selected text is highlighted and is saved in the global cut buffer and made the PRIMARY selection when the button is released. Double-clicking selects by words. Triple-clicking selects by lines. Multiple-click is determined by the time from button up to button down, so you can change the selection unit in the middle of a selection. If the key/button bindings specify that an X selection is to be made,

HDSterm will leave the selected text highlighted for as long as it is the selection owner.

Pointer button two (usually middle) 'types' (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button three (usually right) extends the current selection. The original selection starting point is retained, and the endpoint is moved to correspond to the new pointer selection. When button three is released, the newly extended selection is saved.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell, for example, or take output from a program and insert it into your favorite editor. Since the cut buffer is globally shared among different applications, you should regard it as a 'file' whose contents you know. The terminal emulator and other text programs should be treating it as if it were a text file, i.e. the text is delimited by new lines.

The scroll bar position and length corresponds to the position and amount of text currently showing in the window, relative to the amount of text actually saved. As more text is saved (up to the maximum), the length of the scrollbar thumb decreases.

CUSTOMIZING HDSTERM

Once you have created an HDSterm window, you can choose the operating features you want the terminal to have by using the HDSterm Customize menu.

The **Window...** menu item pops up a dialog box which allows you to modify the size and appearance of the terminal window. The current settings are indicated by shaded buttons.

Use the Apply button to view your changes without closing the dialog box. The OK button will apply your changes and close the dialog.

The Window dialog will allow you to enable or disable the vertical scroll bar, change fonts, and change the number of rows and/or columns in the displayed window.

The **Display...** menu item pops up a dialog box which allows you to modify various VT320 emulation features which affect the way text is displayed, including Auto Wrap, Cursor Blink, Cursor Type, and Normal/Reverse video.

The **General...** menu item pops up a dialog box which allows you to modify internal settings of the VT320 emulation, including Linefeed/Newline mode, keyboard autorepeat, keyboard emulation, and the ascii codes generated by numeric keypad keys and cursor keys.

ACTIONS

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input, by changing the translations for the *screen* widget. Changing the translations for events other than key and button events is not expected, and will cause

unpredictable behavior. The following actions are provided for using within the *screen* **translations** resource:

key-input()

Without arguments, this action inserts the 8-bit ISO Latin-1 character or string associated with the keysym that was pressed. If the Mod1 modifier is pressed with the key event, the high-order bit will be set. If a single parameter is supplied, this string will be inserted instead.

insert-selection ***(sourcename [, ...])***

This action inserts the string found in the selection or cutbuffer indicated by sourcename. Sources are checked in the order given (case is significant) until one is found. Commonly-used selections include: *PRIMARY, SECONDARY, and CLIPBOARD*. Cut buffers are typically named *CUT_BUFFER0* through *CUT_BUFFER7*.

select-start()

This action begins text selection at the current pointer location. See the section on **POINTER USAGE** for information on making selections.

select-extend()

This action tracks the pointer and extends the selection.

select-end*(destname [, ...])*

This action puts the currently selected text into all of the selections or cutbuffers specified by destname.

focus-in()

This action modifies the text cursor to be a solid block or solid underline, showing that the window has keyboard focus. If blinking cursor is enabled, the cursor will begin blinking.

focus-out()

This action modifies the text cursor to be an outline block or underline, showing that the window does not have keyboard focus. If blinking cursor is enabled, but blinkWOFocus is False, the cursor stops blinking and becomes solid.

call-parser*(opcode, buffer, [register, int_val], ...)*

This action calls the reverse parser directly. Currently this function is for internal use only.

call-canvas*(opcode, buffer, [register, int_val], ...)*

This action calls the canvas widget function dispatcher directly. Currently this function is for internal use only.

The default bindings in the VT320 window are:

<FocusIn>:	focus-in() \n\
<FocusOut>:	focus-out() \n\
Shift <Key> osfPageUp:	call-canvas(46, "", a, 200, b, 1) \n\

Shift <Key> osfPageDown:	call-canvas(46, "", a, -200, b, 1) \n\
Shift <Key> osfUp:	call-canvas(46, "", a, 1, b, 1) \n\
Shift <Key> osfDown:	call-canvas(46, "", a, -1, b, 1) \n\
<Btn1Down>:	select-start() \n\
<Btn1Motion>:	select-extend() \n\
<Btn2Up>:	insert-selection(PRIMARY, CUT_BUFFER0)\n\
<Btn3Down>:	select-extend() \n\
<Btn3Motion>:	select-extend() \n\
<Btn1Up>:	select-end(PRIMARY, CUT_BUFFER0) \n\
<Btn3Up>:	select-end(PRIMARY, CUT_BUFFER0) \n\
<Key>:	key-input()

ENVIRONMENT

HDSterm sets the environment variable "TERM" to vt320 when the command resource is set to telnet, or when starting HDSterm from **Setup**'s *Telnet Terminal Window* menu button.

SEE ALSO

Xterm(1), X(1)

BUGS

Please report all bugs to Human Designed Systems, Inc. at *support@hds.com*.

COPYRIGHT

Copyright 1992, Human Designed Systems, Inc. Portions adapted from emu, copyright 1990, PCS Computer Systeme GmbH.

AUTHORS

Edward Parks, Michael Hartman, Bill Bredbenner (Human Designed Systems). Thanks to Jordan Hubbard, Michael Elbel, Terry Jones, and Thomas Bagli (PCS Computer Systeme GmbH) for donating the emu source to the MIT Consortium, and to all those *far too many people* who developed Xterm.

2.2.5 HDSTerm Resource Files

Resource files for configuring the HDSterm appearance and operation are contained in resource files located in the ViewStation server code, specifically in the path /HDS$$ROM/app-defaults. These default resources will always be found when the ViewStation is running.

Note that HDSterm resources use uppercase "T" in the client name.

2.2.5.1 HDSTerm Resources in HDS$$ROM

A listing of the default resources follows:

```
HDSTerm*smallFont: 6x10
```

```
HDSTerm*mediumFont: 8x13
HDSTerm*mediumBoldFont: 8x13bold
HDSTerm*largeFont: 10x20
HDSTerm*term.marginWidth: 10
HDSTerm*term.marginHeight: 10
HDSTerm*term.shadowThickness: 4
HDSTerm*screen.WrapAround: True
HDSTerm*screen.CursorBlinking: True
HDSTerm*screen.BlinkWOFocus: False
HDSTerm*screen.BlinkInterval: 400
HDSTerm*screen.CursorHeight: 3
HDSTerm*screen.CursorWidth: 0
HDSTerm*screen.font: 9x15
HDSTerm*screen.background: lightsteelblue
HDSTerm*screen.printDevice:  /dev/pp0
HDSTerm*top.scrollBarPlacement: bottom_left
HDSTerm*XmText.fontList: 9x15
HDSTerm*XmTextField.fontList: 9x15
HDSTerm*XmPushButton.fontList: menu12
HDSTerm*XmPushButtonGadget.fontList: menu12
HDSTerm*XmLabel.fontList: menu12
HDSTerm*XmLabelGadget.fontList: menu12
HDSTerm*XmCascadeButton.fontList: menu12
HDSTerm*XmCascadeButton.marginWidth: 10
HDSTerm*XmToggleButtonGadget.fontList: menu12
HDSTerm*XmList.fontList: 9x15
HDSTerm*hosts.selectionLabelString: Selected Host:
HDSTerm*hosts.listLabelString: Host List:
HDSTerm*hosts.listVisibleItemCount: 6
HDSTerm*serial_port.messageString: Please Select Serial
Port:
HDSTerm*sessionl.labelString: Session:
```

```
HDSTerm*hostl.labelString: Connected To:
```

2.2.5.2 HDSTerm Resources in $HOME/.Xdefaults

You can use your own resource specifications for HDSterm by creating a resource file of your own. These resources will be read from the home directory, or some other location, and will replace some or all of the default resources in HDS$$ROM as you specify.

For example, if you want to use different settings or different fonts or colors, you could change the resource specifications as follows:

```
HDSTerm*largeFont: 9x15

HDSTerm*screen.BlinkInterval: 200

HDSTerm*screen.background: darksteelblue
```

2.2.6 HDSterm Command Sequences

The HDSterm client is a VT320 terminal emulator. It responds to all the standard VT320 command sequences. These are listed below:

HDSterm uses 7-bit control characters. Therefore, 8-bit control codes (values 0x80 through 0x9f) should be expressed as two byte sequences:

For example, the CSI character (decimal 155, or hex 9B) :

CSI (ASCII 0x09B) should be expressed as "ESC ["

DCS (ASCII 0x090) should be expressed as "ESC P"

and so on.

2.2.6.1 HDSterm Supported Command Set:

■ Designating "Hard" Character Sets

ESC (B	- ASCII -> G0
ESC) B	- ASCII -> G1
ESC * B	- ASCII -> G2
ESC + B	- ASCII -> G3
ESC (<	- DEC Suppl. -> G0
ESC) <	- DEC Suppl. -> G1
ESC * <	- DEC Suppl. -> G2
ESC + <	- DEC Suppl. -> G3
ESC (A	- U.K. Ntl. -> G0
ESC) A	- U.K. Ntl. -> G1
ESC (0	- DEC spec. graphics -> G0
ESC) 0	- DEC spec. graphics -> G1
ESC * 0	- DEC spec. graphics -> G2

ESC + 0 - DEC spec. graphics -> G3

■ Invoking Character Sets Using Locking Shifts

SI (Ctrl O) - Invoke G0 into GL (default)

SO (Ctrl N) - Invoke G1 into GL

ESC ~ - Invoke G1 into GR

ESC n - Invoke G2 into GL

ESC } - Invoke G2 into GR (default)

ESC o - Invoke G3 into GL

ESC | - Invoke G3 into GR

■ Invoking G2 or G3 into GL For a Single Character Using Single Shifts

SS2 (ESC N) - Single Shift G2

SS3 (ESC O) - Single Shift G3

■ Terminal Modes

ANSI modes:

CSI *Ps ; Ps ;* ... l - Reset mode

CSI *Ps ; Ps ;* ... h - Set mode

Ps = 4 - replace mode / insert mode

Ps = 20 - linefeed mode / newline mode

DEC private modes:

CSI ? *Ps ; Ps ;* ... h - Set mode

CSI ? *Ps ; Ps ;* ... l - Reset mode

Ps = 1 - cursor keys generate "application control" /

ANSI cursor control sequences

Ps = 3 - 132 / 80 columns per line

Ps = 5 - black chars on white / white chars on black

Ps = 6 - cursor positioning relative to scroll area / global

Ps = 7 - wrap / don't wrap at the end of the line

Ps = 8 - keyboard autorepeat on / off

Ps = 25 - cursor visible / invisible

ESC = - select application keypad mode

ESC > - select numeric keypad mode

■ Cursor Positioning

CSI *Pn* A - move cursor up Pn lines

CSI *Pn* B	- move cursor down Pn lines
CSI *Pn* C	- move cursor right Pn columns
CSI *Pn* D	- move cursor left Pn columns
CSI *Pl ; Pc* H	- move cursor to line Pl, columns Pc
CSI *Pl ; Pc* f	- move cursor to line Pl, columns Pc
IND (ESC D)	- move cursor down one line, scrolled
RI (ESC M)	- move cursor up one line, scrolled
NEL (ESC E)	- move cursor down one line and to the first column, scrolled
^H (0x08)	- move cursor left one column
^J (0x0a)	- perform "newline" function, which is either the same as IND or NEL, depending on the current linefeed / newline mode.
^K (0x0b)	- vertical tab, behaves as ^J above
^L (0x0c)	- form feed, behaves as ^J above
^M (0x0d)	- move cursor to first column of current line
ESC 7	- save states: cursor postion graphic rendition character set shift state wrap flag origin mode (not supported) selective erase (not supported)
ESC 8	- restore saved states

■ TAB Stops

HTS (ESC H)	- Set tab stop at the current column
CSI g	- Clear tab at current column
CSI 0 g	- Clear tab at current column
CSI 3 g	- Clear all tab stops
^I (0x09)	- Move cursor right to next tab stop

■ Character Rendition and Attributes

CSI *Ps ; Ps ;* ... m	- Select character rendition with
Ps = 0	- everything off
Ps = 1	- display bold

Ps = 4	- display underlined
Ps = 5	- display blinking
Ps = 7	- display reverse image
Ps = 22	- display normal intensity
Ps = 24	- display not underlined
Ps = 25	- display not blinking
Ps = 27	- display positive image

■ Double width / height text.

ESC # 3	- double height line top half
ESC # 4	- double height line bottom half
ESC # 5	- single width single height line
ESC # 6	- double width single height line

■ Editing

CSI *Pn* L	- insert Pn lines
CSI *Pn* M	- delete Pn lines
CSI *Pn* @	- insert Pn blank characters with attributes set to normal
CSI *Pn* P	- delete Pn characters
CSI *Pn* X	- erase Pn characters
CSI K	- erase to the end of the line including cursor
CSI 0 K	- erase to the end of the line including cursor
CSI 1 K	- erase form the beginning of the line including cursor
CSI 2 K	- erase the complete line
CSI J	- erase from cursor to end of screen
CSI 0 J	- erase from cursor to end of screen
CSI 1 J	- erase from beginning of screen to cursor
CSI 2 J	- erase the complete display

■ Scrolling Margins

CSI *Pt ; Pb* r	- Sets top and bottom margins of the scrolling region to Pt and Pb
CSI r	- Sets top and bottom margins of the scrolling region to the first and last displayed lines

■ Screen Alignment pattern

ESC # 8	- displays a full screen of E's

■ Bell

^G (0x07)	- ring keyboard bell

■ Reset

ESC c	- reset terminal emulation

■ Reports

CSI *Ps* n	- Device Status Report
Ps = 5	- Operating status
Ps = 6	- Request cursor position
Ps = 7	- Request screen size
CSI 0 c	(or CSI c)- Terminal ID
CSI *Ps* x	- Request Terminal Parameters
^E (0x05)	- Send answerback message

■ Set Window, Icon Label

ESC] 0 ; <*label*> ^G	- Set window and icon labels to <*label*>

■ Local Printing

ESC [i	- Print screen
ESC [0 i	- Print screen
ESC [4 i	- Printer controller mode off
ESC [5 i	- Printer controller mode on
ESC [? 1 i	- Print line which has cursor
ESC [?4 i	- Autoprint mode off
ESC [?5 i	- Autoprint mode on

■ ANSI Color Support

HDSterm supports escape sequences to change foreground and background colors, as follows:

ESC [30 m	Foreground black
ESC [31 m	Foreground red
ESC [32 m	Foreground green
ESC [33 m	Foreground yellow
ESC [34 m	Foreground blue
ESC [35 m	Foreground magenta
ESC [36 m	Foreground cyan
ESC [37 m	Foreground white
ESC [40 m	Background black

ESC [41 m	Background red
ESC [42 m	Background green
ESC [43 m	Background yellow
ESC [44 m	Background blue
ESC [45 m	Background magenta
ESC [46 m	Background cyan
ESC [47 m	Background white

Note that ESC [0 m resets all attributes to Off. It also resets the text background and foreground colors to the initial colors selected for the screen widget (default is foreground = black and background = lightsteelblue). These escape sequences are ignored on a monochrome server.

These colors are the ANSI standard selections, but the actual colors can be modified by using HDSterm resources that correspond to the escape sequences above. The HDSterm resources to define the colors are in the following form:

HDSTerm*screen.ansiFg_0 = defines the color for ESC [30 m

HDSTerm*screen.ansiFg_6 = defines the color for ESC [36 m

HDSTerm*screen.ansiBg_1 = defines the color for ESC [41 m

HDSTerm*screen.ansiBg_3 = defines the color for ESC [43 m

and so on.

2.2.7 HDSterm Print Commands

HDSterm has local printing support available from a sub-menu on the Commands pull-down menu. The printer is connected to the ViewStation, either to the parallel or one of the serial ports. This printer location is specified by an HDSterm resource:

```
HDSTerm*screen.printDevice:  /dev/pp0
```

which specifies the parallel port (/dev/pp0 is the default entry). You can also specify either of the serial ports, /dev/tty1 (serial port 1 - RJ11) or /dev/tty2 (serial port 2 - DB25).

The following commands are available from the Print Options sub-menu.

- Print Window
- Printer Logging ON
- Printer Logging OFF
- Close Printer

Clicking on the "Print Window" line sends the current contents of the HDSTerm window to the printer. This corresponds to the VT320 command CSI i.

The "Printer Logging ON" and "Printer Logging OFF" lines refer to the VT320 Autoprint function, corresponding to the CSI ? 4 i and CSI ? 5 i commands, respectively.

The "Close Printer" line will empty the printer buffer and close the connection to the printer.

The following VT320 command sequences for printing are supported:

ESC [i	- Print screen
ESC [0 i	- Print screen
ESC [4 i	- Printer controller mode off
ESC [5 i	- Printer controller mode on
ESC [? 1 i	- Print line which has cursor
ESC [?4 i	- Autoprint mode off
ESC [?5 i	- Autoprint mode on

If the printer device driver fills its buffer so printing is stopped (usually caused by the printer going Offline or running out of paper), HDSterm pops up a warning window asking you to OK the command to close the printer. If you can fix the problem so the printer can resume its printing and the buffer starts to empty, the warning will disappear. All communications with the printer port are suspended while the warning window is displayed. Clicking on the "OK" or closing the warning window with the window manager closes the printer port and loses the data remaining in the printer buffer.

2.2.8 VT320 Keyboard Mapping

The HDS ViewStation can use different keyboards, though the most common types are IBM 101+ and the DEC VT320 type. These keyboards have different layouts and handle several important keyboard functions in different ways, specifically, the Numeric keypad, the editing keypad, and the function keys. HDSterm, as a VT320 emulation, generally expects these key functions to correspond to DEC keys and key codes.

The HDSterm client allows you to select, from the Customize Window General dialog box, whether your keyboard (IBM or DEC type) sends VT320 sequences or the normal keyboard sequences. The "VT On" selection is recommended for HDSterm for both the IBM and DEC keyboards. The actual selections and their corresponding sequences are shown in the tables below.

2.2.8.1 Using IBM 101+ or 102 keyboards:

Key Pressed	IBM -> VT On (rec.)	IBM -> VT Off
F1	ESC O P	ESC [11~
F2	ESC O Q	ESC [12~
F3	ESC O R	ESC [13~
F4	ESC O S	ESC [14~
Insert	ESC [1~	ESC [2~
Home	ESC [2~	- nothing -
Page Up	ESC [3~	ESC [5~

Delete	ESC [4~	DEL (hex 0x7F)
End	ESC [5~	- nothing -
Page Down	ESC [6~	ESC [6~
Keypad "+"	+	+
Alt Keypad "+"	, (or ESC Ol)	, (or ESC Ol)

2.2.8.2 Using DEC VT320 keyboards

Key Pressed	IBM -> VT On	IBM -> VT Off(rec.)
F1	ESC O P	ESC O P
F2	ESC O Q	ESC O Q
F3	ESC O R	ESC O R
F4	ESC O S	ESC O S
PF1	ESC O P	ESC O P
PF2	ESC O Q	ESC O Q
PF3	ESC O R	ESC O R
PF4	ESC O S	ESC O S
Find	ESC [1~	ESC [1~
Insert Here	ESC [1~	ESC [2~
Remove	ESC [3~	ESC [3~
Select	ESC [4~	ESC [4~
Prev	ESC [5~	ESC [5~
Next	ESC [6~	ESC [6~
Keypad ","	, (or ESC Ol)	, (or ESC Ol)

The Numeric keypad sequence depends on whether HDterm is using Numeric or Application mode. In Application mode the sequence is ESC O l, which is uppercase O, not zero, and lowercase L, not one.

Note also that using the Meta key (if defined with xmodmap for your keyboard) sets the high bit (8th bit) on for any ASCII character.

2.2.9 HDSterm Key Translations

Unlike many X Windows applications, the keyboard translations for HDSterm are not applied as a single default resource, which you modify by applying your own "HDSTerm*translations" resource. Instead, the translations are dynamically applied by HDSterm at run time, using several sets of translations. These sets are stored as application sub-resources, using the instance name "*term.vt220-aux-Trans-<set name>".

The following sets of translations are used (see full listings below):

Key-set Name Translation Application

appKeypad	Map numeric keypad keys to DEC VT220/320 "application mode" strings
numKeypad	Map numeric keypad keys to DEC VT220/320 "numeric mode" strings
appCKeys	Map cursor keys to DEC VT220/320 application mode" strings
normCKeys	Map cursor keys to DEC VT220/320 normal mode strings
fkeys	Map function keys to the strings generated by XTerm
ibmFkeys	Map function keys F1 to F4 to the DEC VT220/320 PF1 to PF4 strings
miscFkeys	Map "editing" keys to their closest equivalent to DEC VT220/320 editing keys, **based on their key legends**
ibmMiscFkeys	Map "editing" keys on IBM keyboard **based on their positional equivalent** on a DEC VT220/320 keyboard
telnetFkeys	Map Alt-F10 to Alt-F12 to generate complete shell commands to set the DISPLAY environment variable to the ViewStation's IP address:0

The key to these translations is the use of the action procedure "key-input". If key-input has a parameter, it is taken to be an ASCII string which should be transmitted when the specified input event occurs. If key-input is used without a parameter, the specified event sends the normal ASCII binding returned by Xlib's XLookupString() function.

The "application" mode vs. "normal" or "numeric" mode sets are applied as a result of either user input (from the Dialog box), or receipt of escape sequences. The "normal" vs. "ibm" sets are applied only when the user selects/deselects IBM->VT Kybd Emulation from the Customize General dialog box. The "Telnet" set is applied if HDSTerm is started from Setup as a Telnet window, or from a shell with the "-e telnet" command-line option.

2.2.9.1 Sample Key Translations

To customize the keyboard translations, you must first determine which keyset the keys you wish to change belong to. Keys that belong to one of the special sets can be redefined as part of the set. Keys that do not belong to a special set can be redefined with normal key override translations.

If the key to be redefined is not in one of the special sets, you can simply use the "*screen.translations" resource. Note that some keys have special definitions in the Motif widget set, such as F1, F4, and F10 (see the following table). If you use

these keys you must refer to them by that name in the translation table. In the special case of F10, which is an accelerator key, you must explicitly disable the key before you redefine it in order to disable the accelerator function, which normally overrides the key translation (an example is given below).

For ordinary keys, you can just redefine them; for instance, the Backspace key on an IBM keyboard, designated as osfBackSpace, often needs to be remapped to the Delete key for DEC applications. Since this key doesn't appear in any keyset group, you can use a normal screen translation, such as (but note the use of the osfBackSpace name for the keysym):

```
HDSTerm*screen.translations:  #override \
          <Key>osfBackSpace:  key-input("\177")
```

If the key you want to redefine does occur in one of these keyset groups, you must use the entire keyset in your translation. This is necessary because your new keyset resource definition **completely replaces** the built-in default for that resource. Therefore, you may not use an "#override" directive and simply list only those keys you want changed. Rather, you must include all keys which have translations defined in the built-in defaults. For keys not found in the special sets, you may use the "#override" directive on the *screen.translations resource, as these are applied at widget initialization.

For example, if you want Shift-F9 to generate "cd $HOME" and F10 to function as a normal function key instead of the osfMenuBar, you would have to replace the entire "*term.vt220-auxTrans-fkeys" resource. In the example below, this definition is added for the Shift-F9 key. Note that the unshifted F9 must now be explicitly prefaced by ~Shift (this means "not Shift") and the beginning colon removed (the colon used with other keys means "ignore modifier keys"). The F10 key first must have the accelerator function disabled by entering a null value, and is then remapped in the normal way. Note the use of the osf key binding names instead of the normal keysyms.

```
! this sets the osfMenuBar key (F10) to a null value
HDSTerm*menuBar.menuAccelerator: ""
HDSTerm*term.vt220-auxTrans-fkeys: \
 ~Shift <Key>osfHelp: key-input("Ctrl-[[11~")\n \
 Shift <Key>osfHelp: key-input("Ctrl-[[25~")\n \
 :<Key>F2: key-input("Ctrl-[[12~")\n \
 :<Key>F3: key-input("Ctrl-[[13~")\n \
 :<Key>osfMenu: key-input("Ctrl-[[14~")\n \
 :<Key>F5: key-input("Ctrl-[Ctrl-[[15~")\n \
 :<Key>F6: key-input("Ctrl-[[17~")\n \
 :<Key>F7: key-input("Ctrl-[[18~")\n \
```

```
    :<Key>F8: key-input("Ctrl-[[19~")\n \
    ~Shift <Key>F9: key-input("Ctrl-[[20~")\n \
    Shift <Key>F9: key-input("cd $HOME")\n \
    ~Shift <Key>osfMenuBar: key-input("Ctrl-[[21~")\n \
    Shift <Key>osfMenuBar: key-input("Ctrl-[[22~")\n \
    :<Key>F11: key-input("Ctrl-[[23~")\n \
    :<Key>F12: key-input("Ctrl-[[24~")\n \
    :<Key>F13: key-input("Ctrl-[[25~")\n \
    :<Key>F14: key-input("Ctrl-[[26~")\n \
    :<Key>F15: key-input("Ctrl-[[28~")\n \
    :<Key>Help: key-input("Ctrl-[[28~")\n \
    :<Key>F16: key-input("Ctrl-[[29~")\n \
    :<Key>Menu: key-input("Ctrl-[[29~")\n \
    :<Key>F17: key-input("Ctrl-[[31~")\n \
    :<Key>F18: key-input("Ctrl-[[32~")\n \
    :<Key>F19: key-input("Ctrl-[[33~")\n \
    :<Key>F20: key-input("Ctrl-[[34~")\n \
    :<Key>KP_F1: key-input("Ctrl-[OP")\n \
    :<Key>KP_F2: key-input("Ctrl-[OQ")\n \
    :<Key>KP_F3: key-input("Ctrl-[OR")\n \
    :<Key>KP_F4: key-input("Ctrl-[OS")\n \

*****************************************************************************
   HDSTERM DEFAULT TRANSLATION DEFINITIONS
*****************************************************************************
/* Note: The default MOTIF key bindings are used, so the following     */
/* standard X Windows keysyms must be replaced by their MOTIF          */
/* bindings in any event translation resources:                        */
/*                                                                     */
/* X KeySym                    MOTIF KeySym                            */
/* ------------                ----------------------                  */
/* Escape                      osfCancel                               */
/* Left                        osfLeft                                 */
```

```
/* Up                          osfUp                    */
/* Right                       osfRight                 */
/* Down                        osfDown                  */
/* End                         osfEndLine               */
/* Home                        osfBeginLine             */
/* Prior                       osfPageUp                */
/* Next                        osfPageDown              */
/* BackSpace                   osfBackSpace             */
/* Delete                      osfDelete                */
/* Insert                      osfInsert                */
/* F1                          osfHelp                  */
/* F4                          osfMenu                  */
/* Shift F8                    osfAddMode               */
/* F10                         osfMenuBar               */
/* Select                      osfSelect                */
/* KP_Enter                    osfActivate              */
/* Clear                       osfClear                 */
/* Undo                        osfUndo                  */

/* Application Keypad */
*term.vt220-auxTrans-appKeypad:                        \
    /* Group 1 */                                      \
    :<Key>KP_0:             key-input("Ctrl-[Op")\n    \
    :<Key>KP_1:             key-input("Ctrl-[Oq")\n    \
    :<Key>KP_2:             key-input("Ctrl-[Or")\n    \
    :<Key>KP_3:             key-input("Ctrl-[Os")\n    \
    :<Key>KP_4:             key-input("Ctrl-[Ot")\n    \
    :<Key>KP_5:             key-input("Ctrl-[Ou")\n    \
    :<Key>KP_6:             key-input("Ctrl-[Ov")\n    \
    :<Key>KP_7:             key-input("Ctrl-[Ow")\n    \
    :<Key>KP_8:             key-input("Ctrl-[Ox")\n    \
    :<Key>KP_9:             key-input("Ctrl-[Oy")\n    \
    /* Group 2 */                                      \
    :<Key>KP_Decimal:       key-input("Ctrl-[On")\n    \
    :<Key>KP_Subtract:      key-input("Ctrl-[Om")\n    \
```

```
    :<Key>KP_Separator:          key-input("Ctrl-[Ol")\n        \
Alt  <Key>KP_Add:                key-input("Ctrl-[l")\n         \
    :<Key>osfActivate:           key-input("Ctrl-[OM")\n

/* Numeric Keypad */
*term.vt220-auxTrans-numKeypad:                                 \
    /* Group 1 */                                               \
    :<Key>KP_0:                  key-input()\n                  \
    :<Key>KP_1:                  key-input()\n                  \
    :<Key>KP_2:                  key-input()\n                  \
    :<Key>KP_3:                  key-input()\n                  \
    :<Key>KP_4:                  key-input()\n                  \
    :<Key>KP_5:                  key-input()\n                  \
    :<Key>KP_6:                  key-input()\n                  \
    :<Key>KP_7:                  key-input()\n                  \
    :<Key>KP_8:                  key-input()\n                  \
    :<Key>KP_9:                  key-input()\n                  \
    /* Group 2 */                                               \
    :<Key>KP_Decimal:             key-input()\n                 \
    :<Key>KP_Subtract:           key-input()\n                  \
    :<Key>KP_Separator:          key-input()\n                  \
Alt  <Key>KP_Add:                key-input(",")\n               \
    :<Key>osfActivate:           key-input()\n

/* Application Cursor Keys */
*term.vt220-auxTrans-appCKeys:                                  \
    ~Shift <Key>osfLeft:         key-input("Ctrl-[OD")\n        \
    ~Shift <Key>osfRight:        key-input("Ctrl-[OC")\n        \
    ~Shift <Key>osfDown:         key-input("Ctrl-[OB")\n        \
    ~Shift <Key>osfUp:           key-input("Ctrl-[OA")\n

/* Normal Cursor Keys */
*term.vt220-auxTrans-normCKeys:                                 \
    ~Shift <Key>osfLeft:         key-input("Ctrl-[[D")\n        \
    ~Shift <Key>osfRight:        key-input("Ctrl-[[C")\n        \
```

```
    ~Shift <Key>osfDown:          key-input("Ctrl-[[B")\n          \
    ~Shift <Key>osfUp:            key-input("Ctrl-[[A")\n

/* Function Keys */
*term.vt220-auxTrans-fkeys:                                        \
    :<Key>osfHelp:                key-input("Ctrl-[[11~")\n        \
    :<Key>F2:                     key-input("Ctrl-[[12~")\n        \
    :<Key>F3:                     key-input("Ctrl-[[13~")\n        \
    :<Key>osfMenu:                key-input("Ctrl-[[14~")\n        \
    :<Key>F5:                     key-input("Ctrl-[[15~")\n        \
    :<Key>F6:                     key-input("Ctrl-[[17~")\n        \
    :<Key>F7:                     key-input("Ctrl-[[18~")\n        \
    :<Key>F8:                     key-input("Ctrl-[[19~")\n        \
    :<Key>F9:                     key-input("Ctrl-[[20~")\n        \
    :<Key>F10:                    key-input("Ctrl-[[21~")\n        \
    :<Key>F11:                    key-input("Ctrl-[[23~")\n        \
    :<Key>F12:                    key-input("Ctrl-[[24~")\n        \
    :<Key>F13:                    key-input("Ctrl-[[25~")\n        \
    :<Key>F14:                    key-input("Ctrl-[[26~")\n        \
    :<Key>F15:                    key-input("Ctrl-[[28~")\n        \
    :<Key>Help:                   key-input("Ctrl-[[28~")\n        \
    :<Key>F16:                    key-input("Ctrl-[[29~")\n        \
    :<Key>Menu:                   key-input("Ctrl-[[29~")\n        \
    :<Key>F17:                    key-input("Ctrl-[[31~")\n        \
    :<Key>F18:                    key-input("Ctrl-[[32~")\n        \
    :<Key>F19:                    key-input("Ctrl-[[33~")\n        \
    :<Key>F20:                    key-input("Ctrl-[[34~")\n        \
    :<Key>KP_F1:                  key-input("Ctrl-[OP")\n          \
    :<Key>KP_F2:                  key-input("Ctrl-[OQ")\n          \
    :<Key>KP_F3:                  key-input("Ctrl-[OR")\n          \
    :<Key>KP_F4:                  key-input("Ctrl-[OS")\n

/* IBM -> VT Remap Function Keys */
*term.vt220-auxTrans-ibmFkeys:                                     \
    :<Key>osfHelp:                key-input("Ctrl-[OP")\n          \
```

```
    :<Key>F2:                       key-input("Ctrl-[OQ")\n      \
    :<Key>F3:                       key-input("Ctrl-[OR")\n      \
    :<Key>osfMenu:                  key-input("Ctrl-[OS")\n

/* Misc. Function Keys (editing keys) */
*term.vt220-auxTrans-miscFkeys:                                  \
    ~Shift <Key>Find:               key-input("Ctrl-[[1~")\n     \
    ~Shift <Key>osfInsert:          key-input("Ctrl-[[2~")\n     \
    ~Shift <Key>DRemove:            key-input("Ctrl-[[3~")\n     \
    ~Shift <Key>osfSelect:          key-input("Ctrl-[[4~")\n     \
    ~Shift <Key>osfPageUp:          key-input("Ctrl-[[5~")\n     \
    ~Shift <Key>osfPageDown:        key-input("Ctrl-[[6~")\n     \
    ~Shift <Key>osfBeginLine:       key-input()\n                \
    ~Shift <Key>osfEndLine:         key-input()\n                \
    ~Shift <Key>osfDelete:          key-input()\n

/* IBM -> VT Remap Misc. Function Keys (editing keys) */
*term.vt220-auxTrans-ibmMiscFkeys:                               \
    ~Shift <Key>osfInsert:          key-input("Ctrl-[[1~")\n     \
    ~Shift <Key>osfBeginLine:       key-input("Ctrl-[[2~")\n     \
    ~Shift <Key>osfPageUp:          key-input("Ctrl-[[3~")\n     \
    ~Shift <Key>osfDelete:          key-input("Ctrl-[[4~")\n     \
    ~Shift <Key>osfEndLine:         key-input("Ctrl-[[5~")\n     \
    ~Shift <Key>osfPageDown:        key-input("Ctrl-[[6~")\n

/* Redefine function keys for use with Telnet, to provide DISPLAY variable */
/* osfMenuBar is XK_F10, osfCancel is XK_Escape */

*term.vt220-auxTrans-telnetFkeys:       \
 Alt  ~Ctrl ~Shift<Key>osfMenuBar:      key-input("set display/create
                                        /server=0/")\
                                        key-input("trans=tcpip/node=")
                                        call-parser(10,"")\ key-input(\015)
                                        \n\
 Alt  ~Ctrl ~Shift<Key>F11:             key-input("DISPLAY=") call
                                        -parser(10,"")\
```

	key-input(":0;export DISPLAY") key -input(\015) \n\
Alt ~Ctrl ~Shift<Key>osfCancel:	key-input("DISPLAY=") call -parser(10,"")\
	key-input(":0;export DISPLAY") key -input(\015) \n\
Alt ~Ctrl ~Shift<Key>F12:	key-input("setenv DISPLAY ") call -parser(10,"")\
	key-input(":0") key-input(\015)

HDS Local Terminal Emulation Clients

This chapter describes other HDS ViewStation Local Terminal Emulation clients, HDS3270, Open Connect tn3270, and Tandem 6530.

Section 2.0 - HDS ViewStation Local Clients

2.3 HDS Local Terminal Emulation Clients

ViewStation terminal emulation clients are the primary user interface to a host computer. They control the appearance of the login window, its keymappings, and its operating characteristics.

The principal terminal emulation is HDSterm, which is described in Section 2.2.

Other terminal emulations are HDS3270, which simulates an IBM 3270 terminal. The HDS3270 client is useful when connecting to an IBM mainframe computer that expects 3270 terminal commands. The HDS3270 client uses a normal IBM 101 keyboard, though a Lexmark 122 keyboard is also supported.

2.3.1 Using Local Clients

The HDSperx local clients are available with HDSware (except for licensed products) in the */hds-fx/bin* directory. There are also entries for the clients in the */hds-fx/config/clients.config* file, though the entries for some clients may be commented out. You can add them to your window manager's pull-down menu by removing the comment characters.

Refer to Section 2.1 for more details. Note that the hds3270 client requires a hostname on its command line, so simply uncommenting the entry will not start the client; you must add the host name to the defArgs command line.

2.3.2 HDS3270 Terminal Emulation

This is the HDS3270 client manual page.

NAME

hds3270 - IBM remote host access tool

SYNOPSIS

hds3270 [-model [2345]] [-port portnum] [-keymap name] [-initstr string] [-mono] host

DESCRIPTION

hds3270 opens a telnet connection to the given IBM host in an X window. The created window provides a fairly complete IBM 3278 terminal emulation. Since the window created by hds3270 uses its own font for displaying characters, is a fairly accurate representation of a 3278. It is similar to tn3270 (1) except that it is X based, not curses based.

hds3270 is a toolkit based program, it also understands all of the standard options and resources. The most important are "-rv" and "*reverseVideo" on mono systems.

The type of 3278 to be emulated is given by the -mode switch. The sizes of various emulated models are:

l. Model 2 80x24 Model 3 80x30 Model 4 80x43 Model 5 132x27

The default is Model 2 emulation.

The -port switch can be used to define the TCP port to connect to. The default port is the telnet port.

The -initstr switch can be used to specify a string to send the remote host once the TCP connection is established.

KEYMAPS

The type of keyboard may be specified with the -keymap switch or using either the KEYMAP or KEYBD environment variables. The types of supported keyboards include ibm (standard IBM 101/102 keyboards) and lex3270 (Lexmark 122 keyboard configured to use 3270 KeySyms). If you make changes to the keymap file, you must specify the -keymap <name> argument, otherwise the client reads its internal default ibm keymap.

The X Toolkit translation mechanism is used to provide keyboard emulation. It maps events into actions. The best documentation can be found with X11R4 X toolkit documents, but the following should suffice for simple customization.

An Xt event consists of (at least) four fields. The first is called a modifier. It may be any combination of alt, meta, shift and ctrl. If it is prefaced by !, it means those modifiers only. The second field is the specific event, in hds3270 usually just <Key>. The third field is the detail field, which gives the actual key. The name of the key may be determined using the R4 xev program. The last field is the action, which is the internal emulator function. A complete list of actions may be found later in the manual.

There are three levels of translation tables in hds3270. The first is a default, compiled in table. It defines alphabetic, numeric, function keys, and such basic functions as Enter and Delete. It allows a minimal useful functionality.

The second level is a keyboard specific table, which is found in the application default file, which defines actions for such things as keypad keys, and keys unique to certain keyboards.

The third level is a user customizable table which may be used to augment or override key definitions. You must specify the -keymap <name> argument if you make changes to this file, otherwise the client will simply read its internal default ibm file. The keymap file is typically found in the users .Xdefaults file. The naming for a lex122 keyboard would be:

hds3270.keymap.lex122:

hds3270.keymap.lex122.user:

The default translation table is:

l l .

```
<Expose>:                          Redraw()\n\
Shift<Key>Return:                  Newline()\n\
~Shift<Key>Return:                 Enter()\n\
Shift<Key>osfActivate:             Newline()\n\
~Shift<Key>osfActivate:            Enter()\n\
<KeyPress>Caps_Lock:               ShiftOn()\n\
~Shift<KeyRelease>Caps_Lock:       ShiftOff()\n\
<KeyPress>Shift_L:                 ShiftOn()\n\
~Lock<KeyRelease>Shift_L:          ShiftOff()\n\
<KeyPress>Shift_R:                 ShiftOn()\n\
~Lock<KeyRelease>Shift_R:          ShiftOff()\n\
Shift<Key>Tab:                     BackTab()\n\
~Shift<Key>Tab:                    Tab()\n\
Ctrl<Key>h:                        Home()\n\
 <Key>osfBeginLine:                Home()\n\
~Mod1~Shift<Key>osfLeft:           Left()\n\
Mod1~Shift<Key>osfLeft:             Left2()\n\
~Mod1Shift<Key>osfLeft:            Left2()\n\
~Mod1~Shift<Key>osfRight:          Right()\n\
Mod1~Shift<Key>osfRight:           Right2()\n\
~Mod1Shift<Key>osfRight:           Right2()\n\
<Key>osfUp:                        osfDown: Down()\n\Up()\n\
```

```
        <Key>osfDown:                   Down()\n\
        Shift<Key>osfDelete:            EraseEOF()\n\
        ~Shift<Key>osfDelete:           Delete()\n\
        <Key>osfBackSpace:              Left()\n\
        <Btn1Down>:                     MoveCursor()\n\
    Shift<Key>osfHelp:                  PF13()\n\
    Shift<Key>F2:                       PF14()\n\
    Shift<Key>F3:                       PF15()\n\
    Shift<Key>osfMenu:                  PF16()\n\
    Shift<Key>F5:                       PF17()\n\
    Shift<Key>F6:                       PF18()\n\
    Shift<Key>F7:                       PF19()\n\
    Shift<Key>F8:                       PF20()\n\
     Shift<Key>F9:                      PF21()\n\
    Shift<Key>osfMenuBar:               PF22()\n\
     Shift<Key>F11:                     PF23()\n\
    Shift<Key>F12:                      PF24()\n\
    <Key>osfHelp:                       PF1()\n\
    <Key>F2:                            PF2()\n\
    <Key>F3:                            PF3()\n\
    <Key>osfMenu:                       PF4()\n\
    <Key>F5:                            PF5()\n\
    <Key>F6:                            PF6()\n\
    <Key>F7:                            PF7()\n\
    <Key>F8:                            PF8()\n\
    <Key>F9:                            PF9()\n\
    <Key>osfMenuBar:                    PF10()\n\
    <Key>F11:                           PF11()\n\
    <Key>F12:                           PF12()\n\
    Shift<Key>osfCancel:                Reset()\n\
    ~Shift<Key>osfCancel:               Clear()\n\
    Mod1<Key>c:                         Clear()\n\
    Mod1<Key>r:                         Reset()\n\
    Mod1<Key>1:                         PA1()\n\
    Mod1<Key>2:                         PA2()\n\
```

```
Mod1<Key>i:                    Insert()\n\
<Key>osfInsert:                Insert()\n\
Mod1<Key>d:                    Delete()\n\
Mod1<Key>h:                    Home()\n\
Ctrl<Key>l:                    Redraw()\n\
    <Key>:                     Default()
```

Note that the left button may be used to move the cursor.

Mod1 is the Alt key on an IBM keyboard.

This is the complete list of actions:

l l l l Redraw Insert Enter AltCursor MonoCase Tab BackTab Reset Left Left2 Right Right2 Up Down Newline Dup Field-Mark PA1 PA2 SysReq Clear CursorSelect EraseEOF EraseInput Home Delete Default MoveCursor String ShiftOn ShiftOff PF1-PF24

MACROS

A simple macro facility is provided via the String action. The arguments to String() are one or more double-quoted strings which are inserted directly as if typed. Hex constants may be entered in the form 0xff, and the C backslash conventions are honored as follows:

l l. \b Left \f Clear \n Enter \r Newline \t Tab

An example action would be:

```
Mod1<Key>p: probs clearrdr\n")
```

Note: The strings are in ASCII and converted to EBCDIC, so beware of inserting control codes.

COLOR

hds3270 does not understand IBM 3270 color protocols. It merely draws text in one of three colors depending on the field attributes. Those colors may be customized using the following resources:

```
hds3270.normalColor:    green
hds3270.boldColor:  cyan
hds3270.inputColor: orange
hds3270.colorBackground: black
```

It is possible to get monochrome colors and behaviour using the -mono switch.

FILES

/usr/lib/X11/app-defaults/HDS3270

SEE ALSO

telnet(1), tn3270(1)

X Toolkit Intrinsics (R4 version)

AUTHORS

Robert Viduya - SunView based 3270tool

Jeff Sparkes - X11 port

Edward Parks - HDS ViewStation port

BUGS

Since the custom font is not substitutable, the program cannot be coerced into using a different sized font. This means that the window is essentially fixed in size. If somebody wishes to contribute different sized fonts, I'll welcome them.

Sun Release 4.1 Last change: February 16, 1993

2.3.3 Open Connect tn3270 Terminal Emulation

The HDS ViewStation also supports an IBM 3270 emulation provided by Open Connect Systems. This emulation client is available only with appropriate licensing from HDS Network Systems, Inc. A separate set of manuals and documentation are supplied with the OCtn3270 client.

The Open Connect tn3270 client provides some additional functionality for the 3270 interface, including X11 fonts, color support, and so on. However, these are capabilities not always used or supported by the IBM host. If you are using an Open Connect emulation on the host computer, you will want to use the HDS OCtn3270 client. If, on the other hand, you are using the normal 3270 interface, the HDS3270 client may provide a connection more like the IBM 3270 terminal.

The Open Connect tn3270 ViewStation client is available with a software license form HDS Network Systems, Inc. Separate software and documentation is provided.

2.3.4 Tandem 6530 Terminal Emulation

The Tandem 6530 terminal emulation is available with appropriate licensing from HDS Network Systems, Inc. A separate set of manuals and documentation are supplied with the Tandem 6530 client.

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.

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.

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.

2.4.3.4 Selecting a Window with MWM

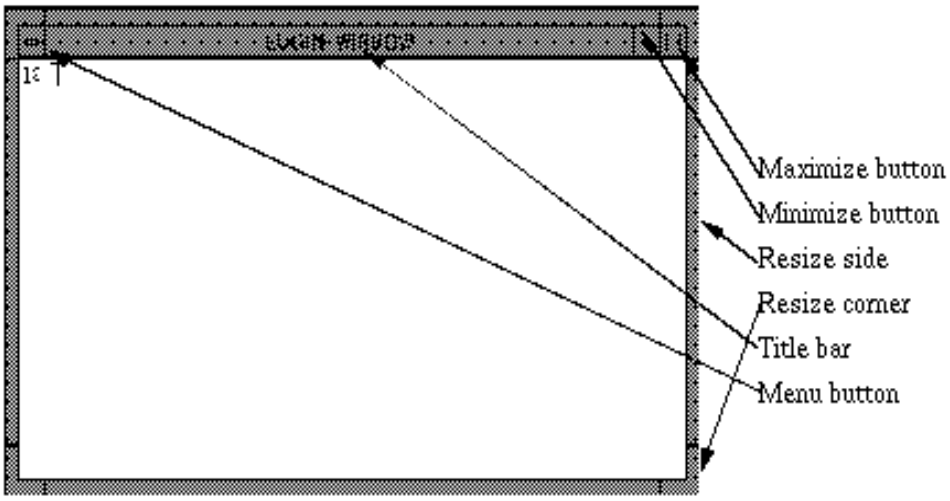
When you have many windows on the screen, you move from one to another by moving the mouse pointer and clicking (with the left button) on the window to select it. You will notice that the border of the window changes color or shade as an indication that it is the currently selected window. This is called 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.

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.

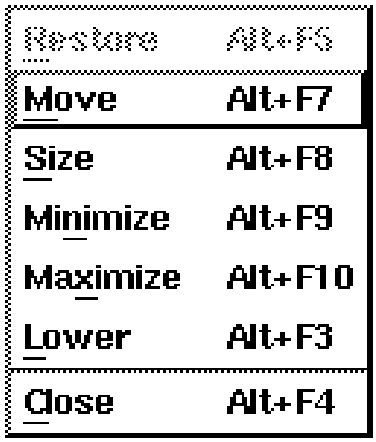
2.4.3.9 Maximizing a Window

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*

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.

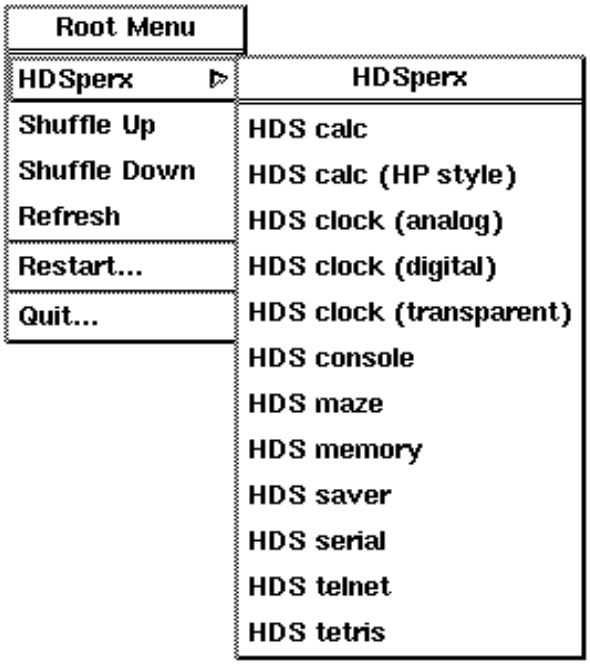
2.4.4 Using the MWM Root Menu

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

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.

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

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

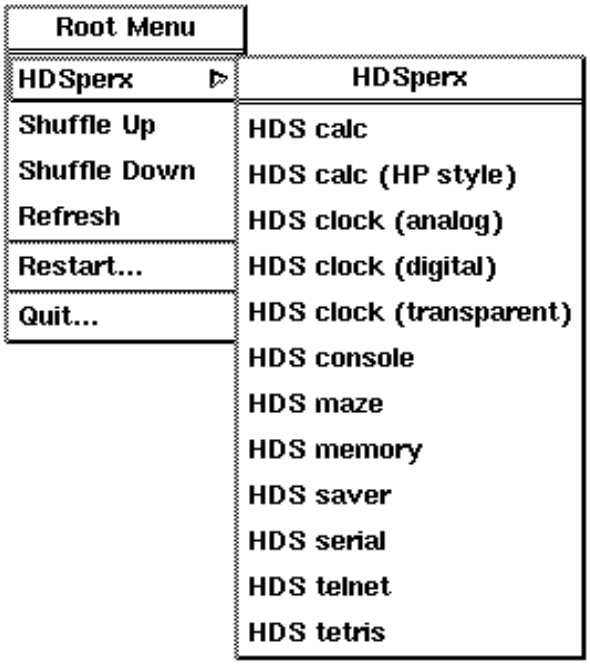
The details of making changes to this menu and setting it up manually are given later in this Section. Refer also to Section 2.1 on using the HDSperx clients.

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:

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
```

```
"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 "autoClient-Menu" 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
```

```
"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
#
```

```
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.

- 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.

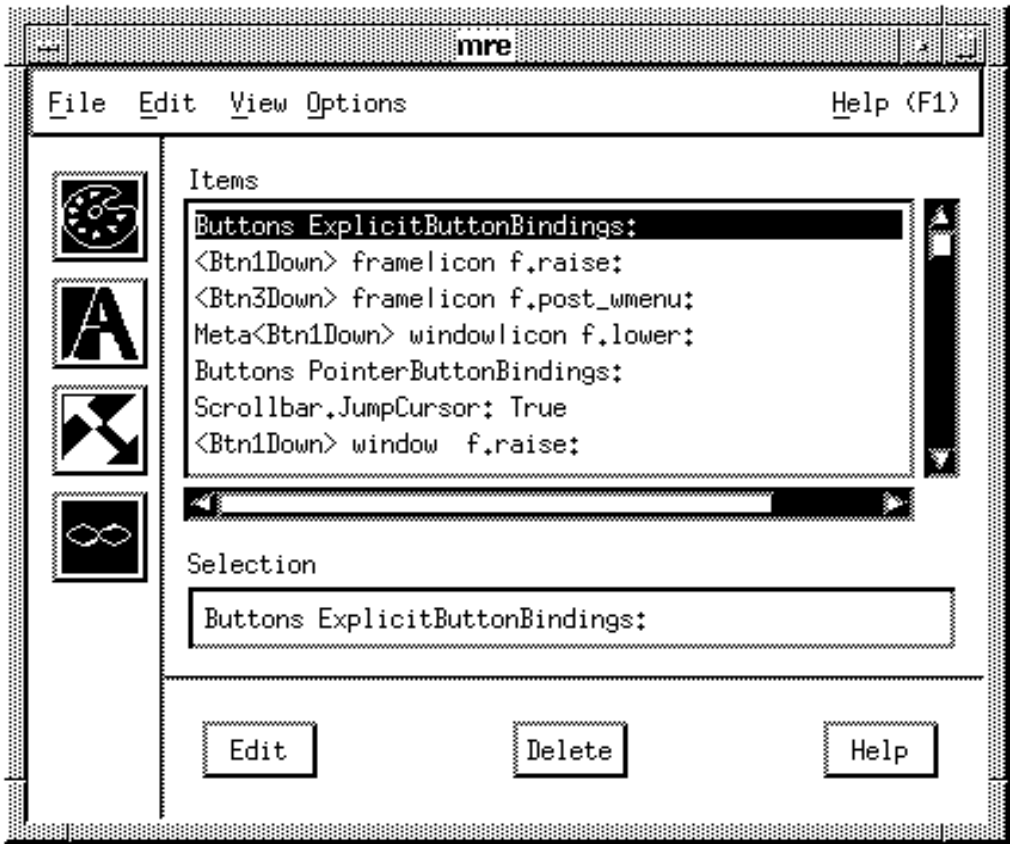
An example of the kind of problem you might encounter with conflicting settings or definitions of this type is with key assignments (bindings) for different clients. In the X world, the xmodmap utility is used to remap the keyboard, but this is a global setting and will apply to all clients and all windows. 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;

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.



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:

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
```

```
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.

```
#
```

```
#  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
}
```

```
#
# 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
```

```
}
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.

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,

.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.

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:

```
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.

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
```

OLVWM - OPEN LOOK Virtual Window Manager

This chapter describes the operation of the OPEN LOOK Virtual Window Manager. Some examples of customizations are also included.

Section 2.0 - HDS ViewStation Local Clients

2.5 OPEN LOOK Virtual Window Manager

The HDS ViewStation can download the OPEN LOOK Virtual Window Manager to run locally with its server code. Effective with ViewStation server code version 2.1, this OPEN LOOK Window manager is version 3.0 (replacing the version 1.0). OPEN LOOK version 3.0 supports the "virtual window manager" extension, which is described later. You can use either "OLWM" or "OLVWM" on your command lines to start olvwm; the terms are used interchangeably in this manual, though only OLVWM is actually run. If you don't want to use the Virtual window manager, you can resize its window to a single screen tile size.

Since the OPEN LOOK Virtual 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 OPEN LOOK Virtual Window Manager, but be careful with differences between OLWM and OLVWM versions.

This chapter gives a brief overview of the OPEN LOOK Virtual Window Manager. The OPEN LOOK Virtual Window Manager has the same functions as OLWM, but manages a virtual desktop that is larger than the actual screen. There are some simple exercises to help you learn what symbols perform which operations, and so on, but you should refer to individual OPEN LOOK reference materials for full details. The OLVWM man pages and supporting example files are found on the HDSware tape in */hds-fx/man*.

This chapter also contains sample files for OPEN LOOK resources and some suggestions for customizing the OPEN LOOK Virtual Window Manager for your system environment and your personal preferences. Again, this material is only

suggestive and you should refer to the OPEN LOOK reference materials for details.

2.5.1 Starting OPEN LOOK Virtual Window Manager

You can start the OPEN LOOK Virtual Window Manager by selecting the OPEN LOOK Window Manager icon 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 will automatically be controlled by the OPEN LOOK Virtual Window Manager.

If you use resource customizations or set OLWM environments, these must be loaded into the ViewStation server before the local OLVWM is started. Setting resources is discussed in Section 3.4 and some OLWM environment settings are discussed in the next section.

2.5.2 Starting Local OLVWM Remotely

Normally, you just click on the OPEN LOOK Virtual Window Manager option on the Setup Menu to start it. Many times it is more convenient to start the local OLVWM from an Xsession file or from a remote host login file. Starting the OLVWM from a remote host copies your customized window manager files, such as .openwin-menu, to the local window manager's operation.

The local ViewStation OPEN LOOK Virtual Window Manager can be started with the rsh (remote shell command) as follows:

rsh -n <*hostname*> <*olvwm*> [arguments] '&'

(you can use *olvwm* or *olwm*; both names are supported)

(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. 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 .openwin-menu, etc. should be read before the remote shell command is given.

When the window manager is started from a remote host, the ViewStation attempts to read any applicable resource files, such as, /usr/lib/X11/app-defaults/olwm, /usr/home/username/.Xdefaults, /usr/home/username/.openwin-menu, and so on with the **rcp** (remote copy) command. 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.

2.5.3 The OPEN LOOK Environment

OPEN LOOK GUI (Graphical User Interface) has been designed as a total environment, including its own application programs, editors, and so on. These things are not automatically included with the local OLWM, since they are not local functions but run remotely on the host computer. These application programs are available on your host machine, typically a SUN machine, but they require that you set a number of paths and environments so they can be located and used.

The following is a simple shell script to set those environments. You might label this script do.olwm and put it in your home directory. If you are using the local window manager to start remote processes, these settings must be in your .cshrc file. When you start a session, you could start OLWM by typing the do.olwm command, or you could have it executed automatically by xdm, the X display manager.

This sample script starts the OLWM remotely and contains some environment settings and the command to start the window manager (if you want to set these things for the local OLWM, you should enter them in your .cshrc file in your home directory):

```
setenv OPENWINHOME  /usr/openwin

setenv LD_LIBRARY_PATH  $OPENWINHOME/lib

setenv MANPATH  $OPENWINHOME/share/man:/usr/man

set path=( $OPENWIN/bin  $OPENWINHOME/bin/xview  $path )

echo "Starting OPEN LOOK Virtual Window Manager"

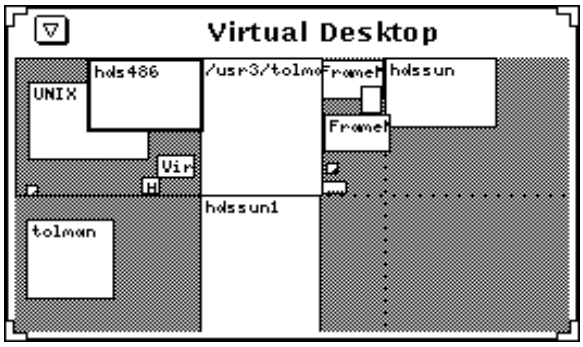
echo "OLWM DISPLAY = $DISPLAY"

$OPENWINHOME/bin/olvwm &
```

This example uses simple settings and "default" paths and directories; your installation may use different locations and directories. This shell script must be executable and should be in your home directory.

2.5.4 OPEN LOOK Virtual Window Manager Operations

The OPEN LOOK Virtual Window Manager is an extension of the basic OPEN LOOK window manager. Its operations are the same. The only difference is that the Virtual window manager works on a desktop, that is, a virtual screen that is larger than the actual screen size. By default, its virtual screen is six screens arranged in a tiled pattern. A small window appears in the upper left corner that shows these six screens with small representations of the windows currently displayed in each screen tile. Where possible, the names of the windows are shown, though the names are often truncated.



You can move a window from one screen tile to another by clicking and dragging it, or by starting it in that window using your window manager's menus. For instance, you might want to keep your xterm and Setup windows in screen tile 1, your database windows in screen tile 2, and your Framemaker documents in screen tile 3, or whatever arrangement you want. You can move from one screen to another by moving to the Virtual window manager window (which appears in all screen windows) and double-clicking on the screen tile you want.

The Virtual Desktop window can be resized to increase or decrease the number of screen tiles. It is a convenient way to organize your activities and manage a large number of clients and windows. If you want to use OLWM (not using the Virtual Desktop feature), you can use an OPEN LOOK resource to restrict the Virtual Desktop to a 1x1 tile. The OPEN LOOK resource is:

```
OpenWindows.VirtualDesktop: 1x1        (default is 3x2)
```

Use this resource in your .Xresource file or in any of your resource files. Note that OPEN LOOK resources may use either:

```
OpenWindows.resource: value
```

or

```
olvwm.resource: value
```

Both resource names are accepted but the OpenWindows syntax is preferred if the resource applies to Open Windows clients other than the window manager.

2.5.5 Using the OLVWM Window Manager

The OPEN LOOK Virtual 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.

The HDS ViewStation has the OPEN LOOK Virtual Window Manager running as a local client. This should not be confused with the OPEN LOOK environment which was designed to use an entire range of functions, including the File Manager program, editing capabilities, and so on. These functions can be used, but they must be added as part of your host environment before you can use them from within your window manager.

2.5.5.1 Pointer Shapes

The mouse cursor, called the Pointer, indicates the kind of operation you are currently using. In OPEN LOOK, the pointer is an arrow. There are other pointer shapes to give you current information, like a watch cursor when a process is busy, or a question mark when a selection isn't appropriate for that situation.

2.5.5.2 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 1.8.3.1 of this manual.)

OPEN LOOK has made some default assignments for the mouse buttons. The three button mouse has these assignments for its buttons:

Left - SELECT - this selects objects or controls

Middle - ADJUST - this extends or reduces the number of selected objects

Right - MENU - this displays and chooses menus

2.5.5.3 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. For instance, OPEN LOOK uses the Caps_Lock key to invoke micropositioning for the mouse, so you will see much slower mouse tracking.

2.5.5.4 Selecting a Window with OLVWM

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 OPEN LOOK Virtual 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; general descriptions of resource handling are in Section 3.4. 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. These questions of configuration are important and are discussed at the end of this chapter.

2.5.5.5 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.

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

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

2.5.5.6 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 or in one of the window borders 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. When the position of the window is correct, simply release the mouse button and the window appears at its new location.

2.5.5.7 Resizing a Window

You can change a window's size to make your work easier. Imagine that the window has elastic sides. By grabbing a corner (point the mouse on 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 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.

2.5.5.8 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 Iconify 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.

2.5.6 OLVWM Menu

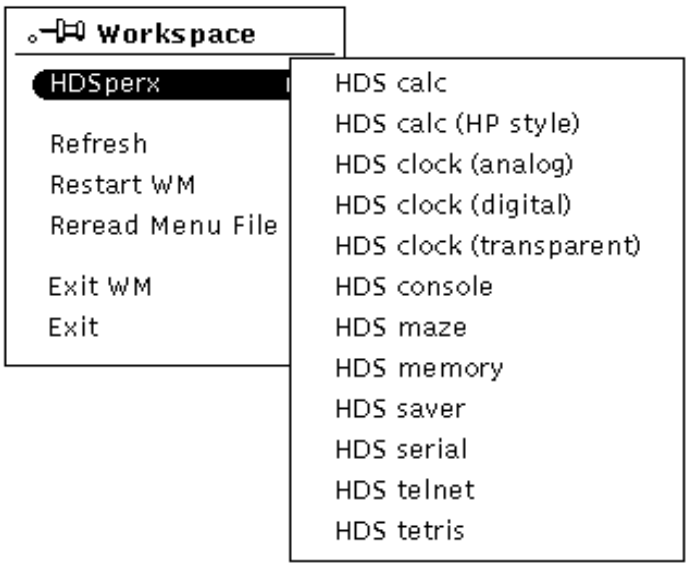
Pressing the right mouse button brings up the OLWM Menu. It allows you to refresh the screen, to restart the window manager, to reread the Menu file, to exit the window manager, or to exit your entire session. The menu is shown below:



Normal operation of pull-down menus is that they disappear as soon as you release the mouse button that brought them up. You'll notice a little push pin in the top left corner of each menu along with a small hole next to it. If you move the push pin over to the hole, using the mouse pointer, it will pop into the hole and pin the menu onto the screen permanently (or at least until you pull it out of the hole). Having the menu constantly visible is convenient if you use its operations frequently.

2.5.7 Using HDSperx with OLVWM

The HDSperx menu is added to your root menu automatically. This allows you to start HDSperx clients easily. The menu is shown below.



2.5.8 Exiting OLVWM

You can exit the OPEN LOOK Virtual Window Manager by selecting the "Exit WM" line in the menu. Clicking on the last line "Exit" closes all of your windows and connections.

2.5.9 Starting Local Processes from the Local OLVWM

When you use the ViewStation's local OLVWM window manager, a submenu for the ViewStation's HDSperx clients is automatically added to your root menu. This 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 at the end of this chapter. Refer also to Section 2.1 on using the HDSperx clients.

2.5.10 Starting Remote Processes with Local OLVWM

The ViewStation runs OLVWM locally on the server. The OPEN LOOK Virtual Window manager may be started from a remote host using the **rsh** (BSD Unix) or **rcmd** (System V Unix) command as described in the previous section. Starting processes on other hosts is done in the same way as with a remote window manager, using the appropriate configuration file. The sample OPEN LOOK configuration files for the window manager menus below show the process. This file resides in the user's home directory.

Note that the OPEN LOOK environment variables and paths must be in the user's .cshrc or .profile file, since it is that shell file (not your .login file) that is read when a remote shell process is started. Having these variables in another location, such as the .login file or in a separate shell script, generally means that the remote shell process fails.

2.5.11 Customizing OLVWM

You can use a number of OLWM resources to customize the appearance and operation of OLWM. A complete list of the OLWM resources can be found in the OLWM man pages.

Note that these resources must be loaded into the ViewStation (usually with the **xrdb** utility) before the local OLVWM is started. These examples show some simple OLVWM resource settings:

```
# Mouse focus policy - either followmouse or select

OpenWindows.SetInput:     followmouse

# Select the window colors

OpenWindows.WindowColor:     Pale Turquoise

OpenWindows.WorkspaceColor:     White

# Virtual Desktop size

OpenWindows.VirtualDesktop: 1x1
```

In this example, the SetInput resource sets the mouse focus policy. The "select" value (note the lowercase) means that you click in a window to give it focus; the "followmouse" value means that focus follows the mouse position.

If you change the resource file, you must reload the resources (check your currently set resources with the appres program described in Section 3.4), then exit and restart OLVWM for the settings to take effect.

2.5.12 OPEN LOOK Window Manager Menu

The menu for the OPEN LOOK Window Manager is a file called "openwin-menu", which is found in the *./openwin/lib* directory. There are a number of sample and submenu files there. Look at them to see their structure and syntax.

The openwin-menu file contains:

```
#

# @(#)openwin-menu 23.15 91/09/14 openwin-menu

#

# OpenWindows default root menu file - top level menu

#

"Workspace"       TITLE

"Programs"        MENU $OPENWINHOME/lib/openwin-menu-pro-
grams

"Utilities"       MENU $OPENWINHOME/lib/openwin-menu-utili-
ties

"Properties..."   PROPERTIES

SEPARATOR

"Help..."         exec $OPENWINHOME/bin/helpopen handbooks/
top.toc.handbook

"Desktop Intro..." exec $OPENWINHOME/bin/helpopen handbooks/
desktop.intro.handbook

SEPARATOR

"Exit..."         EXIT
```

The menu file consists of a label (which appears on the menu) and the command that it executes. Note the use of submenus (there are other submenus, such as the "demo" entries, in the *./openwin/lib* directory that are not used). Note also the use of "SEPARATOR" entries to provide visual organization for the menu.

Modifying the Menu

If you want to modify this menu, just open the file with a text editor and make the desired changes. You may want to delete the "Desktop Intro..." entry, or add or delete some utility programs to the "Utilities" menu (this menu is in its own file,

openwin-menu-utilities, in the same directory). Just make these entries and save the modified file. It is good practice to save a copy of the original file under another name.

2.5.12.1 Starting Clients with OLWM

The OPEN LOOK Virtual Window Manager provides a facility allowing the user to start processes from window manager menus as defined in an .openwin-menu configuration file. The sample file shown in the next section gives some simple examples. Details are available from the OLWM manual pages, including specifications for labeling and syntax; refer also to the .olvwmrc configuration file.

There are three things to note about the openwin-menu file:

1) You must start the ViewStation local window manager from a remote host in order to use these resource specification files for the local window manager. This remote host and the 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.

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 initiating host. If you are using an *.rhosts* file, the appropriate permissions must be entered there as well. Refer to Section 1.11 for more details.

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.

3) Different versions of OLWM (versions 1.0, 2.0, and 3.0 are generally available) use different syntax for this file. For example, the PIN and DEFAULT entries in the file may have to appear at the beginning of an entry or at the end, depending on your version. Further, the TITLE line may require the fields to be reversed, that is: "Workspace" - TITLE. Check your version and its manual pages for the correct usage. If the syntax in this file is not correct, the menus will not be read and the process will fail.

If you have problems starting remote processes, look in the Console window for error messages and consult the checklist for remote starting 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 or the *.rhosts* files on each machine for a ViewStation entry.

2.5.13 Using HDSperx Clients with OLWM

The HDSperx clients 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 olvwm resources:

olvwm*autoClientMenu: True

olvwm*autoClientMenuName: HDSMENU

These are both olvwm resources and should be used with other olvwm resources in .Xdefaults or .olvwmrc. (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 "olvwm*autoClient-Menu: 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 OLVWM has an internally defined menu called "HDSMENU" which can be explicitly referenced in your .openwin-menu file menu table.

2.5.13.1 OLWM .openwin-menu Sample File

This is a sample .openwin-menu file. Use it as a reference when you construct your own menu file.

```
#
# openwin-menu - OpenWindows X11/NeWS Server default root menu file.
"Workspace" TITLE
"Programs" MENU
 "HDS tools"            HDSMENU
 "small xterm"          exec /usr/bin/X11/xterm -fn 8x13b
 "xterm"                exec /usr/bin/X11/xterm -geom 96x35
 "big xterm"            exec /usr/bin/X11/xterm -fn 12x24
 "xload"                exec /usr/bin/X11/xload
 "Text Editor..."       exec $OPENWINHOME/bin/xview/textedit
 "File Manager..."      exec $OPENWINHOME/bin/xview/filemgr
 "Mail Tool..."         exec $OPENWINHOME/bin/xview/mailtool
 "Calendar Manager..."  exec $OPENWINHOME/bin/xview/cm
 "Clock..."             exec $OPENWINHOME/bin/xview/clock
 "Calculator..."        exec $OPENWINHOME/bin/xview/calctool
```

```
 "Print Tool..."          exec $OPENWINHOME/bin/xview/printtool
"Programs" END PIN

"Utilities" MENU
 "Refresh" DEFAULT        REFRESH
 "Reset Input"            POSTSCRIPT /resetinput ClassUI send
 "Save Workspace"         SAVE_WORKSPACE
 "Lock Screen"            exec xlock
 "Console..."             exec $OPENWINHOME/bin/xview/cmdtool -C
"Utilities" END

"Properties..."           PROPERTIES

"Restart..."              RESTART

"Exit WM..."              WMEXIT

"Exit..."                 EXIT
```

The OLVWM window manager can also start HDSperx clients explicitly, without using HDSMENU. For example, you can start a local HDSdclock with an .openwin-menu file containing menu entries:

```
# OPEN LOOK Virtual Window Manager Menu
"Workspace" TITLE
"Winmenu..."                    WINMENU
"Programs" MENU
   "XTerm..."DEFAULT            exec xterm
"Programs"                      ENDPIN

"Calculator"                    exec xcalc
"Clock"                         exec hdslocal hdsdclock
"HDS Local Clients..."          HDSMENU

"Properties..."                 PROPERTIES
"New Menu..."                   REREAD_MENU_FILE
"Restart..."                    RESTART
```

"Quit..."	WMEXIT
"Exit..."	EXIT

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, so olvwm responds to the new settings.

2.5.13.2 HDSperx Menu Contents

The HDSperx menu is a list of the ViewStation's local clients that you have selected in the ViewStation's *clients.config* file. You can add or delete clients from this menu by uncommenting or commenting their client menuName lines in the *clients.config* file.

Note that you must specify HDSperx processes with a "hdslocal" prefix to indicate that they should run on the ViewStation. In the example above, the xcalc will run on the host computer, but the hdsdclock will run on the ViewStation.

2.5.13.3 HDSperx Menu Handling

The menu itself is handled differently depending on your window manager startup procedure.

- **ViewStation window manager with remote start** - If you start the ViewStation's OPEN LOOK window manager using a remote shell command from the host computer, you get the OLWM menu specified in your *.openwin-menu* file with the addition of the HDSperx menu for the ViewStation's local clients. This merging of the menus is done automatically by the ViewStation.
  This is the best option, since it includes all your menus. You can use the remote shell startup mechanism from an *.xsession* file or from your *.profile*, which are read as you login. This is discussed in detail below.
- **ViewStation window manager with local start** - If you start the OPEN LOOK-window manager by clicking on the ViewStation's OLWM icon (or the Autostart button) on the Main Menu of Setup Mode, your OLWM root menu is the HDSperx menu (along with refresh, exit, and other commands). The customization options described here don't apply; you must use the rsh method of starting to get them.
- **Host-based window manager** - If you use the host to run the OPEN LOOK window manager, you get the OLWM menus specified in the *.openwin-menu* file. You could manually add the HDSperx menu to this file, but you would have to make sure the "exec" command syntax was correct for the ViewStation's local clients.

2.5.14 Window Manager Remote Starting Checklist

Starting the ViewStation's window managers from a remote host with the "rsh" command allows the local ViewStation 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.

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 your 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.

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. A .login file is not a good choice for these variables.

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, .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 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, or the .rhosts files if they are used.

■ 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.

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 from 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 hosts on the network to test its operation independent of the ViewStation 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, or 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.

2.5.15 OLWM Version 1.0 Syntax Differences

This note discusses the syntax used with OLWM version 1.0 which was used with ViewStation server code prior to version 2.1. Note the differences and modify your menu configuration files appropriately. You can refer to the OLVWM version 3.0 manual pages included on the FX Utility tape for complete details.

The OLWM version 1.0 allows the user to start processes from window manager menus as defined in an .olwmmenu configuration file. The sample file below shows some simple examples. Details are available from the OLWM manual pages, including specifications for labeling and syntax.

```
# OLWM Menu items
# Note: this menu refers to OLWM version 1.0

TITLE                           "Workspace"
PIN

DEFAULT "Programs"              Menu
    DEFAULT "xterm"             "xterm -fn 10x20 -g 80x32 &"
    "Text Editor..."            textedit
    "File Manager..."           filemgr
    "Mail Tool..."              mailtool
END

"Utilities" MENU
    DEFAULT "Refresh"           REFRESH
    "Save Workspace"            SAVE_WORKSPACE
    "Lock Screen"               xlock
END

"Properties..."                 PROPERTIES

"WM Exit..."                    WMEXIT
"Exit..."                       EXIT
```

There are three things to note:

1) You must start the ViewStation local window manager from a remote host in order to use these resource specification files for the local window manager. This remote host and the 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.

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 initiating 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.

3) Different versions of OLWM (versions 1.0, 2.0, and 3.0 are generally available) use different syntax for this file. For example, the PIN and DEFAULT entries in the file may have to appear at the beginning of an entry or at the end, depending on your version. Further, the TITLE line may require the fields to be reversed, that is: "Workspace" - TITLE. Check your version and its manual pages for the correct usage. If the syntax in this file is not correct, the menus will not be read and the process will fail.

If you have problems starting remote processes, look in the Console window for error messages and consult the checklist for remote starting 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.

2.5.16 HDS OLVWM Extensions

The HDS Network Systems's ViewStation version of **olvwm** adds the following resources to support ViewStation specific extensions.

`AutoClientMenu` *`(boolean)`*

Indicates whether **olwm** should automatically add the HDSperx menu to the top of the Workspace Menu. The HDSperx menu lists tools which can run locally on the ViewStation.

Default value: true.

`AutoClientMenuName` *`(string)`*

Specifies the name of the HDSperx menu to be used if AutoClientMenu is true.

Default value: HDSperx.

HDS **olvwm** also adds a new Workspace menu customization keyword: HDS-MENU, which presents a menu of HDS ViewStation downloadable clients.

2.5.17 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 one, 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.

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 'olwm' in the .xsession file, replace the line

```
olwm -option1 -option2 -option3
```

with

```
hdsrunwm olwm -option1 -option2 -option3
```

HDSperx Clients

This chapter describes the HDSperx local clients for the ViewStation.

Section 2.0 - HDS ViewStation Local Clients

2.6 HDSperx Clients

The ViewStation can run X clients locally because of the power of its processor and its Unix micro-kernel architecture. HDSware supplies a number of these clients. You can also compile your own clients to run on the ViewStation with the HDS ViewStation Developer's Kit, a set of software development tools. The Developer's kit is described in Section 5.0.

2.6.1 HDSperx Manual Pages

This note contains printed copies of the manual pages for each of the HDSperx clients. These manual pages are available on the HDSware tape as nroff files in the *./hds-fx/man* directory. This means that you can add these nroff files to your other man pages and have help for the HDSperx clients available with your normal man utility. If you want to see the file as text output, you can use the command:

nroff -man <filename.1>

The man pages contain all the command line arguments and resources for the HDSperx clients. Man pages for the maze and ico clients are not reprinted here since they are identical to the man pages available on X systems.

The following clients have printed man pages in separate sections of this manual:

HDS Telnet - HDSterm, a VT320 terminal emulator. This client is described fully in Section 2.2, with the man page in Section 2.2.4..

HDS Serial - HDSterm, a VT320 terminal emulator. This client is described fully in Section 2.2.

HDS Memory - a client that reports current available memory, fragments, and largest available fragment of ViewStation memory. Its operation is described in Section 1.5.5.4.

HDS Console Window - a ViewStation shell and Diagnostic message window which uses HDSterm as its interface. This client is described in Section 1.2.2

MWM - Motif Window Manager. This client is described in Section 2.4. Its full OSF man pages are found on the HDSware tape in the */hds-fx/man* directory.

OLVWM - OPEN LOOK Virtual Window Manager. This client is described in Section 2.5. Its complete man pages, including sample files, are found on the HDSware tape in the */hds-fx/man* directory.

FVWM - FVWM Virtual Window Manager. This client and its accompanying modules is described in Section 2.8.

HDS3270 - IBM 3270 terminal emulator. This client is described in Section 2.3.2.

Multimedia Clients - These audio and video clients have a separate section describing their operation and containing their man pages; see Section 2.7.

The Console Window displays a current list of HDSperx clients with the "list" command. Many of these clients are common UNIX or X clients and do not have individual man pages in this manual.

2.6.2 HDScalc Manual Pages

This is the HDScalc man page.

NAME

hdscalc - scientific calculator for X

SYNOPSIS

hdscalc [-stipple] [-rpn] [-toolkitoption...]

DESCRIPTION

hdscalc is a scientific calculator desktop accessory that can emulate a TI-30 or an HP-10C. It is based on the X11 Release5 **xcalc**, but has been rewritten with added customizability and functionality using the Motif toolkit.

OPTIONS

hdscalc accepts all of the standard toolkit command line options along with two additional options:

-stipple

This option indicates that the background of the calculator should be drawn using a stipple of the foreground and background colors. On monochrome displays this improves the appearance.

-rpn

This option indicates that Reverse Polish Notation should be used. In this mode the calculator will look and behave like an HP-10C. Without this flag, it will emulate a TI-30.

OPERATION

Pointer Usage: Operations may be performed with pointer button 1, or in some cases, with the keyboard. Many common calculator operations have keyboard accelerators. To quit, press pointer button 3 on the display of the calculator.

Calculator Key Usage (TI mode): The numbered keys, the +/- key, and the +, -, *, /, and = keys all do exactly what you would expect them to. It should be noted that the operators obey the standard rules of precedence. Thus, entering "3+4*5=" results in "23", not "35". The parentheses can be used to override this. For example, "(1+2+3)*(4+5+6)=" results in "6*15=90".

The entire number in the calculator display can be selected, in order to paste the result of a calculation into text. A number may also be pasted into the display.

The following is a list of all calculator functions available for use in defining a custom calculator. In departure from the normal scheme of only being able to specify actions in a resource file, this program will also allow callbacks to be specified for any calculator button via the user data resource for that button. The calculator function for any button may therefore be specified as a translation for that button or as the user data for that button, and it would be used as an action or callback respectively. Callbacks are being used so that the translations that implement default Motif button behavior are not affected by adding the calcula-

tor function as a translation to that button. Each calculator function has both a translation and a callback with the same name. The only exception is for the digit keys where the action used is digit(n), where n is the corresponding digit, 0..9 while the callbacks are digit0, digit1, ..., digit9. Note that when used as an action the name is followed by () whereas when use as a callback only the name is needed.

1/x

Replaces the number in the display with its reciprocal. The corresponding action procedure is reciprocal().

x^2

Squares the number in the display. The corresponding action procedure is square().

SQRT

Takes the square root of the number in the display. The corresponding action procedure is squareRoot().

CE/C

When pressed once, clears the number in the display without clearing the state of the machine. Allows you to re-enter a number if you make a mistake. Pressing it twice clears the state, also. The corresponding action procedure for TI mode is clear().

AC

Clears the display, the state, and the memory. The action procedure to clear the state is off().

INV

Invert function. See the individual function keys for details. The corresponding action procedure is inverse().

sin

Computes the sine of the number in the display, as interpreted by the current DRG mode (see DRG, below). If inverted, it computes the arcsine. The corresponding action procedure is sine().

cos

Computes the cosine, or arccosine when inverted. The corresponding action procedure is cosine().

tan

Computes the tangent, or arctangent when inverted. The corresponding action procedure is tangent().

DRG

Changes the DRG mode, as indicated by 'DEG', 'RAD', or 'GRAD' at the bottom of of the calculator "liquid crystal" display. When in 'DEG' mode, numbers in the display are taken as being degrees. In 'RAD' mode, numbers are in

radians, and in 'GRAD' mode, numbers are in grads. When inverted, the DRG key has a feature of converting degrees to radians to grads and vice-versa. Example: put the calculator into 'DEG' mode, and enter "45 INV DRG". The display should now show something along the lines of ".785398", which is 45 degrees converted to radians. The corresponding action procedure is degree().

e

The constant 'e'. (2.7182818...). The corresponding action procedure is e().

EE

Used for entering exponential numbers. For example, to get "-2.3E-4" you'd enter "2 . 3 +/- EE 4 +/-". The corresponding action procedure is scientific().

log

Calculates the log (base 10) of the number in the display. When inverted, it raises "10.0" to the number in the display. For example, entering "3 INV log" should result in "1000". The corresponding action procedure is logarithm().

ln

Calculates the log (base e) of the number in the display. When inverted, it raises "e" to the number in the display. For example, entering "e ln" should result in "1". The corresponding action procedure is naturalLog().

y^x

Raises the number on the left to the power of the number on the right. For example "2 y^x 3 =" results in "8", which is 2^3. For a further example, "(1+2+3) y^x (1+2) =" equals "6 y^x 3" which equals "216". The corresponding action procedure is power().

PI

The constant 'pi'. (3.1415927....) The corresponding action procedure is pi().

x!

Computes the factorial of the number in the display. The number in the display must be an integer in the range 0-500, though, depending on your math library, it might overflow long before that. The corresponding action procedure is factorial().

(

Left parenthesis. The corresponding action procedure for TI calculators is leftParen().

)

Right parenthesis. The corresponding action procedure for TI calculators is rightParen().

/

Division. The corresponding action procedure is divide().

*****

Multiplication. The corresponding action procedure is multiply().

-

Subtraction. The corresponding action procedure is subtract().

+

Addition. The corresponding action procedure is add().

=

Perform calculation. The TI-specific action procedure is equal().

STO

Copies the number in the display to the memory location. The corresponding action procedure is store().

RCL

Copies the number from the memory location to the display. The corresponding action procedure is recall().

SUM

Adds the number in the display to the number in the memory location. The corresponding action procedure is sum().

EXC

Swaps the number in the display with the number in the memory location. The corresponding action procedure for the TI calculator is exchange().

+/-

Negate; change sign. The corresponding action procedure is negate().

.

Decimal point. The action procedure is decimal().

Calculator Key Usage (RPN mode): The number keys, CHS (change sign), +, -, *, /, and ENTR keys all do exactly what you would expect them to do. Many of the remaining keys are the same as in TI mode. The differences are detailed below. The action procedure for the ENTR key is enter().

<-

This is a backspace key that can be used if you make a mistake while entering a number. It will erase digits from the display. (See BUGS). Inverse backspace will clear the X register. The corresponding action procedure is back().

ON

Clears the display, the state, and the memory. To clear state, the action procedure is off.

INV

Inverts the meaning of the function keys. This would be the f key on an HP calculator, but xcalc does not display multiple legends on each key. See the individual function keys for details.

10^x

Raises "10.0" to the number in the top of the stack. When inverted, it calculates the log (base 10) of the number in the display. The corresponding action procedure is tenpower().

e^x

Raises "e" to the number in the top of the stack. When inverted, it calculates the log (base e) of the number in the display. The action procedure is epower().

STO

Copies the number in the top of the stack to a memory location. There are 10 memory locations. The desired memory is specified by following this key with a digit key.

RCL

Pushes the number from the specified memory location onto the stack.

SUM

Adds the number on top of the stack to the number in the specified memory location.

x:y

Exchanges the numbers in the top two stack positions, the X and Y registers. The corresponding action procedure is XexchangeY().

R v

Rolls the stack downward. When inverted, it rolls the stack upward. The corresponding action procedure is roll().

blank

These keys were used for programming functions on the HP-10C. Their functionality has not been duplicated in xcalc.

Finally, there are four additional action procedures/callbacks: bell(), which rings the bell; selection(), which performs a cut on the entire number in the calculator's ''liquid crystal'' display; paste(), which pastes the current select buffer into the display if it can be interpreted as a number; and quit(), which turns off the calculator (exits the program).

ACCELERATORS

Accelerators are shortcuts for entering commands. xcalc provides some sample keyboard accelerators; also users can customize accelerators. The numeric keypad accelerators provided by xcalc should be intuitively correct. The accelerators defined by xcalc on the main keyboard are given below:

TI Key	HP Key	Keyboard Accelerator	TI Function	HP Function
SQRT	SQRT	r	squareRoot()	squareRoot()
AC	ON	space	clear()	clear()
AC	<-	Delete	clear()	back()
AC	<-	Backspace	clear()	back()
AC	<-	Control-H	clear()	back()
AC		Clear	clear()	
		q	quit()	quit()
		Control-C	quit()	quit()
INV	i	i	inverse()	inverse()
sin	s	s	sine()	sine()
cos	c	c	cosine()	cosine()
tan	t	t	tangent()	tangent()
DRG	DRG	d	degree()	degree()
e		e	e()	
ln	ln	l	naturalLog()	naturalLog()
y^x	y^x	^	power()	power()
PI	PI	p	pi()	pi()
x!	x!	!	factorial()	factorial()
(		(	leftParen()	
)		)	rightParen()	
/	/	/	divide()	divide()
*	*	*	multiply()	multiply()
-	-	-	subtract()	subtract()
+	+	+	add()	add()
=		=	equal()	
0..9	0..9	0..9	digit()	digit()
.	.	.	decimal()	decimal()
+/-	CHS	n	negate()	negate()
	x:y	x		XexchangeY()
	ENTR	Return		enter()
	ENTR	Linefeed		enter()

CUSTOMIZATION

The application class name is XCalc.

hdscalc has an enormous application defaults file which specifies the position, label, and function of each key on the calculator. It also gives translations to serve

as keyboard accelerators. Because these resources are not specified in the source code, you can create a customized calculator by writing a private application defaults file, using the Motif PushButton and Form widget resources to specify the attachments of the buttons (determining relative position and size), the label for each button, and the function of each button.

The foreground and background colors of each calculator key can be individually specified. For the TI calculator, a classical color resource specification might be:

XCalc.ti.XmPushButton.background: gray50

XCalc.ti.XmPushButton.foreground: white

For each of buttons 20, 25, 30, 35, and 40, specify:

XCalc.ti.button20.background: black

XCalc.ti.button20.foreground: white

For each of buttons 22, 23, 24, 27, 28, 29, 32, 33, 34, 37, 38, and 39:

XCalc.ti.button22.background: white

XCalc.ti.button22.foreground: black

WIDGET HIERARCHY

In order to specify resources, it is useful to know the hierarchy of the widgets which compose xcalc. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XCalc xcalc
    XmForm ti or hp   (the name depends on the mode)
      XmFrame bevel
        XmForm screen
          XmLabel M
          XmToggleButton LCD
          XmLabel INV
          XmLabel DEG
          XmLabel RAD
          XmLabel GRAD
          XmLabel P
      XmPushButton button1
      XmPushButton button2
      XmPushButton button3
  and so on, ...
      XmPushButton button38
```

XmPushButton button39

XmPushButton button40

APPLICATION RESOURCES

rpn (Class Rpn)

Specifies that the rpn mode should be used. The default is TI mode.

stipple (Class Stipple)

Indicates that the background should be stippled. The default is ''on'' for monochrome displays, and ''off'' for color displays.

cursor (Class Cursor)

The name of the symbol used to represent the pointer. The default is ''hand2''.

SEE ALSO

X(1), xrdb(1), the Motif Widget Set

COPYRIGHT

Copyright 1993, Human Designed Systems, Inc. Adapted from xcalc, copyright 1988, 1989, Massachusetts Institute of Technology.

AUTHORS

William Pohlmann (Human Designed Systems)

xcalc by

John Bradley, University of Pennsylvania

Mark Rosenstein, MIT Project Athena

Donna Converse, MIT X Consortium

X Version 11 Last change: 1/6/93

2.6.3 HDS Chooser Client

This is the HDS Chooser man page.

NAME

HDS Chooser - XDMCP host menu

SYNOPSIS

Only available by setting run local chooser to yes in the XDMCP options in setup when the xdmcp access mode is broadcast or indirect.

DESCRIPTION

The HDS Chooser program presents a list of all hosts which are willing to manage the session. The user may connect to any of the hosts presented by selecting the host and pressing the Connect button. A host may be selected via the mouse or by typing the desired selection into the selection box. If a host is typed into the selection box which is not listed, pressing the Connect button will result in a warning beep and the connection will not be attempted.

The list will grow as any new hosts respond that they are willing to manage. The chooser will send out the request with an exponential backoff algorithm so that there will be no network traffic when the chooser is up and the display is idle. The retransmissions start at 2 seconds and back off by factors of 2 to 32 seconds and discontinue after 126 seconds. The transmissions will restart after any keyboard or mouse activity.

The user may also press the Refresh button to discard the current list of willing hosts and begin collecting again. The refresh button is most useful when returning to a display that has been idle for some time. Since the list was compiled over time a host which in the past was willing to manage may no longer be. While the user will be informed of any new hosts that are willing to manage the moment the mouse is touched, hosts which are no longer willing are never removed from the list. By discarding the list and starting again with the refresh button, the list will be made current.

If a host is chosen that is no longer willing or is no longer on the network, the chooser will restart. There could be a 126 second delay before the chooser is run if the host was removed from the network.

Unlike the chooser provided with X11 Release 5 this chooser can work both with X11R4 xdm and X11R5 xdm hosts since it does not work via a forwarded request.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose HDS Chooser. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

Chooser chooser

XmSelectionBox selection

SEE ALSO

X(1) and xdm(1)

COPYRIGHT

Copyright 1993, Human Designed Systems, Inc.

AUTHORS

William Pohlmann(Human Designed Systems)

X Version 11 Last change: 1/7/93

2.6.4 HDSdclock Manual Page

This is the HDS digital clock man page.

NAME

hdsdclock - digital clock for X

SYNOPSIS

hdsdclock [-*toolkitoption* ...] [-option ...]

DESCRIPTION

The *hdsdclock* program displays the time in digital format only. The time is updated on a per second basis or on a per minute basis. This program is nothing more than a wrapper around the dclock widget not associated with any particular widget set.

When the clock is running, the user may change attributes by typing:

r	Toggles **Reverse Video**.
s	Toggles the **seconds** display.
b	Toggles the **bell** attribute.
j	Toggles the **jump/scroll** attribute.
f	Toggles the **fade** attribute.
d	Toggles the **date** format.
m	Toggles the **military time** format.
a	Toggles the **alarm** clock.
t	Toggles the **tails** attribute.
q	quit the program.

The third mouse button puts the clock in the mode where the alarm clock can be set. The alarm must be set in 24-hour format to distringuish between am and pm time. Digits can be changed by selecting the digit with the left or middle mouse button. The Left button advances the time while the Middle button moves the time backwards. Using the left or middle button over the text at the bottom of the clock toggles the alarm from actually being set (or, use the 'a' key).

OPTIONS

Hdsdclock accepts all of the standard X Toolkit command line options along with the additional options listed below:

-help

This option indicates that a brief summary of the allowed options should be printed on the standard error.

-bell

This option indicates that the bell will beep once on the half hour and twice on the hour.

-miltime

This option causes the clock to display the time in 24 hour (military) time format.

-tails

This option puts "tails" on the digits 6 and 9.

-scroll

-noscroll

When the time changes, the digits scroll from the previous digit to the next digit. Since this is on by default, the -noscroll option can turn it off.

-fade

This option causes the display to fade from one digit to the next instead of jumping or scrolling. If fade is specified, the scroll attribute will be forced to off.

-fadeRate *n*

The fadeRate attribute takes an integer value which specifies the number of milli-seconds to wait between iterations when fading from one digit to the next. A larger value will make the fade slower and more pronounced. The default is 50. The fadeRate attribute also controls the scroll rate if the scroll attribute is on and the fade attribute is off.

-date *"format"*

The date is printed under the time in the specified font. The string displayed is in the "format" argument. If the string contains a formatting character (%), then the next character is examined and a value is inserted into the string. Example:

```
hdsdclock -date "Today is %W"
```

The date string will print "Today is Friday" if the weekday name happens to be friday. The formatting characters that are understood are:

%W	Full weekday name
%w	Three-char weekday name (Sun, Mon, Tue...)
%M	Full month name
%m	Three-char month name (Jan, Feb, Mar...)
%d	The date (numerical day number of the month)
%Y	Full year (4 digits)
%y	2-digit year number

-alarm

-noalarm

The alarm is turned on or off. Alarm rings bell and toggles the reverse video five times.

-alarmTime *HH:MM*

If alarm is set, it goes off at time specified in 24-hour format.

-seconds

This option will update the clock every second and display the time including the seconds.

-bg *color*

This option specifies the color to use for the background of the window. The default is ''white.''

-bd *color*

This option specifies the color to use for the border of the window. The default is ''black.''

-bw *number*

This option specifies the width in pixels of the border surrounding the window.

-fg *color*

This option specifies the color to use for displaying text. The default is ''black''.

-fn *font*

This option specifies the font to be used for displaying normal text. The default is ''Fixed.''

-rv

This option indicates that reverse video should be simulated by swapping the foreground and background colors.

-geometry *geometry*

This option specifies the prefered size and position of the clock window.

-display *host:display*

This option specifies the X server to contact.

-xrm *resourcestring*

This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

X DEFAULTS

The application class name is DClock. Hdsdclock understands all of the core resource names and classes as well as:

width (class Width)

Specifies the width of the clock.

height (class Height)

Specifies the height of the clock.

foreground (class Foreground)

Specifies the color for the tic marks. Using the class specifies the color for all things that normally would appear in the foreground color. The default is ''black'' since the core default for background is ''white.''

font (class Font)

Specifies the font to be used for the date.

miltime (class Boolean)

Specifies whether the military (24hr) time format should be used.

reverseVideo (class ReverseVideo)

Specifies that the foreground and background colors should be reversed.

scroll (class Boolean)

Specifies whether the digits should scroll or not.

fade (class Boolean)

Specifies whether the digits should fade or not.

fadeRate (class Time)

The number of milliseconds of delay between iterations of fading.

seconds (class Boolean)

Specifies whether the seconds should be displayed or not.

bell (class Boolean)

Specifies whether the bell should sound on the half hour and on the hour.

tails (class Boolean)

Specifies whether tails should be put on the digits 6 and 9.

alarm (class Boolean)

Specifies whether the alarm should go off at the specified time (alarmTime).

alarmTime (class String)

Specifies the time alarm goes off if alarm is set.

ENVIRONMENT

DISPLAY

to get the default host and display number.

XENVIRONMENT

to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

SEE ALSO

X(1), xrdb(1), time(3C).

BUGS

Hdsdclock determines the current time (GMT) by broadcasting to the internet *time* service provided by local UNIX hosts. The first valid reply is used. The conversion to local time is done by one of the following methods: If the environment variable TZ is a POSIX style time zone specification, it will be parsed to obtain the normal and daylight-savings offsets from GMT. If the environment variable TZ is a path to a time zone information file (typically /etc/localtime or /usr/share/lib/zoneinfo/localtime on a UNIX host), this file will be used to perform local time conversion. If the previous two methods fail, an approximation to local time conversion will be obtained by querying the internet *daytime* service on the same host which replied to the original *time* service broadcast. This calibration of system time to GMT and local time is local to each instance of hdsdclock, so multiple clocks may show different times if the system clocks of UNIX hosts on the local network are not exactly synchronized.

Scrolling from 9 to 10 O'Clock seems weird, but chances are you won't notice it.

If the window is too small, the seconds are tough to read. This should be recognized by the program and should display seconds using a font rather than the bitmaps used by the clock itself.

COPYRIGHT

Copyright (c) 1988, Dan Heller.

AUTHOR

Dan Heller -- <argv@sun.com> or <dheller@cory.berkeley.edu>

Fade & tails by Steve Reinhardt, <steven@fts.webo.dg.com>

2.6.5 HDSoclock Manual Page

This is the HDS oclock manual page.

NAME

hdsoclock - display time of day

SYNOPSIS

hdsoclock [-option ...]

DESCRIPTION

Hdsoclock simply displays the current time on an analog display

OPTIONS

-fg *foreground color*

choose a different color for the both hands and the jewel of the clock

-bg *background color*

choose a different color for the background.

-jewel *jewel color*

choose a different color for the jewel on the clock.

-minute *minute color*

choose a different color for the minute hand of the clock.

-hour *hour color*

choose a different color for the hour hand of the clock.

-geometry *geometry*

define the initial window geometry; see X(1).

-display *display*

specify the display to use; see X(1).

-bd *border color*

choose a different color for the window border.

-bw *border width*

choose a different width for the window border. As the Clock widget changes its border around quite a bit, this is most usefully set to zero.

-noshape

causes the clock to not reshape itself and ancestors to exactly fit the outline of the clock.

-transparent

causes the clock to consist only of the jewel, the hands, and the border.

COLORS

The application class name is Clock. Below are the default colors:

Clock*Background: grey

Clock*BorderColor: light blue

Clock*hour: yellow

Clock*jewel: yellow

Clock*minute: yellow

SEE ALSO

X(1), X Toolkit documentation

BUGS

Hdsoclock determines the current time (GMT) by broadcasting to the internet *time* service provided by local UNIX hosts. The first valid reply is used. The conversion to local time is done by one of the following methods: If the environment variable TZ is a POSIX style time zone specification, it will be parsed to obtain the normal and daylight-savings offsets from GMT. If the environment variable TZ is a path to a time zone information file (typically /etc/localtime or /usr/share/lib/zoneinfo/localtime on a UNIX host), this file will be used to perform local time conversion. If the previous two methods fail, an approximation to local time conversion will be obtained by querying the internet *daytime* service on the same host which replied to the original *time* service broadcast. This calibration of system time to GMT and local time is local to each instance of hdsoclock, so multiple clocks may show different times if the system clocks of UNIX hosts on the local network are not exactly synchronized.

COPYRIGHT

Copyright 1989, Massachusetts Institute of Technology.

See X(1) for a full statement of rights and permissions.

AUTHOR

Keith Packard, MIT X Consortium

2.6.6 HDSxclock Manual Page

This is the HDS xclock man page.

NAME

hdsxclock - analog / digital clock for X

SYNOPSIS

hdsxclock [-*toolkitoption* ...] [-help] [-analog] [-digital] [-chime] [-hd *color*] [-hl *color*] [-update *seconds*] [-padding *number*]

DESCRIPTION

The *hdsxclock* program displays the time in analog or digital form. The time is continuously updated at a frequency which may be specified by the user. This program is nothing more than a wrapper around the Athena Clock widget.

OPTIONS

Hdsxclock accepts all of the standard X Toolkit command line options along with the additional options listed below:

-help

This option indicates that a brief summary of the allowed options should be printed on the standard error.

-analog

This option indicates that a conventional 12 hour clock face with tick marks and hands should be used. This is the default.

-digital

This option indicates that a 24 hour digital clock should be used.

-chime

This option indicates that the clock should chime once on the half hour and twice on the hour.

-hd *color*

This option specifies the color of the hands on an analog clock. The default is *black*.

-hl *color*

This option specifies the color of the edges of the hands on an analog clock, and is only useful on color displays. The default is *black*.

-update *seconds*

This option specifies the frequency in seconds at which *hdsxclock* should update its display. If the clock is obscured and then exposed, it will be updated immediately. A value of less than 30 seconds will enable a second hand on an analog clock. The default is 60 seconds.

-padding *number*

This option specifies the width in pixels of the padding between the window border and clock text or picture. The default is 10 on a digital clock and 8 on an analog clock.

X DEFAULTS

The application class name is XClock. This program uses the *Athena Clock* widget. It understands all of the core resource names and classes as well as:

width (class Width)

Specifies the width of the clock. The default for analog clocks is 164 pixels; the default for digital clocks is whatever is needed to hold the clock when displayed in the chosen font.

height (class Height)

Specifies the height of the clock. The default for analog clocks is 164 pixels; the default for digital clocks is whatever is needed to hold the clock when displayed in the chosen font.

update (class Interval)

Specifies the frequency in seconds at which the time should be redisplayed.

foreground (class Foreground)

Specifies the color for the tic marks. The default is depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *white*, otherwise the default is *black*.

hands (class Foreground)

Specifies the color of the insides of the clock's hands. The default is depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *white*, otherwise the default is *black*.

highlight (class Foreground)

Specifies the color used to highlight the clock's hands. The default is depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *white*, otherwise the default is *black*.

analog (class Boolean)

Specifies whether or not an analog clock should be used instead of a digital one. The default is True.

chime (class Boolean)

Specifies whether or not a bell should be rung on the hour and half hour.

padding (class Margin)

Specifies the amount of internal padding in pixels to be used. The default is 8.

font (class Font)

Specifies the font to be used for the digital clock. Note that variable width fonts currently will not always display correctly.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *hdsxclock*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

XClock hdsxclock

Clock clock

ENVIRONMENT

DISPLAY

to get the default host and display number.

XENVIRONMENT

to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

SEE ALSO

X(1), xrdb(1), time(3C), Athena Clock widget

BUGS

Hdsxclock determines the current time (GMT) by broadcasting to the internet *time* service provided by local UNIX hosts. The first valid reply is used. The conversion to local time is done by one of the following methods: If the environment variable TZ is a POSIX style time zone specification, it will be parsed to obtain the normal and daylight-savings offsets from GMT. If the environment variable TZ is a path to a time zone information file (typically /etc/localtime or /usr/share/lib/zoneinfo/localtime on a UNIX host), this file will be used to perform local time conversion. If the previous two methods fail, an approximation to local time conversion will be obtained by querying the internet *daytime* service on the same host which replied to the original *time* service broadcast. This calibration of system time to GMT and local time is local to each instance of hdsxclock, so multiple clocks may show different times if the system clocks of UNIX hosts on the local network are not exactly synchronized.

When in digital mode, the string should be centered automatically.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.

See X(1) for a full statement of rights and permissions.

AUTHORS

Tony Della Fera (MIT-Athena, DEC)

Dave Mankins (MIT-Athena, BBN)

Ed Moy (UC Berkeley)

2.6.7 HDSscreensaver Manual Page

This is the HDS screensaver manual page.

NAME

hdssaver - local ViewStation FX screensaver, chooser, and locking client.

SYNOPSIS

hdssaver [-display host:display.screen] [-program screensaver] [-timeout int] [-cycle int] [-fade] [-nofade] [-unfade] [-nounfade] [-nographics] [-lock] [-lockalways] [-password password] [-xrm resources]

OPTION

hdssaver provides the ViewStation user with a list of locally run screensaver graphics programs using a standard Motif file selection box widget. This program waits until the keyboard and mouse have been idle for a period, and then run the currently chosen graphics demo. It turns off the demo as soon as there is any mouse or keyboard activity. Its purpose is to display pretty pictures on your screen when it is not in use, in keeping with the philosophy that unattended monitors should always be doing something interesting, just like they do in the movies. By running these programs locally, network load is diminished.

This program also acts as a ViewStation screen locker. When lock is selected the user is prompted for a password (maximum of ten characters). Once the password is verified, the currently chosen graphics demo will run. Any further use of the keyboard or mouse causes the ViewStation to prompt for the password. If the correct password is entered within 30 seconds the graphics demo is turned off and the ViewStation returned to its previous state.

The following graphics demo programs are available and can be customized either through command line options, or resource settings:

hdslogo	Blinking HDS logo
attraction	A bouncing ball demo, or a qix-like demo, or a wild color-cycling thing, with some odd rules.
blitspin	Rotate a bitmap using bitblts.
greynetic	Random colored / stippled rectangles.
helix	Generates spirally "stringart" patterns.
hopalong	Fractals.
hypercube	2D projection of a 4D object.
imsmap	Generates random maps or cloud formations

noseguy	A guy with a big nose wanders around the screen saying things.
pyro	Fireworks.
qix	Flying string art.
random	Cycle through list randomly.
rocks	Flying through an asteroid field.
rorschach	Random inkblot patterns.
slidescreen	Permute the screen image like an 8-puzzle
xroger	Blinking jolly roger flag.

See the individual man pages for more details.

Provisions are made to allow the user to add and delete from this list, modify timeout, cycle, and fade times, as well as instantly view each graphics program.

OPTIONS

hdssaver accepts the following options:

-program screensaver

The screensaver will activate this program after the keyboard and mouse have been idle for the time set by timeout. This program must be a set as a local client of type screensaver as defined by the user's client.config file. Please refer to the ViewStation User's manual for more information concerning local clients. Default value is hdslogo (a blinking HDS logo).

-timeout seconds

The screensaver will activate after the keyboard and mouse have been idle for this many seconds. Default value is 120 seconds with a minimum of ten.

-cycle seconds

After the screensaver has been running for this many seconds, the currently running screensaver will be killed (with SIGTERM), and a new one started. This is only applicable when random is selected and has a default value of 30 seconds with a minimum of 10.

-fade

If set, the current screen will fade out before the screensaver graphics program is run. Default set.

-nofade

If set, the current screen will just blank before the screensaver graphics program is run.

-unfade

If set, the screensaver graphics program will fade out before the screen is restored. Default set.

-nounfade

If set, the current screensaver will just blank before the screen is restored.

-nographics

If set, the user interface widget will not be displayed.

-password password

The password used if screensaver is set as lock.

-lock

If set the display will be locked immediately. The user will be prompted for a locking password if one has not already been provided. The default saver is used unless a different one is specified.

-lockalways

If set, every screensaver will be invoked with the locking option set. The password option MUST be used in conjunction for this option to be valid.

X DEFAULTS

hdsscreensaver understands the following resources:

program (class Program)

Same as the -program command-line option. Default HDSlogo.

timeout (class Time)

Same as the -timeout command-line option. Default 120 seconds.

cycle (class Time)

Same as the -cycle command-line option. Default 30 seconds.

fade (class Boolean)

Same as the -fade command-line option. If this is true, then when the screensaver activates, the current contents of the screen will fade to black instead of simply winking out. This only works on displays with writable colormaps. Default true. A fade will also be done when switching graphics screensavers (when the cycle timer expires and random is selected.)

unfade (class Boolean)

Same as the -unfade command-line option. If this is true, then when the screensaver deactivates, the original contents of the screen will fade in from black instead of appearing immediately. This only works on displays with writable colormaps. Default True.

fadeSeconds (class Time)

If fade or unfade is true, this is how long the fade will be in seconds (default 1.)

fadeTicks (class Integer)

If fade is true, this is how many times a second the colormap will be changed to effect a fade. Higher numbers yield smoother fades, but may make the fades take longer if your server isn't fast enough to keep up. Default 75.

nographics (class Boolean)

If this is true the graphics user interface widget will not be displayed. up. Default False.

lock (class Boolean)

If true the display will be locked immediately. The user will be prompted for a locking password if one has not already been provided. The default saver is used unless a different one is specified.

lockalways (class Boolean)

If this is true each screensaver will be invoked with the lock option set. Note that the screensaver password must also be set for this to work. up. Default False.

password (class Char)

The screensaver password used when the lock option is set. Default Null.

programs (class Programs)

The graphics demo programs which hdsscreensaver runs when the user is idle. The value of this resource is a string, one screensaver command per line. Each line must contain exactly one command -- no semi-colons.

If the value of this resource is empty, then no programs will be run; the screen will simply be made black.

Note that you must escape the newlines; here is an example of how you might set this in your .Xdefaults file:

```
hdsscreensaver.programs: \
rocks \n\
qix \n\
rocks \n\
hdslogo
```

HOW IT WORKS

When it is time to activate the screensaver, a full-screen black window is created. This window is given the appropriate properties so that, to any subsequently-created programs, it will appear to be a ''virtual root'' window. The currently selected graphics screensaver program is then run.

When the user becomes active again, the screensaver window is unmapped and the running subprocess is killed by sending it SIGTERM. This is also how the subprocesses are killed when the screensaver decides that it's time to run a different demo if random is selected: the old one is killed and a new one is launched.

Before launching a subprocess, hdsscreensaver stores an appropriate value for $DISPLAY in the environment that the child will recieve. (This is so that if you

start hdsscreensaver with a -display argument, the programs which hdsscreensaver launches will draw on the same display.)

When the screensaver turns off, or is killed, care is taken to restore the ''real'' virtual root window if there is one. Because of this, it is important that you not kill the screensaver process with kill -9 if you are running a virtual-root window manager. If you kill it with -9, you may need to restart your window manager to repair the damage. This isn't an issue if you aren't running a virtual- root window manager.

ENVIRONMENT

DISPLAY to get the default host and display number.

XENVIRONMENT to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

SEE ALSO

hdslogo(1), attraction(1), blitspin(1), greynetic(1),

helix(1), hopalong(1), hypercube(1), imsmap(1), noseguy(1),

pyro(1), qix(1), random(1), rocks(1), rorschach(1),

slidescreen(1), xroger(1).

BUGS

Apparently there are some problems with ''XView'' programs getting confused and thinking that the screensaver window is the real root window even when the screensaver is not active: ClientMessages intended for the window manager are sent to the screensaver window instead. This could be solved by making HDS saver forward all unrecognised ClientMessages to the real root window, but there may be other problems as well.

COPYRIGHT

Copyright 1993, Human Designed Systems, Inc.

AUTHORS

Joseph H Durant (Human Designed Systems)

Modified merge of xscreensaver and xnlock clients ported as a local client to the ViewStation FX.

Please let me know if you find any bugs or make any improvements.

2.6.8 HDStetris Manual Page

This is the HDS tetris man page.

NAME

hdstetris - X Window block dropping game

SYNOPSIS

hdstetris [X options]

DESCRIPTION

hdstetris is a game involving dropping blocks. As they drop you can move them to the left or to the right by clicking the left or right mouse buttons respectively. Pressing the shift key while clicking the button causes the falling shape to rotate anti-clockwise (left mouse button) or clockwise (right mouse button). Pressing the middle mouse button causes the shape to quickly drop. Note that the mouse must be in the window in which the shapes are dropping for these actions to work correctly.

You can also use the keyboard: h for moving to the left, l for moving to the right, j to rotate clockwise, k to rotate counter-clockwise and space for dropping quickly..

You can also use the direction pad, if your keyboard has one: (Left) to move left, (Down) to rotate clockwise, (Up) to rotate counter clockwise, and (Right) to move right.

Points are scored for each block that comes to rest on the gradually building up pile of blocks. Different blocks in different orientations have different point values. When the pile reaches the top of the screen and no further blocks can be dropped the game ends. When a row of blocks across the screen is completely filled, that row is removed and all the blocks above it drop down. As more rows are deleted the blocks drop faster.

Almost everything about hdstetris can be configured via the application defaults file.

OPTIONS

hdstetris takes standard X options (see X(1)). It adds the following:

-speed value

This sets the game's speed. By default the speed is 10. 20 causes the game to be 2 times faster, 5 makes it 2 times slower. "-speed 50" makes for an extremely fast game. Keep in mind that as you knock out rows, the game's speed increases.

SEE ALSO

X(1)

AUTHORS

William Pohlmann (Human Designed Systems), converted from Athena to Motif

Dan Greening wrote most of xtetris 2.0-2.3, based on earlier work by Didier Tallot, Phill Everson, Martyn Shortley, and Adam Marguilies.

2.6.9 HDSview Manual Page

This is the HDS view man page.

NAME

hdsview - View PostScript documents using hdsscript

SYNOPSIS

hdsview [filename]
or
hdsview [-staticgray] [-grayscale] [-staticcolor] [-pseudocolor] [-truecolor] [-directcolor] [-[no]install] [-[no]private] [-[no]center] [-[no]title] [-[no]date] [-[no]locator] [-[no]labels] [-resolution dpi] [-dpi dpi] [-xdpi dpi] [-ydpi dpi] [-[no]quiet] [-preload file] [-magstep n] [-portrait] [-landscape] [-upsidedown] [-seascape] [-letter] [-tabloid] [-ledger] [-legal] [-statement] [-executive] [-a3] [-a4] [-a5] [-b4] [-b5] [-folio] [-quarto] [-10x14] [-force] [-forceorientation] [-forcemedia] [-[no]swap] [-[no]openwindows] [-page label] [-toolkitoption ...] [filename]

DESCRIPTION

The hdsview program provides a Motif user interface for the hdsscript interpreter. Hdsview and hdsscript function as two cooperating programs. Hdsview creates the viewing window and hdsscript draws in it.

Don't be alarmed by the number of options. Generally, one invokes hdsview with just one parameter, the name of the file to be previewed. If the filename is ''-'', hdsview will read from ''stdin''. The options provide a way to set X resources from the command line for a single invocation of hdsview. For that reason, discussion of the options is delayed until after the X resources are discussed.

MAIN WINDOW

The main viewport is on the right side of the main window. There are scroll bars along the bottom and right edges of the viewport. To the left of the viewport is the table of contents. If the PostScript file has document structuring convention (DSC) comments, the table of contents will display the page labels (i.e. usually page numbers). At the top of the application is the menu bar, consisting of push buttons. Each push button brings up a pulldown menu. Over the table of contents there is one optional label that contains the locator. The locator shows the location of the cursor in the viewport. The location is expressed in the default user coordinate system. The locator is useful for measuring bounding boxes.

Within the main viewport the mouse cursor is a ''target'' when hdsscript is doing work. The cursor is a ''crosshair'' when hdsscript is idle. When moving to another page in a document, it is generally best to wait for hdsscript to become idle. Otherwise, the current hdsscript process must be killed and the overhead of reading the prologue is incurred again.

Clicking anywhere within the viewport will popup a zoom window. The window is centered about the location that was clicked. Clicking with the first mouse button pops up a low resolution zoom window. Clicking with the second mouse button pops up a medium resolution zoom window. Clicking with the third mouse button pops up a high resolution zoom window. The cursor in the zoom window will display a ''target'' or ''cross hair'' depending on the state of hdsscript. The locator reports the position of the cursor in the zoom windows as well as the main viewport.

In the table of contents, the first and third mouse button functions exactly as they do in a Text widget. That is the first mouse button selects text and the third mouse button extends selections. However, clicking on a page label with the second mouse button will cause that page to be shown. The page being displayed is marked with a '<' in the right margin of the table of contents.

POPUP WINDOWS

The are several additional windows that may appear. The copyright window will appear whenever copyright information is requested. The information window appears whenever the hdsscript process writes to ''stdout'' or ''stderr''. Whenever there is an error in the PostScript program, the stack trace will appear in this window. When hdsscript is invoked without the quiet option, informational message regarding the state of hdsscript will appear in the information window. The copyright and information windows can be dismissed by pushing the ''Dismiss'' button.

MENUS

There are six buttons on the menu bar. They are the File, Page, Magstep, Orientation, Media, and Help buttons. There are keyboard accelerators for most of the menu items. Be sure to read the keyboard accelerator section.

File

This popup menu controls file access:

Open...

Pops up a dialog widget to ask for the file name. The widget is in total control until it pops down. Opens the file for viewing.

Reopen

Reopens the current file.

Print...

Pops up a dialog widget to ask for the printer name. Sends the whole document to the printer.

Print marked pages...

Pops up a dialog widget to ask for the printer name. Sends the marked pages to the printer. If no pages have been marked, the mark procedure is called before printing and the unmark procedure is called after printing.

Save marked pages...

Pops up a dialog widget to ask for the file name. Saves the marked pages in the selected file. If no pages have been marked, the mark procedure is called before saving and the unmark procedure is called after saving.

Copyright...

Pops up the copyright window.

Quit

Causes hdsview to exit.

Page

This popup menu controls page access:

Next

Display the next page.

Redisplay

Display the current page.

Previous

Display the previous page.

Center

Center the page in the viewport.

Mark

Mark the pages that have been selected in the table of contents. If no pages have been selected, mark the current page.

Unmark

Unmark the pages that have been selected in the table of contents. If no pages have been selected, unmark the current page.

Magstep

The Magstep menu controls the magnification at which the document is viewed. The default magstep of 0 implies no magnification (i.e. the size on the screen should match the size on paper). Hdsview borrows the notion of magstep from TeX. The magnification is defined to be 1.2**magstep. At magstep 1, the document is magnified by 1.2. At magstep -1, the document is reduced by 1.2. The Magstep menu lists values from -5 to 5.

Orientation

The Orientation menu controls the display orientation. A ''diamond'' appears in front of the current orientation.

Portrait

Set the orientation to Portrait. This implies no rotation. A portrait orientation indicates that the longest edge of the paper is parallel to the vertical (y) axis.

Landscape

Set the orientation to Landscape. This implies a clockwise rotation of the paper by 90 degrees. A landscape orientation indicates that the longest edge of the paper is parallel to the horizontal (x) axis.

Upside-down

Set the orientation to Upside-Down.

Seascape

Set the orientation to Seascape. This implies a counterclockwise rotation of the paper by 90 degrees.

Swap Landscape

Swap the meaning of Landscape and Seascape, since some documents need to be rotated the other way. The swap landscape button allows hdsview to automatically rotate the document the right way in response to the %%Orientation comment in the PostScript file.

Media

The entries on the Media menu set the page media. Media defined in the document appear at the beginning of the menu separated by a line from the standard media. A ''diamond'' appears in front of the current media.

Here are the standard media names and their sizes. The size is given as the width and height in PostScript points.

Letter 612 x 792 (8.5 x 11 in.)

Tabloid 792 x 1224 (11 x 17 in.)

Ledger 1224 x 792 (17 x 11 in.)

Legal 612 x 1008 (8.5 x 14 in.)

Statement 396 x 612 (5.5 x 8.5 in.)

Executive 540 x 720 (7.5 x 10 in.)

A3 842 x 1190

A4 595 x 842

A5 420 x 595

B4 729 x 1032

B5 516 x 729

Folio 612 x 936 (8.5 x 13 in.)

Quarto 610 x 780

10x14 720 x 1008 (10 x 14 in.)

KEYBOARD ACCELERATORS

Most of the pulldown menu commands have an equivalent action that can be invoked from the keyboard. The pulldown menu entry must be sensitive (i.e. not grayed out) for the accelerator to have effect. Here are the default keyboard accelerators:

Q	Equivalent to pushing the Quit menu button on the File menu.
O	Equivalent to pushing the Open... menu button on the File menu.
R	Equivalent to pushing the Reopen menu button on the File menu.
S	Equivalent to pushing the Save marked pages... menu button on the File menu.
P	Equivalent to pushing the Print marked pages... menu button on the File menu.
Shift-P	Equivalent to pushing the Print... menu button on the File menu.
BackSpace	Equivalent to pushing the Previous menu button on the Page menu.
space	Equivalent to pushing the Next menu button on the Page menu.
Ctrl-L	Equivalent to pushing the Redisplay menu button on the Page menu.
M	Equivalent to pushing the Mark menu button on the Page menu.
N	Equivalent to pushing the Unmark menu button on the Page menu.
0	Equivalent to pushing the 0 menu button on the Magstep menu.
1	Equivalent to pushing the 1 menu button on the Magstep menu.
2	Equivalent to pushing the 2 menu button on the Magstep menu.

3	Equivalent to pushing the 3 menu button on the Magstep menu.
4	Equivalent to pushing the 4 menu button on the Magstep menu.
5	Equivalent to pushing the 5 menu button on the Magstep menu.

ACTIONS

Most of the pulldown menu commands have an equivalent action that can be used in a translation. The pulldown menu entry must be sensitive (i.e. not grayed out) for the action to have effect. Here is the list of actions:

hdsviewCopyright()

Equivalent to pushing the Copyright... menu button on the File menu.

hdsviewQuit()

Equivalent to pushing the Quit menu button on the File menu.

hdsviewOpen()

Equivalent to pushing the Open... menu button on the File menu.

hdsviewReopen()

Equivalent to pushing the Reopen menu button on the File menu.

hdsviewSave()

Equivalent to pushing the Save marked pages... menu button on the File menu.

hdsviewPrintWhole()

Equivalent to pushing the Print... menu button on the File menu.

hdsviewPrintMarked()

Equivalent to pushing the Print marked pages... menu button on the File menu.

hdsviewPrevious()

Equivalent to pushing the Previous menu button on the Page menu.

hdsviewShow()

Equivalent to pushing the Redisplay menu button on the Page menu.

hdsviewNext()

Equivalent to pushing the Next menu button on the Page menu.

hdsviewCenter()

Equivalent to pushing the Center menu button on the Page menu.

hdsviewMark()

Equivalent to pushing the Mark menu button on the Page menu.

hdsviewUnmark()

Equivalent to pushing the Unmark menu button on the Page menu.

hdsviewSetMagstep(magstep)

Sets the magstep. The parameter must be an integer.

hdsviewIncreaseMagstep()

Increases magstep by one.

hdsviewDecreaseMagstep()

Decreases magstep by one.

hdsviewSetOrientation(orientation)

Set the orientation to the passed parameter. The parameter must be portrait, landscape, upside-down, or seascape.

hdsviewSwapLandscape()

Equivalent to pushing the Swap Landscape menu button on the Orientation menu.

hdsviewSetPageMedia(media-name)

Sets the media. The parameter should be either a media defined in the document or a standard media.

hdsviewDefault()

The orientation or media being set is not forced on the document. This action is called before the action that sets the orientation or media.

hdsviewForce()

The orientation or media being set is forced on the document. This action is called before the action that sets the orientation or media.

hdsviewDeleteWindow()

Destroy the current window. This provides a way to implement the Delete Window protocol for window managers.

hdsviewDismiss()

Pop down the current window. This provides a way to implement the Delete Window protocol for window managers.

hdsviewScrollUp()

Scroll the main viewport up.

hdsviewScrollDown()

Scroll the main viewport down.

hdsviewScrollLeft()

Scroll the main viewport left.

hdsviewScrollRight()

Scroll the main viewport right.

hdsviewEraseLocator()

Used to erase the locator when leaving an hdsview widget.

hdsviewCheckFile()

Checks to see if the file changed and refreshes the screen if necessary.

KEYBOARD TRANSLATIONS

The pulldown menu entry must be sensitive (i.e. not grayed out) for the action to have effect. Here is the default keyboard binding:

+	Bound to hdsviewIncreaseMagstep() which increases the magstep by 1.
-	Bound to hdsviewDecreaseMagstep() which decreases the magstep by 1.
G	Bound to hdsviewLeft() which scrolls the main viewport left.
J	Bound to hdsviewDown() which scrolls the main viewport down.
K	Bound to hdsviewUp() which scrolls the main viewport up.
L	Bound to hdsviewRight() which scrolls the main viewport right.
Up (arrow)	Bound to hdsviewDefault()hdsviewSetOrientation(portrait) which is equivalent to pushing Portrait with the first mouse button on the Orientation menu.
Right (arrow)	Bound to hdsviewDefault() hdsviewSetOrientation(landscape) which is equivalent to pushing Landscape with the first mouse button on the Orientation menu.
Down (arrow)	Bound to hdsviewDefault() hdsviewSetOrientation(upside-down) which is equivalent to pushing Upside-down with the first mouse button on the Orientation menu.
Left (arrow)	Bound to hdsviewDefault() hdsviewSetOrientation(seascape) which is equivalent to pushing Seascape with the first mouse button on the Orientation menu.
Shift-Up (arrow)	Bound to hdsviewForce() hdsviewSetOrientation(portrait) which is equivalent to pushing Portrait with the second mouse button on the Orientation menu.
Shift-Right (arrow)	Bound to hdsviewForce() hdsviewSetOrientation(landscape) which is equivalent to pushing Landscape with the second mouse button on the Orientation menu.
Shift-Down (arrow)	Bound to hdsviewForce() hdsviewSetOrientation(upside -down) which is equivalent to pushing Upside-down with the second mouse button on the Orientation menu.

Shift-Left (arrow) Bound to hdsviewForce() hdsviewSetOrientation(seascape) which is equivalent to pushing Seascape with the second mouse button on the Orientation menu.

APPLICATION RESOURCES

The following application resources may be set to control the default behavior of hdsview.

showLocator (class Labels)

Tells whether to display the locator. The default is ''true''.

installStdCmap (class InstallStdCmap)

Tells whether to attempt to install a standard colormap. The default is ''false''.

privateCmap (class PrivateCmap)

Tells whether to use a standard colormap that is not the default colormap. The default is ''false''.

autoCenter (class AutoCenter)

Tells whether to center the page within the viewport whenever the page size changes. The default is ''true''.

horizonalMargin (class Margin)

Tells how many pixels hdsview should reserve for window decorations in the horizontal direction. The default value is ''20''.

verticalMargin (class Margin)

Tells how many pixels hdsview should reserve for window decorations in the vertical direction. The default value is ''44''.

minimumMagstep (class Magstep)

Tells the smallest magstep to display. The default is ''-5''.

maximumMagstep (class Magstep)

Tells the largest magstep to display. The default is ''5''.

magstep (class Magstep)

Sets the default magstep. The default is ''0''.

orientation (class Orientation)

Sets the default orientation. The default is ''Portrait''.

page (class Page)

Gives the initial page to display. This resource only affects the display of the file listed on the command line. The default is NULL.

pageMedia (class PageMedia)

Sets the default page media. The default is ''Letter''.

forceOrientation (class Force)

Tells whether to force the orientation on the document. The default is ''false''.

forcePageMedia (class Force)

Tells whether to force the page media on the document. The default is ''false''.

swapLandscape (class SwapLandscape)

Tells whether to swap the meaning of Landscape and Seascape. The default is ''false''.

printCommand (class PrintCommand)

Sets the command used for printing. The printer environment variable is set to the desired printer and then this command is executed using popen. This command should read from ''stdin'' and send the file to the appropriate printer. The default value is ''lp''.

printerVariable (class PrinterVariable)

Gives the name of the printer environment variable. The default value is ''LPDEST''.

defaultPrinter (class DefaultPrinter)

Gives the printer name to use when the printer environment variable is not set. The default value is NULL.

printPrompt (class PrintPrompt)

Sets the prompt used to ask for the printer name. The default value is ''Printer Name: ''.

printFail (class printFail)

Sets the string used to inform the user that the printer command failed. The default is ''"lpr" command failed.''.

openPrompt (class OpenPrompt)

Sets the prompt used to ask for a file name to open. The default value is ''Open File: ''.

openFail (class OpenFail)

Sets the string used to inform the user that the open failed. The default value is ''Cannot open file: ''.

savePrompt (class SavePrompt)

Sets the prompt used to ask for a file name to save. The default value is ''Save File: ''.

saveFail (class SaveFail)

Sets the string used to inform the user that the save failed. The default value is ''Cannot save file: ''.

HDSVIEW WIDGET RESOURCES

Certain resources in the hdsview widget may be set by the user. These selected resources are presented below.

busyCursor (class Cursor)

The cursor shown when hdsscript is rendering to the window. The busy cursor is set to the ''target'' by the application defaults.

cursor (class Cursor)

The cursor shown when hdsscript is idle. The default cursor is the ''crosshair''.

interpreter (class Interpreter)

The name of the executable to call to render the PostScript. It is convenient to set this resource to the path of an alternate version of hdsscript for testing. The default value is ''hdsscript''.

preload (class Preload)

Files to load before the PostScript program. It is convenient to name files that preload fonts here for PostScript programs that continually reload fonts while rendering a page. The default is to not preload any files.

quiet (class Quiet)

Tells hdsscript whether to produce informational messages. The default value is ''false''.

useBackingPixmap (class UseBackingPixmap)

Tells whether to use a backing pixmap. If this resource is false, backing store is requested on the hdsview window. Some X servers have limited resources for large pixmaps. Also, some X servers' backing store is much faster than using a backing pixmap. You should reset this resource if your X server is one of the server types mentioned. The default value is ''true''.

xdpi (class Resolution)

Sets the X resolution of the window in dots per inch. You can use this resource to affect the main window. Zoom windows have their X dpi set explicitly in the program. The default value is calculated from the screen metrics.

ydpi (class Resolution)

Sets the Y resolution of the window in dots per inch. You can use this resource to affect the main window. Zoom windows have their Y dpi set explicitly in the program. The default value is calculated from the screen metrics.

HDSVIEW WIDGET ACTIONS

notify(width height xdpi ydpi)

The notify event is used by the hdsview application for the locator and popup zoom windows. If the width and height are 0, the event is used for the locator. Otherwise, it triggers a popup zoom window. The default width and height are

72. The default xdpi and ydpi are 300. The height will default to the width if the height is omitted. The ydpi will default to the xdpi if the ydpi is omitted.

OPTIONS

-staticgray

Equivalent to setting ''*Visual: StaticGray''.

-grayscale

Equivalent to setting ''*Visual: GrayScale''.

-staticcolor

Equivalent to setting ''*Visual: StaticColor''.

-pseudocolor

Equivalent to setting ''*Visual: PseudoColor''.

-truecolor

Equivalent to setting ''*Visual: TrueColor''.

-directcolor

Equivalent to setting ''*Visual: DirectColor''.

-install

Equivalent to setting ''HDSview.installStdCmap: True''.

-noinstall

Equivalent to setting ''HDSview.installStdCmap: False''.

-private

Equivalent to setting ''HDSview.privateCmap: True''.

-noprivate

Equivalent to setting ''HDSview.privateCmap: False''.

-center

Equivalent to setting ''HDSview.autoCenter: True''.

-nocenter

Equivalent to setting ''HDSview.autoCenter: False''.

-locator

Equivalent to setting ''HDSview.showLocator: True''.

-nolocator

Equivalent to setting ''HDSview.showLocator: False''.

-labels

Equivalent to setting ''HDSview.Labels: True''.

-nolabels

Equivalent to setting ''HDSview.Labels: False''.

-quiet

Equivalent to setting ''*hdsview.quiet: True''.

-noquiet

Equivalent to setting ''*hdsview.quiet: False''.

-preload file

Equivalent to setting ''*hdsview.preload: file''.

-xdpi dpi

Equivalent to setting ''*hdsview.xdpi: dpi''.

-ydpi dpi

Equivalent to setting ''*hdsview.ydpi: dpi''.

-resolution dpi

Equivalent to setting ''*hdsview.Resolution: dpi''.

-dpi dpi

Equivalent to setting ''*hdsview.Resolution: dpi''.

-magstep magstep

Equivalent to setting ''HDSview.magstep: magstep''.

-portrait

Equivalent to setting ''HDSview.orientation: Portrait''.

-landscape

Equivalent to setting ''HDSview.orientation: Landscape''.

-upsidedown

Equivalent to setting ''HDSview.orientation: Upside-down''.

-seascape

Equivalent to setting ''HDSview.orientation: Seascape''.

-forceorientation

Equivalent to setting ''HDSview.forceOrientation: True''.

-page label

Equivalent to setting ''HDSview.page: label''.

-letter

Equivalent to setting ''HDSview.pageMedia: Letter''.

-tabloid

Equivalent to setting ''HDSview.pageMedia: Tabloid''.

-ledger

Equivalent to setting ''HDSview.pageMedia: Ledger''.

-legal
Equivalent to setting ''HDSview.pageMedia: Legal''.

-statement
Equivalent to setting ''HDSview.pageMedia: Statement''.
-executive
Equivalent to setting ''HDSview.pageMedia: Executive''.
-a3
Equivalent to setting ''HDSview.pageMedia: A3''.
-a4
Equivalent to setting ''HDSview.pageMedia: A4''.
-a5
Equivalent to setting ''HDSview.pageMedia: A5''.
-b4
Equivalent to setting ''HDSview.pageMedia: B4''.
-b5
Equivalent to setting ''HDSview.pageMedia: B5''.
-folio
Equivalent to setting ''HDSview.pageMedia: Folio''.
-quarto
Equivalent to setting ''HDSview.pageMedia: Quarto''.
-10x14
Equivalent to setting ''HDSview.pageMedia: 10x14''.
-forcemedia
Equivalent to setting ''HDSview.forcePageMedia: True''.
-force
Equivalent to setting ''HDSview.Force: True''.
-swap
Equivalent to setting ''HDSview.swapLandscape: True''.
-noswap
Equivalent to setting ''HDSview.swapLandscape: False''.
-openwindows
Equivalent to setting ''HDSview.openWindows: True''.
-noopenwindows
Equivalent to setting ''HDSview.openWindows: False''.

-ncdwm

Equivalent to setting ''HDSview.ncdwm: True''.

-noncdwm

Equivalent to setting ''HDSview.ncdwm: False''.

WIDGET HIERARCHY

The hierarchy of the hdsview application:

HDSview hdsview

- XmForm form
- XmRowColumn menuBar
- XmRowColumn fileButton
- XmPushButton open
- XmPushButton reopen
- XmPushButton printwhole
- XmPushButton printmarked
- XmPushButton save
- XmSeparator
- XmPushButton copyright
- XmPushButton quit
- XmRowColumn pageButton
- XmPushButton next
- XmPushButton show
- XmPushButton prev
- XmSeparator
- XmPushButton center
- XmSeparator
- XmPushButton mark
- XmPushButton unmark
- XmRowColumn magstepButton
- XmToggleButton -5
- XmToggleButton -4
- XmToggleButton -3
- XmToggleButton -2
- XmToggleButton -1

XmToggleButton zero
XmToggleButton one
XmToggleButton two
XmToggleButton three
XmToggleButton four
XmToggleButton five
XmRowColumn orientationButton
XmToggleButton portrait
XmToggleButton landscape
XmToggleButton upsidedown
XmToggleButton seascape
XmSeparator
XmToggleButton swap
XmRowColumn pagemediaButton
XmToggleButton Letter
XmToggleButton Tabloid
XmToggleButton Ledger
XmToggleButton Legal
XmToggleButton Statement
XmToggleButton Executive
XmToggleButton A3
XmToggleButton A4
XmToggleButton A5
XmToggleButton B4
XmToggleButton B5
XmToggleButton Folio
XmToggleButton Quarto
XmToggleButton 10x14
XmRowColumn helpButton
XmPushButton generalhelp
XmPushButton filehelp
XmPushButton pagehelp
XmPushButton magstephelp
XmPushButton orienthelp
XmPushButton mediahelp

XmLabel locator

XmScrolledList toc

XmScrolledWindow pageview

Core clip

hdsview page

TopLevelShell viewzoom

XmForm form

hdsview page

XmPushButton dismiss

TopLevelShell information

XmForm form

XmText infotext

XmPushButton dismiss

XmMessageBox error

XmMessageBox copyright

ENVIRONMENT

LPDEST The LPDEST environment variable gives the default printer destination.

LIMITATIONS

If the document does not begin with "%!PS-Adobe-", it does not claim conformance to the document structuring convention. When these documents are encountered, the functionality of hdsview is limited to giving you scroll bars and a next page capability. Because there is no table of contents, skipping around the document and marking pages is impossible.

If there is no table of contents for the document, the popup zoom window will always show the first page.

There is no visual indication of when an orientation or media is being "forced".

The NFS file access protocol must be used to view remote multiple page documents.

There is no support for local printing.

Remote printing works only when hdsview has been started by rshing from a remote host.

Saving the document to a file is not supported.

The zoomview window does not popup under the cursor as it should.

BUGS

Please report all bugs to Technical Support at Human Designed Systems, Inc. Bug reports can be e-mailed to support@hds.com. Be sure to include the hdsview version and the ViewStation server code release version in the bug report.

COPYRIGHT

Copyright (C) 1993 Human Designed Systems, Inc. Portions adapted from ghostview, copyright 1992, Timothy O. Theisen.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

AUTHOR

Michael S. Hartman, Human Designed Systems, Inc.

2.6.10 Waitforserver Manual Page

This is the ViewStation's waitforserver local client man page.

NAME

waitforserver - utility to insure X server is accepting connections

SYNOPSIS

waitforserver

DESCRIPTION

waitforserver is a trivial X client which attempts repeatedly to open a connection to the default server, using XOpenDisplay(""). Each attempt will try for the "network timeout" time period, and if the call returns a NULL Display, the XOpenDisplay() function will be called again. Once a successful connection is made to the X Server, the client will exit with a status of 0.

This client is useful in ViewStation FX Startup Command Files, in order to wait for the X Server to get up and running before attempting to start any clients in the background. This is necessary since the Startup Command File is executed before the X Server is started.

A "chicken and egg" problem exists if you try to include waitforserver directly in a Startup Command File, since if it is started in the foreground it will never succeed in making an X connection, and the ViewStation will appear to "hang". This is because the X Server is not started until after the Startup Command File finishes executing, and the waitforserver client will not terminate (and therefore the Startup Command File will not finish executing) until a successful X connection is made. The solution to this problem is to make another shell script which consists of a single command to invoke the original Startup Command File in the background, and use this new shell script as the Startup Command File.

For example, if you wish to invoke the mwm and hdsdclock clients after the X Server is running, create a file with the following lines:

```
waitforserver

mwm &

hdsdclock &
```

You could name this file start_clients, and then create a Startup Command File called startup which consists of the single line:

```
sh <path>/start_clients &
```

where <path> is the local filesystem path to the start_clients shell script.

OPTIONS

waitforserver takes no command-line options, but will use the DISPLAY environment variable to specify a display other than the local X Server.

2.6.11 Xhost Manual Page

This is the ViewStation's xhost local client man page.

NAME

xhost - server access control program for X

SYNOPSIS

xhost [[+-]hostname ...]

DESCRIPTION

The xhost program is used to add and delete hosts to the list of machines that are allowed to make connections to the X server. This provides a rudimentary form of privacy control and security. It is only sufficient for a workstation (single user) environment, although it does limit the worst abuses. Environments which require more sophisticated measures should use the hooks in the protocol for passing authentication data to the server.

The server initially allows network connections only from programs running on the same machine or from machines listed in the file /etc/X*.hosts (where * is the display number of the server). The xhost program is usually run either from a startup file or interactively to give access to other users.

Hostnames that are followed by two colons (::) are used in checking DECnet connections; all other hostnames are used for TCP/IP connections.

OPTIONS

Xhost accepts the following command line options described below. For security, the options that effect access control may only be run from the same machine as the server.

[+]hostname

The given hostname (the plus sign is optional) is added to the list of machines that are allowed to connect to the X server.

-hostname

The given hostname is removed from the list of machines that are allowed to connect to the server. Existing connections are not broken, but new connection attempts will be denied. Note that the current machine is allowed to be removed; however, further connections (including attempts to add it back) will not be permitted. Resetting the server (thereby breaking all connections) is the only way to allow local connections again.

+

Access is granted to everyone, even if they aren't on the list of allowed hosts (i.e. access control is turned off).

-

Access is restricted to only those machines on the list of allowed hosts (i.e. access control is turned on).

nothing

If no command line arguments are given, the list of hosts that are allowed to connect is printed on the standard output along with a message indicating whether or not access control is currently enabled. This is the only option that may be used from machines other than the one on which the server is running.

FILES

/etc/X*.hosts

SEE ALSO

X(1), Xserver(1)

ENVIRONMENT

DISPLAY to get the default host and display to use.

BUGS

You can't specify a display on the command line because -display is a valid command line argument (indicating that you want to remove the machine named ''display'' from the access list).

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.

See X(1) for a full statement of rights and permissions.

AUTHORS

Bob Scheifler, MIT Laboratory for Computer Science,

Jim Gettys, MIT Project Athena (DEC).

2.6.12 Xmodmap Manual Page

This is the ViewStation xmodmap local client man page.

NAME

xmodmap - utility for modifying keymaps in X

SYNOPSIS

xmodmap [-options ...] [filename]

DESCRIPTION

The xmodmap program is used to edit and display the keyboard modifier map and keymap table that are used by client applications to convert event keycodes into keysyms. It is usually run from the user's session startup script to configure the keyboard according to personal tastes.

OPTIONS

The following options may be used with xmodmap:

-display display

This option specifies the host and display to use.

-help

This option indicates that a brief description of the command line arguments should be printed on the standard error channel. This will be done whenever an unhandled argument is given to xmodmap.

-grammar

This option indicates that a help message describing the expression grammar used in files and with -e expressions should be printed on the standard error.

-verbose

This option indicates that xmodmap should print logging information as it parses its input.

-quiet

This option turns off the verbose logging. This is the default.

-n

This option indicates that xmodmap should not change the mappings, but should display what it would do, like make(1) does when given this option.

-e expression

This option specifies an expression to be executed. Any number of expressions may be specified from the command line.

-pm

This option indicates that the current modifier map should be printed on the standard output.

-pk

This option indicates that the current keymap table should be printed on the standard output.

-pke

This option indicates that the current keymap table should be printed on the standard output in the form of expressions that can be fed back to xmodmap.

-pp

This option indicates that the current pointer map should be printed on the standard output.

-

A lone dash means that the standard input should be used as the input file.

The filename specifies a file containing xmodmap expressions to be executed. This file is usually kept in the user's home directory with a name like .xmodmaprc.

EXPRESSION GRAMMAR

The xmodmap program reads a list of expressions and parses them all before attempting to execute any of them. This makes it possible to refer to keysyms that are being redefined in a natural way without having to worry as much about name conflicts.

```
keycode NUMBER = KEYSYMNAME ...
```

The list of keysyms is assigned to the indicated keycode (which may be specified in decimal, hex or octal and can be determined by running the xev program in the examples directory).

```
keysym KEYSYMNAME = KEYSYMNAME ...
```

The KEYSYMNAME on the left hand side is translated into matching keycodes used to perform the corresponding set of keycode expressions. The list of keysym names may be found in the header file <X11/keysymdef.h> (without the XK_ prefix) or the keysym database /usr/lib/X11/XKeysymDB. Note that if the same keysym is bound to multiple keys, the expression is executed for each matching keycode.

```
clear MODIFIERNAME
```

This removes all entries in the modifier map for the given modifier, where valid name are: Shift, Lock, Control, Mod1, Mod2, Mod3, Mod4, and Mod5 (case does not matter in modifier names, although it does matter for all other names). For example, ''clear Lock'' will remove all any keys that were bound to the shift lock modifier.

```
add MODIFIERNAME = KEYSYMNAME ...
```

This adds all keys containing the given keysyms to the indicated modifier map. The keysym names are evaluated after all input expressions are read to

make it easy to write expressions to swap keys (see the EXAMPLES section).

```
remove MODIFIERNAME = KEYSYMNAME ...
```

This removes all keys containing the given keysyms from the indicated modifier map. Unlike add, the keysym names are evaluated as the line is read in. This allows you to remove keys from a modifier without having to worry about whether or not they have been reassigned.

```
pointer = default
```

This sets the pointer map back to its default settings (button 1 generates a code of 1, button 2 generates a 2, etc.).

```
pointer = NUMBER ...
```

This sets to pointer map to contain the indicated button codes. The list always starts with the first physical button.

Lines that begin with an exclamation point (!) are taken as comments.

If you want to change the binding of a modifier key, you must also remove it from the appropriate modifier map.

EXAMPLES

Many pointers are designed such that the first button is pressed using the index finger of the right hand. People who are left-handed frequently find that it is more comfortable to reverse the button codes that get generated so that the primary button is pressed using the index finger of the left hand. This could be done on a 3 button pointer as follows:

```
% xmodmap -e "pointer = 3 2 1"
```

Many editor applications support the notion of Meta keys (similar to Control keys except that Meta is held down instead of Control). However, some servers do not have a Meta keysym in the default keymap table, so one needs to be added by hand. The following command will attach Meta to the Multi-language key (sometimes labeled Compose Character). It also takes advantage of the fact that applications that need a Meta key simply need to get the keycode and don't require the keysym to be in the first column of the keymap table. This means that applications that are looking for a Multi_key (including the default modifier map) won't notice any change.

```
% xmodmap -e "keysym Multi_key = Multi_key Meta_L"
```

One of the more simple, yet convenient, uses of xmodmap is to set the keyboard's "rubout" key to generate an alternate keysym. This frequently involves exchanging Backspace with Delete to be more comfortable to the user. If the ttyModes

resource in xterm is set as well, all terminal emulator windows will use the same key for erasing characters:

```
% xmodmap -e "keysym BackSpace = Delete"
% echo "XTerm*ttyModes: erase ^?" | xrdb -merge
```

Some keyboards do not automatically generate less than and greater than characters when the comma and period keys are shifted. This can be remedied with xmodmap by resetting the bindings for the comma and period with the following scripts:

```
!
! make shift-, be < and shift-. be >
!
keysym comma = comma less
keysym period = period greater
```

One of the more irritating differences between keyboards is the location of the Control and Shift Lock keys. A common use of xmodmap is to swap these two keys as follows:

```
!
! Swap Caps_Lock and Control_L
!
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
```

The keycode command is useful for assigning the same keysym to multiple keycodes. Although unportable, it also makes it possible to write scripts that can reset the keyboard to a known state. The following script sets the backspace key to generate Delete (as shown above), flushes all existing caps lock bindings, makes the CapsLock key be a control key, make F5 generate Escape, and makes Break/Reset be a shift lock.

```
!
! On the HP, the following keycodes have key caps as listed:
```

```
!
! 101 Backspace
! 55 Caps
! 14 Ctrl
! 15 Break/Reset
! 86 Stop
! 89 F5
!
keycode 101 = Delete
keycode 55 = Control_R
clear Lock
add Control = Control_R
keycode 89 = Escape
keycode 15 = Caps_Lock
add Lock = Caps_Lock
```

ENVIRONMENT

DISPLAY to get default host and display number.

SEE ALSO

X(1), xev(1), Xlib documentation on key and pointer events

BUGS

Every time a keycode expression is evaluated, the server generates a MappingNotify event on every client. This can cause some thrashing. All of the changes should be batched together and done at once. Clients that receive keyboard input and ignore MappingNotify events will not notice any changes made to keyboard mappings.

Xmodmap should generate "add" and "remove" expressions automatically whenever a keycode that is already bound to a modifier is changed.

There should be a way to have the remove expression accept keycodes as well as keysyms for those times when you really mess up your mappings.

COPYRIGHT

Copyright 1988, 1989, 1990 Massachusetts Institute of Technology.

Copyright 1987 Sun Microsystems, Inc.

See X(1) for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium, rewritten from an earlier version by David Rosenthal of Sun Microsystems.

2.6.13 Xset Manual Page

This is the ViewStation xset local client man page.

NAME

xset - user preference utility for X

SYNOPSIS

xset [-display display] [-b] [b on / off] [b [volume [pitch [duration]]] [[-]bc] [-c] [c on / off] [c [volume]] [[-+]fp[-+=] path[,path[,...]]] [fp default] [fp rehash] [[-]led [integer]] [led on / off] [m[ouse] [accel_mult[/ accel_div] [threshold]]] [m[ouse] default] [p pixel color] [[-]r] [r on / off] [s [length [period]]] [s blank / noblank] [s expose / noexpose] [s on / off] [s default] [q]

DESCRIPTION

This program is used to set various user preference options of the display.

OPTIONS

-display display

This option specifies the server to use; see X(1).

b

The b option controls bell volume, pitch and duration. This option accepts up to three numerical parameters, a preceding dash(-), or a 'on / off' flag. If no parameters are given, or the 'on' flag is used, the system defaults will be used. If the dash or 'off' are given, the bell will be turned off. If only one numerical parameter is given, the bell volume will be set to that value, as a percentage of its maximum. Likewise, the second numerical parameter specifies the bell pitch, in hertz, and the third numerical parameter specifies the duration in milliseconds. Note that not all hardware can vary the bell characteristics. The X server will set the characteristics of the bell as closely as it can to the user's specifications.

bc

The bc option controls bug compatibility mode in the server, if possible; a preceding dash(-) disables the mode, otherwise the mode is enabled. Various pre-R4 clients pass illegal values in some protocol requests, and pre-R4 servers did not correctly generate errors in these cases. Such clients, when run against an R4 server, will terminate abnormally or otherwise fail to operate correctly. Bug compatibility mode explicitly reintroduces certain bugs into the X server, so that many such clients can still be run. This mode should be used with care; new application development should be done with this mode disabled. The server must support the MIT- SUNDRY-NONSTANDARD protocol extension in order for this option to work.

c

The c option controls key click. This option can take an optional value, a preceding dash(-), or an 'on/off' flag. If no parameter or the 'on' flag is given, the system defaults will be used. If the dash or 'off' flag is used, keyclick will be disabled. If a value from 0 to 100 is given, it is used to indicate volume, as a percentage of the maximum. The X server will set the volume to the nearest value that the hardware can support.

fp= path,...

The fp= sets the font path to the directories given in the path argument. The directories are interpreted by the server, not by the client, and are server-dependent. Directories that do not contain font databases created by mkfontdir will be ignored by the server.

fp default

The default argument causes the font path to be reset to the server's default.

fp rehash

The rehash argument causes the server to reread the font databases in the current font path. This is generally only used when adding new fonts to a font directory (after running mkfontdir to recreate the font database).

-fp or fp-

The -fp and fp- options remove elements from the current font path. They must be followed by a comma-separated list of directories.

+fp or fp+

This +fp and fp+ options prepend and append elements to the current font path, respectively. They must be followed by a comma-separated list of directories.

led

The led option controls the keyboard LEDs. This controls the turning on or off of one or all of the LEDs. It accepts an optional integer, a preceding dash(-) or an 'on' flag is given, all LEDs are turned on. If a preceding dash or the flag 'off' is given, all LEDs are turned off. If a value between 1 and 32 is given, that LED will be turned on or off depending on the existence of a preceding dash. A common LED which can be controlled is the ''Caps Lock'' LED. ''xset led 3'' would turn led #3 on. ''xset -led 3'' would turn it off. The particular LED values may refer to different LEDs on different hardware.

m

The m option controls the mouse parameters. The parameters for the mouse are 'acceleration' and 'threshold'. The acceleration can be specified as an integer, or as a simple fraction. The mouse, or whatever pointer the machine is connected to, will go 'acceleration' times as fast when it travels more than 'threshold' pixels in a short time. This way, the mouse can be used for precise alignment when it is moved slowly, yet it can be set to travel across the screen in a flick of the wrist when desired. One or both parameters for the m option can be omitted, but if only

one is given, it will be interpreted as the acceleration. If no parameters or the flag 'default' is used, the system defaults will be set.

p

The p option controls pixel color values. The parameters are the color map entry number in decimal, and a color specification. The root background colors may be changed on some servers by altering the entries for BlackPixel and WhitePixel. Although these are often 0 and 1, they need not be. Also, a server may choose to allocate those colors privately, in which case an error will be generated. The map entry must not be a read-only color, or an error will result.

r

The r option controls the autorepeat. If a preceding dash or the 'off' flag is used, autorepeat will be disabled. If no parameters or the 'on' flag is used, autorepeat will be enabled.

s

The s option lets you set the screen saver parameters. This option accepts up to two numerical parameters, a 'blank / noblank' flag, an 'expose / noexpose' flag, an 'on / off' flag, or the 'default' flag. If no parameters or the 'default' flag is used, the system will be set to its default screen saver characteristics. The 'on / off' flags simply turn the screen saver functions on or off. The 'blank' flag sets the preference to blank the video (if the hardware can do so) rather than display a background pattern, while 'noblank' sets the preference to display a pattern rather than blank the video. The 'expose' flag sets the preference to allow window exposures (the server can freely discard window contents), while 'noexpose' sets the preference to disable screen saver unless the server can regenerate the screens without causing exposure events. The length and period parameters for the screen saver function determines how long the server must be inactive for screen saving to activate, and the period to change the background pattern to avoid burn in. The arguments are specified in seconds. If only one numerical parameter is given, it will be used for the length.

q

The q option gives you information on the current settings.

These settings will be reset to default values when you log out.

Note that not all X implementations are guaranteed to honor all of these options.

SEE ALSO

X(1), Xserver(1), xmodmap(1), xrdb(1), xsetroot(1)

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.

See X(1) for a full statement of rights and permissions.

AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science

David Krikorian, MIT Project Athena (X11 version)

2.6.14 FDFORMAT Disk Format Utility

This is the manual page for the ViewStation's floppy disk formatting utility.

NAME

fdformat - format floppy diskettes

SYNOPSIS

fdformat [-l] [-n] [-v label] device

DESCRIPTION

fdformat is a utility for formatting floppy diskettes. Al new blank diskettes must be formatted before they can be used. fdformat formats and verifies each track on the diskette, and terminates if it finds any bad sectors. All existing data on the diskette, if any, is destroyed by formatting.

By default, fdformat formats a high density diskette with a capacity of 1.44 megabytes. Use the -l option to format low density diskettes (720K capacity). Note: it is not possible to put a low density format onto a high density floppy diskette. High density diskettes can be recognized by the high density detect hole in the lower right corner of the diskette, opposite the write protect hole in the lower left corner.

A MS-DOS file system is installed on the diskette after the format is done.

OPTIONS

-v Put a MS-DOS label on the disk after formatting it.

-l Format a low density (720K) diskette.

-n Skip verifying the floppy diskette after formatting.

NOTES

The MS-DOS file system installed by fdformat does not have the necessary system files to make it bootable on a PC.

FILES

/dev/fd1

/dev/fd2

etc.

BUGS

Currently bad sector mapping is not supported on floppy diskettes. Therefore, a diskette is unusable if fdformat finds an error (bad sector).

2.6.15 PEX 3D Server Extension

This section describes the PEX-3D server extension. Note that this is an extension of the server code rather than a local client. This extension is loaded with entries in the clients.config file, described in Section 2.1.6.

PEX-3D Extension V1.0

Introduction

This extension is based on the X Consortium PEX 5.1 protocol standard.

The PEX extension is a 3-D graphics extension to the HDSware X11 server.

2.6.15.1 Installation of the HDS PEX-3D Extension

The PEX extension is a downloaded extension controlled by the "hds.serverExtensions" resource in the */FXUtilities/hds-fx/config/clients.config* file. There are two files, named *X3D-PEX-Vnn* and *X3D-PEX-Vnn-D*, which are located in */FXUtilities/hds-fx/extensions*. These files are for use with ViewStation server versions Vnn.hds and Vnn-D.hds respectively. These two versions of the ViewStation's server code differ in their ability to download local clients (designated by the -D option). The appropriate PEX-3D extension files can be used with either version since these extensions are loaded as an integral part of the server code.

HDS PEX-3D utilizes the PEX_FONTPATH environment variable to set the PEX font path which is separate from the X11 font path. The default value is set to */FXUtilities/hds-fx/fonts/PEX* which contains two compressed PEX fonts, Roman and Roman_m.

The PEX-3D extension requires the NFS file transfer protocol in order to read fonts. In addition the extension needs to read the Roman font in order to initialize. This means that the user would need to set the HDSware Host Access Protocol to NFS prior to starting the server.

ViewStation Requirements

The PEX-3D extensions are available with the HDSware 3.1.1.

HDS PEX-3D is supported on all Viewstation architectures. While it is possible to load the HDS PEX-3D extension with the Vnn.hds (non-downloadable) server version into a ViewStation containing only 4 MB of memory there is only about 500 KB of memory left. It is recommended that at least 8MB of memory be installed on the ViewStation when using the PEX-3D extension.

2.6.15.2 HDS PEX-3D Limitations

- Interior styles "Pattern" and "Hatch" are not supported. These styles will be rendered using the "Solid" interior style.

- PickOne methods "ClosestZ", "VisibleAny", and "VisibleClosest" are not supported. These methods will default to "Last".
- PickAll method "Visible" is not supported. This method will default to "All".
- Curve approximation methods "DC_ChordalSize", "DC_ChordalDev", "WCS_Relative", "NPC_Relative", and "DC_Relative" are not supported. These methods are rendered as Implementation Dependent which defaults to ConstantBetweenKnots with a tolerance of 10.0.
- Surface interpolation methods "DotProduct" and "Normal" are not supported. These methods default to the "Color" (Gouraud).
- Surface approximation methods "DC_ChordalSize", "DC_ChordalDev", "WCS_Relative", "NPC_Relative", and "DC_Relative" are not supported. These methods are rendered as Implementation Dependent which defaults to ConstantBetweenKnots with a tolerance of 10.0 for u and v.
- Color types "CIEFloat", "HDSVFloat", "HLSFloat", "RGBInt8", and "RGBInt16" are not supported. Attempts to use these types will result in a ColorTypeError.
- Float formats "IEEE_754_64", and "DEC_D_Floating" are not supported.
- HLHSR modes "ZBufferId", "HiddenLineOnly", "Scanline", and "Painters" are not supported. These modes default to "Off".
- Prompt echo types "EchoStructure" and "EchoNetwork" are not supported. The types default to "EchoPrimitive".
- Color approximation types "CIE", "HSV", "HLS" and "YIQ" are not supported. Attempts to use these types will exit with a BadValue.
- "gdp", "gdp3", and "gse" types are not supported.
- Rendering color models "CIE", "HSV", and "HLS" are not supported. Requests to use these models will silently exit.
- Parametric surface characteristics "MC_LevelCurves", and "WC_Levelcurves" are not supported. These types will default to "None".
- Transparency is not supported.

2.6.15.3 HDS PEX-3D Functional Description

These are the HDS PEX-3D supported Enumerated Types

- Marker Type - there are 5 marker types supported.
  1. Dot
  2. Cross
  3. Asterisk
  4. Circle
  5. X
- Annotation text style - there are two types supported.
  1. Not Connected
  2. Connected

- Interior Style - there are three types supported.
  1. Hollow
  2. Solid
  3. Empty
- Line Type - there are four types supported.
  1. Solid
  2. Dashed
  3. Dotted
  4. DashDot
- Pick Device Type - there are two types supported.
  1. DC_HitBox
  2. NPC_HitVolume
- Pick One Methods - there are two types supported.
  1. 1
  2. Last
- Pick All Methods - there are two types supported.
  1. 1
  2. All
- Polyline Interpolation Methods - there are 2 types supported.
  1. None
  2. Color
- Curve Approximation Method - there are 6 types supported.
  1. Implementation Dependent - HDS PEX-3D utilizes the ConstantBetweenKnots method with a tolerance of 10.0
  2. ConstantBetweenKnots
  3. WCS_ChordalSize
  4. NPC_ChordalSize
  5. WCS_ChordalDev
  6. NPC_ChordalDev
- Reflection Methods - there are four types supported.
  1. NoShading
  2. Ambient
  3. Diffuse
  4. Specular
- Surface Interpolation Methods - there are two types supported.
  1. None

2. Color

- Surface Approximation Methods - there are 6 types supported.
  1. Implementation Dependent - HDS PEX-3D utilizes the ConstantBetweenKnots method with u and v tolerances of 10.0
  2. ConstantBetweenKnots
  3. WCS_ChordalSize
  4. NPC_ChordalSize
  5. WCS_ChordalDev
  6. NPC_ChordalDev
- Trim Curve Approximation Methods - there are 2 methods supported.
  1. Implementation Dependent - HDS PEX-3D utilizes the ConstantBetweenKnots method with a tolerance of 10.0
  2. ConstantBetweenKnots
- Model Clip Operator - there are two types supported.
  1. Replace
  2. Intersection
- Light Types - there are 4 types supported.
  1. Ambient
  2. WCS_Vector
  3. WCS_Point
  4. WCS_Spot
- Color Type - there are two types supported.
  1. Indexed
  2. RGBFloat
- Float Format - there are two formats supported.
  1. IEEE_754_32
  2. DEC_F_Floating
- HLHSR modes - there are 2 modes supported.
  1. Off
  2. ZBuffer
- Prompt Echo Type - there is one type supported.
  1. EchoPrimitive.
- Display Update Modes - there are 5 modes supported.
  1. VisualizeEach
  2. VisualizeEasy

3. VisualizeNone
4. SimulateSome
5. VisualizeWhenever

- Color Approximation Types - there are two types supported.
  1. ColorSpace
  2. ColorRange
- Color Approximation Models - there is one model supported.
  1. RGB
- Escapes - one escape is supported.
  1. "SetEchoColor"
- Rendering Color Models - there is one model supported.
  1. RGB
- Parametric Surface Characteristics - there are three types supported.
  1. None
  2. Implementation Dependent - HDS PEX-3D utilizes IsoparametricCurves with the number of u and v curves set to 20.
  3. IsoparametricCurves

2.6.15.4 HDS PEX-3D Implementation Dependent Constants

Name	Value
Dithering	True
Maximum Edge Width	~((unsigned long)0)
Minimum Edge Width	1
Nominal Edge Width	1
Supported Edge Widths	~((unsigned long)0)
Maximum Line Width	~((unsigned long)0)
Minimum Line Width	1
Nominal Line Width	1
Supported Line Widths	~((unsigned long)0)
Maximum Marker Size	~((unsigned long)0)
Minimum Marker Size	1
Nominal Marker Size	1
Supported Marker Sizes	~((unsigned long)0)
Model Clip Planes	64
Name Set Names	256
Max Non Ambient Lights	64
Maximum Nurb Order	10

Maximum Trim Curve Order	6
Best Color Approximation Values	ColorApproxAnyValues
Transparency support	False
Double Buffering support	True
Max Hits Event Support	True
ChromaticityRedU	0.628
ChromaticityRedV	0.346
LuminanceRed	1.0
ChromaticityGreenU	0.268
ChromaticityGreenV	0.588
LuminanceGreen	1.0
ChromaticityBlueU	0.150
ChromaticityBlueV	0.070
LuminanceBLue	1.0
ChromaticityWhiteU	0.313
ChromaticityWhiteV	0.329
LuminanceWhite	1.0

2.6.15.5 HDS PEX-3D Lookup Tables

Name	Maximum # of Entries	# Predefined
Line Bundle	20	1
Marker Bundle	20	1
TextBundle	20	1
Interior Bundle	20	1
Edge Bundle	20	1
Pattern	0	0
Color	256	8
TextFont	4	1
View	6	1
Light	16	1
Depth Cue	6	1
ColorApprox	6	1

2.6.15.6 HDS PEX-3D Lookup Table default predefined values

Name	Index	Values
LineBundle	0	

Line Type		1
polyline_interp		1
curve_approx		1
curve_tolerance		1.0
Line Width		1
line_color type		indexed
line_color index		1
MarkerBundle	0	
marker_type		asterisk
marker_scale		1.0
marker_color type		indexed
marker_color index		1
TextBundle	1	
textFontIndex		1
textPrecision		StringPrecision
charExpansion		1.0
charSpacing		0.0
textColor type		indexed
textColor index		1
InteriorBundle	1	
interiorStyle		InteriorStyleHollow
interiorStyleIndex		1
reflectionModel		ReflectionNoShading
surfaceInterp		SurfaceInterpNone
bfInteriorStyle		InteriorStyleHollow
bfInteriorStyleIndex		1
bfReflectionModel		ReflectionNoShading
surfaceApprox.approxMethod		1
surfaceApprox.uTolerance		1.0
surfaceApprox.vTolerance		1.0
surfaceColor type		indexed
surfaceColor index		1

bfSurfaceColor type		indexed
bfSurfaceColor index		1
reflectionAttr.ambient		1.0
reflectionAttr.diffuse		1.0
reflectionAttr.specular		1.0
reflectionAttr.specularConc		0.0
reflectionAttr.transmission		0.0
reflectionAttr.specularColor type		indexed
reflectionAttr.specularColor index		1
bfreflectionAttr.ambient		1.0
bfreflectionAttr.diffuse		1.0
bfreflectionAttr.specular		1.0
bfreflectionAttr.specularConc		0.0
bfreflectionAttr.transmission		0.0
bfreflectionAttr.specularColor type		indexed
bfreflectionAttr.specularColor ind		1
Edge Bundle	1	
edges		off
edgeType		SurfaceEdgeSolid
edgeWidth		1.0
edgeColor type		indexed
edgeColor index		1
TextFont	1	
numFonts		1
font		defaultFont (Roman)
(16 character set id's allowed per Font group)		
View	0	
clipFlags		ClipXY | ClipBack | ClipFront
clipLimits		(0.0,0.0,0.0),(1.0,1.0,1.0)
orientation		identity matrix
mapping		identity matrix

Light	1	
lightType		Ambient
direction		(0.0,0.0,0.0)
point		(0.0,0.0,0.0)
concentration		0.0
spreadAngle		0.0
attenuation1		0.0
attenuation2		0.0
lightColor type		RgbFloat
lightColor red		1.0
lightColor green		1.0
lightColor blue		1.0
DepthCue	0	
mode		Off
frontPlane		1.0
backPlane		0.0
frontScaling		1.0
backScaling		0.5
depthCueColor type		indexed
depthCueColor index		1
Color Approximation	0	
approxType		ColorSpace
approxModel		ColorApproxRGB
max1		5
max2		5
max3		5
dither		Off
mult1		1
mult2		6
mult3		36
weight1		1.0
weight2		1.0
weight3		1.0

basePixel 16

Color 0

Index	Type	Red	Green	Blue
0	RgbFloat	0.0	0.0	0.0
1	RgbFloat	1.0	1.0	1.0
2	RgbFloat	1.0	0.0	0.0
3	RgbFloat	0.0	1.0	0.0
4	RgbFloat	0.0	0.0	1.0
5	RgbFloat	1.0	1.0	0.0
6	RgbFloat	0.0	1.0	1.0
7	RgbFloat	1.0	0.0	1.0

2.6.16 Xscope Manual Page

This is the manual page for xscope. It runs as a local client on the ViewStation.

NAME

xscope - X Window Protocol Viewer

SYNOPSIS

xscope [option] ...

DESCRIPTION

xscope sits in-between an X11 client and an X11 server and prints the contents of each request, reply, error, or event that is communicated between them. This information can be useful in debugging and performance tuning of X11 servers and clients.

To operate, xscope must know the host, port, and display to use to connect to the X11 server. In addition, it must know the port on which it should listen for X11 clients.

Two cases are common:

(1) The X11 server is on the same host as xscope. In this case, the input port for xscope should be selected as an X11 server on a different display, and the client DISPLAY argument adjusted to select xscope. For example, if the X11 server is on port 6000, display 1, then xscope can use port 6002 as its input port. The client can use display 1 for direct access to X11 or display 2 for access to xscope. Note that running as a ViewStation local client xscope is on the same host (the Viewstation) as the server. The default host, input port, and output ports are already set correctly, so the user need only set the verbosity level desired. To connect to xscope the user would then set his DISPLAY environment variable to ViewStation IP:1 before running the client they wish to test.

(2) The X11 server is on a different host than xscope. In this case the same input and output ports can be used, and the host component of the DISPLAY is used to select xscope or X11.

ARGUMENTS

-i<input-port>

Specify the port that xscope will use to take requests from clients (defaults to 1). For X11, this port is automatically biased by 6000.

-o<output-port>

Determines the port that xscope will use to connect to X11 (defaults to 0). For X11, this port is automatically biased by 6000.

-h<host>

Determines the host that xscope will use to find its X11 server.

-d<display>

Defines the display number. The display number is added to the input and output port to give the actual ports which are used by xscope.

-q

Quiet output mode. Gives only the names of requests, replies, errors, and events, but does not indicate contents.

-v<print-level>

Determines the level of printing which xscope will provide. The print-level can be 0 (same as quiet mode), 1, 2, 3, 4. The larger numbers give more and more output. For example, a successful setup returns a string which is the name of the vendor of the X11 server. At level 1, the explicit field giving the length of the string is suppressed since it can be inferred from the string. At level 2 and above the length is explicitly printed.

EXAMPLES

xscope -v4 -hcleo -d0 -o0 -i1

This command would have xscope communicate with an X11 server on host ''cleo'', display 0; xscope itself would be available on the current host as display 1 (display of 0 plus the 1 of -i1). Verbose level 4.

xscope -q -d1 -o1 -o3

The X11 server for the current host, display 2 (1 for -d1 plus 1 for -o1) would be used by xscope which would run as display 4 (1 for -d1 plus 3 for -o3). Quiet mode (verbose level 0).

SEE ALSO

X(1), X11 Protocol document (doc/Protocol/X11.protocol)

2.6.17 PING Manual Page

Ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (''pings'') have an IP and ICMP header, followed by a ''struct timeval'' and then an arbitrary number of ''pad'' bytes used to fill out the packet. The options are as follows:

-c count

Stop after sending (and receiving) count ECHO_RESPONSE packets.

-d

Set the SO_DEBUG option on the socket being used.

-i wait

Wait wait seconds between sending each packet. The default is to wait for one second between each packet.

-l preload

If preload is specified, ping sends that many packets as fast as possible before falling into its normal mode of behavior.

-n

Numeric output only. No attempt will be made to lookup symbolic names for host addresses.

-p pattern

You may specify up to 16 ''pad'' bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, ''-p ff'' will cause the sent packet to be filled with all ones.

-q

Quiet output. Nothing is displayed except the summary lines at startup time and when finished.

-r

Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by routed(8)).

-s packetsize

Specifies the number of data bytes to be sent. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data.

-v

Verbose output. ICMP packets other than ECHO_RESPONSE that are received are listed.

When using ping for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be "pinged". Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation, although the round trip time of these packets is used in calculating the minimum / average / maximum round-trip time numbers. When the specified number of packets have been sent (and received) or if the program is terminated with a SIGINT, a brief summary is displayed.

Refer also to Section 1.4.5.3 for information about the ViewStation's local route command.

2.6.18 HDSshare Manual Page

This client is available with HDSware version 3.2. You must use the special configuration files /HDSware/hds-fx/config/config.ext (or /HDSware/hds-fx/config/config.ext.dsk) and /HDSware/hds-fx/config/clients.fx in order the hdsshare client.

NAME

hdsshare - a simple multi-headed ViewStation

SYNTAX

hdsshare [toolkit options] [-nobuttons] [-md] -rd RemoteDisplay [-rd RemoteDisplay ...]

DESCRIPTION

This Motif utility is used to logically control multiple X servers from one X server's keyboard and mouse, and switch between X servers. This utility requires the XTrap extension to be running on both the master X server and on all remote (slave) X servers.

OPTIONS

-no buttons

Do not use Motif buttons to hop the cursor from one display to another. If this option is specified, the -md option must also be present to identify the left-right order of the displays.

-md

This option enables automatic cursor hopping by moving the cursor to the left or right edge of the screen. The displays are assumed to be physically ordered from left to right in the same order that they are given on the command line. The -md option is a place holder for the master display's position in the display order. For example,

hdsshare -rd slave1:0 -rd slave2:0 -md -rd slave:0

specifies that the order of the display screens, from left to right, is slave1, slave2, the master display, and then slave3.

-rd RemoteDisplay

Specified in the same manner as the control display, for each remote display.

Example usage

hdsshare -rd remote:0

hdsshare -display local:0 -rd remote:0 -md

hdsshare -nobuttons -md -rd near:0 -rd far:0

ACKNOWLEDGMENTS

Similar to functionality of "xhydra" by Robert Chesler, but developed independently before the author was aware of xhydra.

SEE ALSO

X(1), xtrap(1)

2.6.18.1 Using the HDSshare Client

HDSshare is a local ViewStation client that lets you share a keyboard and mouse between several ViewStations. This is sometimes convenient where ViewStations are used without their own input devices (such as process control installations or in isolated environments such as clean rooms, etc.).

Starting HDSshare

HDSshare is included with the HDSperx clients in the HDSware 3.2 distribution but is not included as a default client on the HDSperx menu. You must make two entries to the clients configuration file in order to run HDSshare.

Make these changes in the ./hds-fx/config/clients.config file:

1. Load the Xtrap downloadable extension:

Your line should look like this (this may vary slightly by HDSware version)

`hds.serverExtensions: /FXUtilities/hds-fx/extensions/MIT-SHM /FXUtilities/hds-fx/extensions/DEC-XTRAP` *(use a single line for the extensions, use a space between entries)*

2. Add a line for the HDSshare client:

```
hds.client.hdsshare.path: /FXUtilities/hds-fx/bin/hdsshare
```

3. Reboot ALL of the HDSshare ViewStations so the server extension is loaded on each.

Refer to the hdsshare man page for details of HDSshare operation.

HDSshare requires that the Xtrap extension be running on all of the ViewStations controlled by HDSshare, so they should all use this clients.config file and all be rebooted with the extension.

Start HDSshare from the ViewStation's Console window

You must specify one or more remote displays for HDSshare to manage. Suppose your ViewStation is named "hds40" and you want to control ViewStations named "hds42" and "hds46". In the Console window enter:

```
hdsshare -rd hds42:0 -rd hds46:0 &
```

which gives you three name tiles (hds40, hds42, and hds46) in the HDSshare window. Note that HDSshare should run in the background.

The local ViewStation is the default controller; you can also specify another ViewStation as the controller with the -display argument:

```
hdsshare -display local:0 -rd remote1:0 -rd remote2:0 -rd
remote3:0 &
```

HDSshare Operation

When HDSshare starts, it puts up a window with a name tile for the local display and for each remote ViewStation display you've specified. You can use the keyboard and mouse for any of those displays just by moving your local mouse to the appropriate tile and clicking on it.

This moves the mouse and keyboard control to the remote ViewStation's monitor screen (which may be on your desktop or may be viewable through a clean-room window, etc.). It is just as if the keyboard and mouse were connected to the remote ViewStation base. It is not necessary for the remote display to have either a keyboard or a mouse; if they are present, the HDSshare display overrides their function when it has control. The HDSshare window with its name tiles appears on the new display so you can go to any other HDSshare ViewStation from the new display.

If you use the -nobuttons command line option, HDSshare doesn't put up tiles for each display but allows you to move the cursor smoothly from one display to another as if they were a single screen. If you use the -nobuttons option, you must use the -md option to locate the master display; the other displays are assumed to be left-to-right as they appear on the command line.

For example the command line:

```
hdsshare -nobuttons -rd slave1:0 -rd slave2:0 -md -rd
slave3:0&
```

specifies the physical arrangement as slave1, slave2, master, slave3. In this example you move the cursor from one display to another without use of the name tiles.

2.6.19 Xpmroot Manual Page

NAME

xpmroot - Sets the root window of the current X display to an Xpm pixmap

SYNOPSIS

xpmroot file

DESCRIPTION

xpmroot reads the Xpm file specified in the command line and displays it in the root window.

BUGS

Repeated use of xpmroot with different xpm pixmaps will use up slots in your color table pretty darn fast.

AUTHOR

Rob Nation

HDS ViewStation: Multimedia Operations

This chapter discusses the operation of the HDS ViewStation's audio and video local clients.

2.0 Section 2.0 - HDS ViewStation Local Clients

2.7 HDS ViewStation Multimedia Operations

The HDS ViewStation supports audio and video multimedia options. These are local ViewStation clients, and depending on the options, require additional hardware and/or software options as discussed below.

2.7.1 HDS ViewStation Audio Operation

This section discusses hardware and software requirements for using the Audio features of the HDS ViewStation.

2.7.1.1 Audio Operation Hardware

The HDS ViewStation Audio option includes a PC Card board with an Audio board installed. It also includes an internal speaker and audio input and audio output jacks on the ViewStation's backpanel.

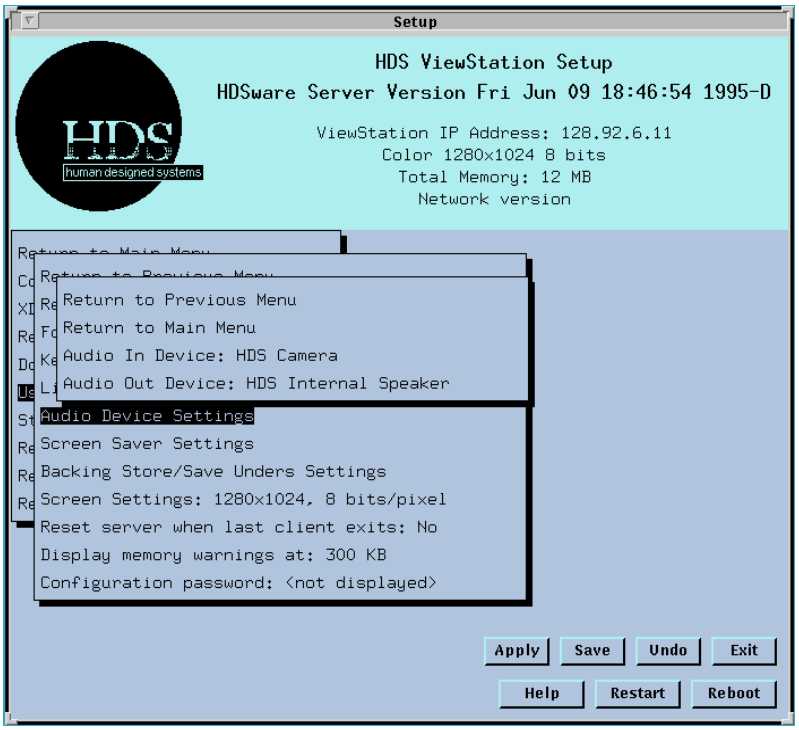
Additional audio equipment is available from HDS, including an integrated speaker and microphone with echo cancelling. Microphones and stereo speakers (external) are available from HDS. Other compatible audio equipment is available from commercial suppliers.

2.7.1.2 Audio Device Setting Menu in Setup Mode

When you start an Audio or Video client without first selecting an audio input device, a message appears requesting the setting:



The **Audio Device Setting** line in the User Preferences menu brings up a sub-menu for selecting the audio input and output devices you are using on your ViewStation.



The **Audio In Device** line toggles between several input devices, HDS Camera (with built-in microphones), HDS headset, the CallPort integrated Speaker / Microphone (with echo-cancelling), HDS microphone, HDS tuner, VCR, and Line Level Audio. This setting selects the appropriate input level for the HDS audio device. These settings can also be made in the HDS audio client.

If you are using your own audio equipment, you will have to determine which input level settings are appropriate.

The **Audio Out Device** line toggles between several output devices, HDS internal speaker, HDS external speakers, HDS headset, and the CallPort integrated Speaker / Microphone (with echo-cancelling).

If you are using your own audio equipment, you will have to determine which output level settings are appropriate.

BE SURE TO SAVE THESE SETTINGS in NVM so they are available to the Audio client when it is started.

2.7.1.3 ViewStation Audio Client Overview

This section includes the HDS audio man pages as well as associated audio programs that run on the HDS ViewStation, as follows:

hdsaudio hdsplay hdsrecord auhds

2.7.1.4 HDSperx Audio Control client - hdsaudio

The HDS audio program is downloadable ViewStation local Motif client which provides a control panel which allows the user to set various audio parameters for presentation on the ViewStation. The HDS audio client executable is called *./hds-fx/bin/hdsaudio* on the HDSware distribution (version 3.0 and higher).

The Audio Control client is shown below. The man pages for hdsaudio are in Section 2.7.6.1.



The Audio Control client contains the following entries:

- Play Volume: This sets the play volume for the audio output
- Play Balance: This sets the right/left balance for stereo sound.
- Record Volume: This sets the recording volume.
- Record Balance: This sets the recording balance.

• Monitor Volume: This controls the amount of input audio to be sent directly to the audio output. This is usually zero.

• Pause Play and Pause Record buttons: These permit pausing the play or record functions. If paused, the buttons change to Resume Play and Resume Record.

• Audio Out Device: This allows you to specify the Audio Output device you have installed. Options are: HDS Internal Speakers, External Speakers, HDS Headset, and CallPort Spkr / Mic. This selection can also be made in Setup Mode from the User Preferences menu.

• Audio In Device: This lets you select the Audio Input device you have installed. Options are: Unspecified, HDS ViewCam, HDS Headset, CallPort Spkr / Mic, HDS Microphone, HDS Tuner, VCR, and Line Level Audio. This selection can also be made in Setup Mode from the User Preferences menu. Refer to Section 2.7.1.2 for more detail.

The lower part of the client screen reports settings selected by client resources or on the audio command line, including sampling rates and precision. Refer to the HDS audio manual pages for details.

2.7.1.5 Starting the HDS Audio Client

The HDS audio client, like other HDSperx clients, can be started in several ways:

• Choosing the 'HDS Audio' selection from a local window manager's 'HDSperx' menu.

• Starting directly from the HDS console window via the 'hdsaudio' command.

• Starting remotely from a host via the Remote Shell protocol.

(e.g. **rsh -n testfx "hdsaudio &"**)

See Section 2.1 in this manual for further information on starting local clients.

The **hdsaudio** client can be used in conjunction with the HDS NetAudio server.

2.7.1.6 HDSperx NetAudio compatible server - auhds

The NetAudio server has a number of functions and features. Some of these NetAudio clients are available with HDSware in the *./hds-fx/contrib/netaudio* directory. More details are in the auhds and NetAudio man pages in Sections 2.7.6.3 and 2.7.6.4.

The downloadable NetAudio server, called **auhds,** is not started automatically. If you intend to use the NetAudio protocol to access the ViewStation's audio capabilities, you must first start the **auhds** Audio Server task. If you intend to use the NetAudio server regularly, you should consider starting it automatically with a command in the ViewStation's Startup Command file:

auhds &

(The client configuration database automatically adds the '-aa' option, by default. Refer to the *./hds-fx/config/clients.config* file.)

As an alternative, the audio server can be started via Remote Shell with:

rsh -n <ViewStation name> "auhds &"

As with other ViewStation messages, any error messages can be examined with the HDS Console **log** and **logpg** commands. The HDS NetAudio server executable is called *./hds-fx/bin/auhds* on the HDSware distribution.

2.7.1.7 HDSperx Audio File Playback program - hdsplay

The **hdsplay** program plays an individual sound file directly to the ViewStation's audio interface. This program does not use the NetAudio protocol, or require the **auhds** server to be running. It can be used by many applications which require a simple means to output a sound.

The **hdsplay** program reads a file specified on its command line (or from its standard input). The Sun / Next audio format is supported directly. Raw audio data files can be played by using the 'hdsplay' command line options to set the sample rate, encoding (linear, ulaw), sample precision (8 or 16 bit), and number of channels (1 or 2).

You should type **hdsplay -help** in the HDS Console for a list of **hdsplay** options.

Examples:

rsh <VS name> hdsplay - < sample.au

rsh <VS name> hdsplay / host / usr / demo / sounds / spacemusic.au

rsh <VS name> hdsplay -s 44100 -p 16 -e linear -c 2 - < cdquality.raw

NOTE: The **hdsplay** program cannot be used if the NetAudio server is running. If the **auhds** server is running, a NetAudio program such as **auplay** should be used to play sound files.

The **hdsplay** program executable is called *./hds-fx/bin/hdsplay* on the HDSware distribution. More details are available in the hdsplay man page in Section 2.7.6.2.

2.7.1.8 HDSperx Audio File Recording program - hdsrecord

The hdsrecord program records an audio file from the ViewStation's audio input and stores it in a file on a host computer. Various recording options and characteristics can be selected from the command line. These are described in the HDSrecord man page in Section 2.7.6.11.

You should run hdsrecord from a host computer with a remote shell command to the ViewStation. You can stop the recording either by specifying a time limit (with the -t option on the command line) or by pressing *Ctrl-C*.

Some examples of recording sound files from the ViewStation are:

rsh <VS name> hdsrecord > audiofile.au

rsh <VS name> hdsrecord -s 44100 -p 16 -e linear -c 2 > cdquality.au

2.7.1.9 Using ViewStation Audio with Application Programs

Application programs that utilize audio files usually try to start a separate "audio player" on the desktop display that actually plays the audio file. This audio player usually has a name, and running the ViewStation's audio usually just

means creating a shell script that substitutes the ViewStation's audio player for the application program's audio player name.

For instance, the Mosaic Internet browser uses an audio player called "showaudio". Mosaic looks for this program in your path and sends showaudio the file when an audio excerpt needs to be played.

This is a script to allow Mosaic and its "showaudio" player to play an audio file on the ViewStation. Name this file "showaudio" and put it in your path so Mosaic can find it when needed.

```
#!/bin/sh
#
# Script to play an audio file on the HDS ViewStation.
#
# Usage: showaudio filename
#
# The ViewStation hostname is determined from the DISPLAY
# environment variable. The IP address cannot be used
# directly with the 'rsh' command.
#
remote_shell_cmd=rsh
remhost=`expr "$DISPLAY" : "\([^:]*\).*"`
if [ `expr "$DISPLAY" : "[0-9]"` -ne 0 ]
then
 remhost=`grep "^$remhost[^0-9]" /etc/hosts | awk '{print
$2}'`
fi
exec $remote_shell_cmd $remhost hdsplay - < $1
```

This example with Mosaic is typical of many application programs. Check the audio process used by your application and make the appropriate adjustments to these examples.

2.7.1.10 ViewStation Audio Specifications

The HDS ViewStation Multimedia unit has both input and output jacks on the back panel, using stereo 3.5mm connectors.

Input jack specifications:

There are several Audio Input device selections:

• Line Level Audio - this is line-level input, which accepts 2 V (RMS) analog signals. This is the preferred setting.

• HDS ViewCam - this is a line-level input setting.

• HDS Headset - this is a boosted microphone input for dynamic type unpowered microphones, 2-10 mV input. These microphones must have JP1 / JP2 set in the 1-2 position (this is the default position so you should not have to change it).

• CallPort Spkr / Mic - this is a line-level input setting.

• HDS Microphone - this is a boosted microphone input (see "Headset" above).

• HDS Tuner - this is a battery / phantom powered microphone (100 mV input).

• VCR - this is a line-level input.

• Unspecified - this should generally not be used; if there is a question of appropriate levels for your device, choose the line-level input setting.

If you are connecting your own audio device, select the entry that matches the specifications of your device.

Output jack specifications:

The Audio Output devices are as follows:

• HDS Internal Speakers - 2 W output at 8 Ohms. Harmonic distortion is 0.055% with power at 1 W.

• External Speakers - self-powered for mono or stereo output

• HDS Headset - 0.75 W at 1 V (peak-to-peak) when driving loads of 25-32 Ohms. Harmonic distortion is approx. 0.005% with 32 Ohm load.

• CallPort Spkr / Mic. - self-powered speakers

This selection can also be made in Setup Mode from the User Preferences menu.

The ViewStation's audio signals can be either stereo or mono, have 8 or 16-bit precision and audio sampling rates from 5 to 48 KHz.

2.7.2 HDS ViewStation Video Operations

HDS ViewStation Multimedia Video is a combination of hardware and software options for HDS ViewStation X Window terminals that provide full-motion video within an X Window. These video options are available with HDSware Version 3.1.0.1 and higher.

Additionally, a software-only version of HDS netVideo, called netVideo Playback is available for ViewStation FX, ViewStation Dual and ViewStation Ultra models. This software version can run the HDS netTV Viewer video client.

2.7.2.1 HDS Video Hardware Configuration

HDS ViewStation can be configured with a hardware option called HDS netVideo, which provides both analog and digital video. This video hardware option (actually a different main circuit board using the fast ViewStation Ultra processor with video hardware built in) allows the ViewStation to support connection of an analog video source (camera, VCR, etc.) and to use all the HDS video clients.

2.7.2.2 ViewStation with HDS netVideo Hardware

These ViewStation models have a video circuit board that allows them to accept analog video input (from the HDS ViewCam camera, a VCR, or from the HDS tuner, etc.). These ViewStations can use all the video clients.

HDS netVideo ViewStations can use either a normal analog video input connector (phono jack), or a mini-DIN 4 connector for SVideo input as a factory-installed option. The mini-DIN connector is supplied with an adapter cable for accepting a phono plug as input. Note that the SVideo input can be selected Using the "Input Connector" option in the HDS video client.

Note on monitors:

You can use the HDS ViewStation with netVideo hardware with any HDS-supplied monitor and is set correctly at the factory. If you use another monitor, or change monitors, there is a jumper setting that must be set depending on whether your monitor uses sync on green in its video signal. The sync on green signal affects the range of the video brightness adjustment (typical symptoms of this problem are dim or overly bright displays).

The HDS ViewStation with netVideo hardware has been set at the factory for the monitor on the original order; if you want to use a different monitor, you may have to make this adjustment. The jumper is at position R60, near the video connector (you will have to remove the PC Card board if one is installed). Align the two pins on R60 with the two arrows on the jumper label as needed for your monitor. Look carefully at the label and the arrows, as there is only 0.1" difference between the two selections.

Note on memory:

HDS ViewStation models with the HDS netVideo hardware option contain two SIMM sockets for memory, with no memory on the main circuit board. At least 4 MB of SIMM memory must be installed in socket U3. The U3 socket must be populated first. The second SIMM socket, U2, can have memory of any size. A minimum of 8 MB of memory is required for units equipped with the HDS netVideo hardware option.

2.7.2.3 HDS netVideo (analog and digital video) Specifications

The HDS ViewStation with HDS netVideo hardware accepts analog video inputs in either NTSC, PAL, SECAM composite formats, or SVideo inputs (also called SVHS signals; this video input uses separate luminance and chrominance signals). The SVideo signals use a mini-DIN 4 pin connector on a modified video board. The ViewStation accepts input from a wide variety of devices using these formats, including the HDS Tuner, HDS ViewCam, standard camcorders, VCRs, and laser disk video players.

2.7.2.4 HDS netVideo Playback (digital video) Specifications

HDS ViewStation FX, Dual and Ultra models can be ordered with the HDS netTV Playback software option. This allows these models to receive multicast audio and video streams from the HDS netTV Simulcast client.

The HDS netTV Playback option requires video software only. It allows the use of the HDS netTV viewer.

2.7.3 HDSware netVideo Software

HDSware with video software is sold separately from the normal HDSware server code. It is installed after the normal HDSware server is installed. You must have HDSware server version 3.1 or higher in order to load HDS netVideo software.

If you have the earlier HDSware 3.1 installed, you can install the HDS netVideo software, but you will have to edit two lines in the clients.config file. If you have HDSware 3.1.0.1, this file has been modified for you. HDSware 3.1.0.1 includes an installation script for the video software installation.

2.7.3.1 Installing HDSware netVideo Software

Installing HDSware 3.1.0.1 netVideo software

If you have HDSware 3.1.0.1 or higher, you can use the installation script to install the video software.

If your site has some video ViewStations and some regular ViewStations, you can install the HDSware 3.1.0.1 with the video options and use that software for all units. The HDS netVideo clients will appear on the HDSperx menu of all ViewStations, but the video clients won't start on non-video ViewStations.

If you have customized your configuration files (config.fx and clients.config), you should copy them to another directory and then restore the customized lines to the new configuration files after your installation is complete.

• Load the HDSware tape on your system (use the *HDSware Installation Guide* for sample tape commands) and get the installation script. For example:

tar xvf <*device name*> HDSvidinstall

This finds the installation script; it is the first file on the tape. The tar command continues to search through the entire tape, which may take a few minutes. Once the screen reports that the HDSvidinstall is loaded, you can discontinue the search by typing *Ctrl-C*.

• Run the installation script by typing:

HDSvidinstall

• HDSvidinstall requests two things:

the path of your HDSware directory

the device name of your tape drive

• HDSvidinstall will check for the presence of the *./hds-fx* directory to determine if your HDSware path is correctly specified. It will also check to see that the files in the directory */HDSware/hds-fx/hdsvideo* are writable, and exits if they are not. Verify that the write permissions are correct.

The installation proceeds and notifies you when complete. You need to reboot your ViewStation in order to use the video software.

Installing HDSware Video Software with Version 3.1

You must have HDSware 3.1 server software installed in order to use the HDSware video software.

If you have customized your configuration files (config.fx and clients.config), you should copy them to another directory and then restore the customized lines to the new configuration files after your installation is complete.

• Load the HDSware video software tape on your system. You should change your directory to the HDSware directory - /path/hds-fx and load the tape into that directory. For example:

cd /<*path*>/hds-fx

tar xvf <*device name*>

The tar command should load the /hdsvideo directory into the hds-fx tree. You can delete the HDSvidinstall script if you wish.

• Edit the /hds-fx/config/clients.config file with a text editor (make sure you have the file permission to do this).

Remove the comment character from the hds.server.Extensions line for the MIT-SHM extension. This is available with HDSware 3.1.

hds.serverExtensions: /FXUtilities/hds-fx/extensions/MIT-SHM

Add this line at the end of the file:

#include "../hdsvideo/config/hdsvideo.clients"

• Save the clients.config file.

You need to reboot your ViewStation in order to use the video software. The clients.config file points to the new HDSware video clients and they will appear on your HDSperx menu in your window manager's root menu.

2.7.4 HDS netVideo Clients

The HDS ViewStation netVideo clients run locally on the HDS ViewStation. They are distributed on a separate HDSware release, such as Version 3.1.0.1. This HDSware has a separate installation script and loads the video clients into the normal HDSware software directory structure. Appropriate configuration files and client paths are configured automatically when the HDSware netVideo software is loaded.

HDSperx netVideo Clients

The HDS netVideo clients are these:

HDS Video - for display of analog video on the ViewStation

HDS Conference - for two ViewStations connecting over a network

HDS Tuner - for displaying live TV/cable signals on the ViewStation

HDS Movie - for playing a digital video file on the ViewStation

HDS netTV Simulcast - for broadcasting video from a ViewStation

HDS netTV Viewer - for viewing video broadcasts from the network

These clients are added automatically to your HDSperx menu. The needed paths and configurations, etc. are updated automatically in the clients.config file.

2.7.4.1 Using the HDS Video Client

The HDS Video client displays analog video on the ViewStation from any ordinary video input, including cameras, TV tuners, VCRs, etc.

This client displays high-quality live (30 frames / sec) video from an analog source. It can display up to 4 simultaneous live video windows. The windows can be resized, even to full-screen size, though their aspect ratio is constant. They can be handled as normal X windows by your window manager. There are options to freeze or capture the image, as well as integrated audio capabilities.

Connect either the camera or the tuner (or other analog video input) to the ViewStation's video input. This connection varies with the input device, but is usually a very simple matter. There are no settings or jumpers to set. The NTSC and PAL signal formats are detected automatically; SECAM formats must be selected on the Options menu. For optimum performance with the PAL format, use the Options menu to select "PAL".

Connect the HDS ViewCam Camera

Any common video camera, VCR, camcorder, etc. can be connected to the ViewStation. The phono plug video connector accepts any NTSC (or PAL or SECAM) input.

The HDS ViewCam comes with an AC transformer with cable, and a connecting cable. It is auto-focusing. There is a power switch on the side and buttons for adjusting the contrast.

• Plug in the AC transformer connector to the socket on the back of the camera.

• Plug the HDS ViewCam's video connector into the back of the camera and then plug the video connector (phono jack with yellow cover) into the ViewStation's video jack located on the back of the ViewStation base.

• Connect the HDS ViewCam's two audio phono jack connectors to 3.5 mm stereo adapter (supplied), and then plug that into the ViewStation's input audio jack.

Set Up the Audio Connections

Audio signals accompanying the video, whether from the microphones built into the camera, or from a VCR, or the from the HDS tuner, etc. are controlled by the local HDS audio client whose controls are integrated into the video clients window. The audio input and output jacks on the ViewStation's back panel should be connected to the microphones / speakers, etc. as needed (the HDS ViewCam has microphones built into the camera head). Make the appropriate selections for internal / external speakers, microphone type, etc. from the Audio Device menus in Setup Mode (User Preferences menu).

Refer to Section 2.7.1 for details of operation of the HDS Audio client.

Start an HDS Video Window

Start the HDS video client by pulling down the HDSperx menu with the right mouse button and clicking on the "HDS video" line. The video control menu appears.



The "File" menu offers a "Quit" option to close the HDS video client.

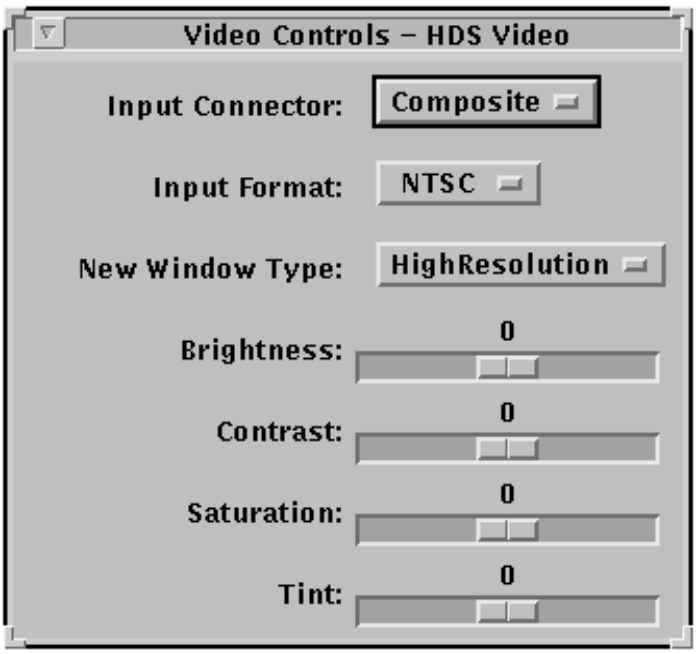
The "Options" menu offers submenus for Video and Audio controls, which are described in detail below.

The "Help" menu offers a brief description of the HDS video client.

There is a button to open a new video window (with settings as specified). There are also slider bars for "Play Volume:", which is the audio signal level sent to the speakers. The "Input Volume:" slider bar is for the audio gain of the input signal; this should normally be left at 1 (one).

The "Audio:" bar is a signal strength meter that continuously monitors the audio level, with green levels indicating good audio and red levels indicating too much volume.

The Video Controls Menu



This menu offers a number of selections:

• Input Connector: Composite or SVideo. Note that the SVideo selection also requires a different connector on the ViewStation's back panel.

• Input Format: NTSC or PAL or SECAM.

• New Window Type: HighResolution or LowResolution. This specifies the resolution used by the "Start New Window" button. The "HighResolution" starts an interlaced video window (the most frequent choice), and "LowResolution" starts a non-interlaced window. Two High Resolution windows can be displayed simultaneously; four Low Resolution windows can be displayed simultaneously.

•There are slider bars for Brightness, Contrast, Saturation, and Tint.

The Audio Controls Menu



The Audio Controls Menu offers a number of selections:

• Input Level: Select the appropriate field either in this menu, or in the Audio Device Setting menu in Setup Mode (refer to Section 2.7.1.4 of this manual for more details).

• Audio Type: Mono or Stereo. This selects either mono or stereo audio.

• Output Type: Select the appropriate field either in this menu, or in the Audio Device Setting menu in Setup Mode (refer to Section 2.7.1.4 of this manual for more details).

• There are also slider bars for "Play Volume", which is the audio signal level sent to the speakers. The "Input Volume" slider bar is for the audio gain of the input signal; this should normally be left at one.

The HDS Video Window

The HDS Video window created when you click "Start New Window" is a full-motion display (30 frames / sec) of the analog input.

There are a number of controls on the video window frame, including:

• Freeze: This temporarily stops the display. This is a toggle button.

• Mute: This temporarily stops the audio output. This is a toggle button.

• Snapshot: This makes a snapshot copy of the video image and displays it in a separate window on the screen. This image can then be copied or manipulated by normal X graphics processing controls (an example) follows.

• Done: This closes the video window.

• Play Volume: There is a slider bar for audio level control.

HDS Video Window



Capturing an Image from the HDS Video Window

It is simple to capture an image from the HDS video window. All you have to do is click on the "Snapshot" button. This creates a full-color window of the video image as a separate window.

The snapshot window may have its own colormap, so it might not display correct colors unless the pointer focus is in the snapshot window. This sometimes occurs because the snapshot window allocates 128 colors. and creates a private colormap if 128 colors aren't available on the server. If the snapshot has a private colormap, the screen will show colormap flashing as you move the cursor in and out of the window. The final image will have a full complement of correct colors.

You can import the snapshot window into documents directly, such as Frame-maker that have a capture option. This typically uses an .xwd (X Window dump) graphics format. You can also use other X graphics programs, such as Xwd, xv , or ImageMagick, to write your snapshot to a file. These programs also allow you to crop or resize, or convert the format of the snapshot image.

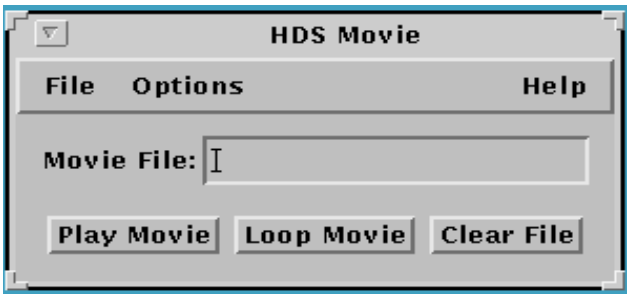
More detailed information is available in the HDS Video manual page in Section 2.7.6.10.

2.7.4.2 Using the HDS Movie Client

The HDS Movie client reads and displays Microsoft AVI format movie files transmitted to the ViewStation. These file are compressed using the Intel INDEO compression algorithm. INDEO versions 2.1, 3.1, and 3.2 are supported. Note that the .avi extension alone does not specify the compression method used in the file.

Microsoft AVI files using different compression schemes are not supported.

The HDS Movie Client



The HDS Movie client offers a number of options:

• The "File" menu offers a "Quit" option.

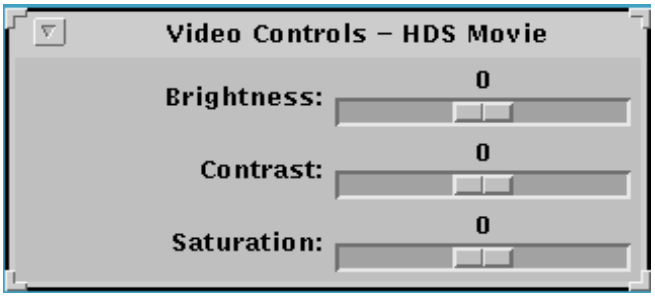
• The "Options" menu offers a "Video Controls" and an "Audio Controls" submenu, which are discussed below.

• The "Help" menu offers a brief description of HDS Movie.

There is line to enter the movie file's path and name. Individual buttons allow you to play the movie file, or loop the movie (continuous replay), or clear the file name.

The file should be loaded with FTP or NFS file transfer protocols (TFTP has a 16 MB limitation); FTP usually gives the best performance. The path should use ViewStation file syntax with a host name specified (e.g. /hostname/path/filename). General discussion of File Transfer Protocols is available in Section 3.1.

The Video Controls - HDS Movie Menu



There are three slider bars to control the HDS Movie window - Brightness, Contrast, and Saturation.

The Audio Controls - HDS Movie Menu



The Audio Controls menu lets you select the Output Type (Speaker or Headphones) and the Play Volume.

The HDS Movie Client



The HDS Movie window displays the movie file. It also offers buttons that let you

- Freeze the movie. This is a toggle button.
- Mute the audio. This is a toggle button.
- Take a Snapshot of the movie.
- Restart the movie. This restarts the movie file at the beginning.
- Done, which stops the movie and closes the movie window.

There is also a slider bar to select the sound level.

More detailed discussion is available in the HDS Movie manual page in Section 2.7.6.6.

2.7.4.3 Using the HDS Conference Client

The HDS Conference client establishes a teleconference between two video ViewStations. The video window from the users desktop is displayed side-by-side with the video window from a remote display; audio signals from each display are also transmitted and played on the ViewStation's speakers.

You start the HDS Conference client from the HDSperx window manager menu. It displays your own video image in the left-hand window. When you try to make a connection, your HDS Conference client tries to connect to the remote ViewStation; it will try to start an HDS Conference session on the remote ViewStation if one is not already running. The remote ViewStation pops up a dialog box reporting the request and asks the remote user to accept the conference session. If it is accepted, the remote ViewStation's video image appears in the right-hand window.

HDS Conference automatically sends video when it starts. If the Audio Device is set to "Callport Spkr / Mic" or "HDS Headset", it automatically sends Audio too. For other audio device selections, you must press and hold the "Hold to Talk" button to send audio.

This is the HDS Video Conference client:



The "File" menu offers a "Quit" option.

The "Options" menu offers three submenus:

Video Controls - to control the video image from the ViewStation's camera

Audio Controls - to control the audio levels of the ViewStation's input

Network Controls - to control the "Quality" and report network activity

There are a number of buttons available to control the HDS Conference session.

• Connect - this button brings up a dialog box asking for the name or IP address of the ViewStation to which you want to conference.

You can also create a list of commonly used hosts (their names must be resolvable by name server) with an appropriate label for each which will be displayed above the window (such as the "Main Conference Room" name above). This list is a simple text file with a name or IP address along with the label. The file is in */FXUtilities/hds-fx/hdsvideo/lib/HDSconf.list*.

The HDSconf.list file looks like this:

The # comment character can be used only at the beginning of a line.

#

Add your own entries below:

123.123.123.120 Main Conference Room

123.123.123.140 Marketing Conference Room

steveafx Steve Ahlbom's Office

mkt26 David Tolman's Office

eng14 Ed Parks' Office

123.123.123.189 Bill Bredbenner's Office

The "Connect" button brings up a chooser box for the hosts listed in the HDSconf.list file, as shown in this example. You can enter the list appropriate for your site and environment and make the connection by clicking on the name you want. The names in this file must be resolvable by your domain name server.



• Disconnect - this button disconnects the HDS Conference connection to the remote host.

• Freeze - this freezes the incoming image. This button is a toggle button.

• Mute - this button mutes the incoming audio. This button is a toggle button.

• Send Video - this button transmits live video automatically. If you toggle this button, the image is not sent; this is useful for pausing the conference while you conduct some off-line conversation.

• Hold to Talk - this button controls whether your audio signal is transmitted to the other ViewStation. When the Audio Device is set to the Callport Spkr/Mic or to the HDS Headset, audio is transmitted automatically. Otherwise, you must select it with the left mouse button and hold it down while you speak. This allows you to control what conversation is sent and also to reduce the network activity by not transmitting any more audio data than necessary.

• Speaker Volume slider bar allows you to control the audio level of the speakers.

• Mic Volume slider bar allows you to adjust the local microphone level. If you adjust the microphone volume, look at the Audio meter to see that the signal ranges remain in the green range. Sustained red audio levels result in a poorer sound quality.

• The "Audio:" bar is a signal strength meter that continuously monitors the audio level, with green levels indicating good audio and red levels indicating too much volume.

The video images are received by each ViewStation as analog signals. They are then converted to digital images, compressed, and sent as packets across the Ethernet connection between the two ViewStations, where the receiving ViewStation uncompresses them and displays them on the screen in the HDS Conference window.

There are several things to observe about this process. The compression done by the ViewStation is variable based on the image itself, and adjustable by the user. The compression technique examines the image and transmits only parts of the image which were changed from the previous frame - so a relatively static image, such as a person talking, has less actual data transmitted than a full-motion image. The picture quality of the received image will vary based on this volume of data, first because of the network constraints, and second, because of the level of compression selected by the user.

The Video Controls - HDS Conference Menu



This menu allows you to select several things about the HDS Conference video on your ViewStation.

• Compression - On or Off. This affects the quality of the transmitted image. You may want to select Off if the image is relatively static and the network is relatively open. This is also variably adjustable with the "Quality" slider on the Network Controls menu.

• Input Connector - Composite or SVideo. You would use this only if you have an SVideo input connector.

• Input Format - NTSC, PAL, or SECAM.

There are slider bars to adjust your local video image:

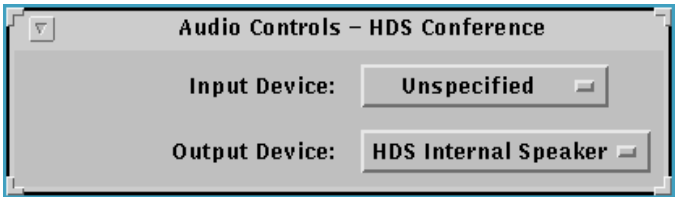
Brightness

Contrast

Saturation

Tint

The Audio Controls - HDS Conference Menu

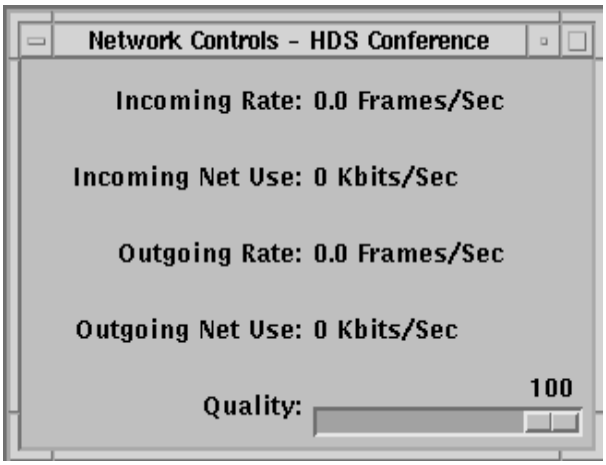


The Audio Controls menu offers two selections:

• Input Device: This lets you select the Audio Input device you have installed. Options are: Unspecified, HDS ViewCam, HDS Headset, CallPort Spkr / Mic, HDS Microphone, HDS Tuner, VCR, and Line Level Audio. This selection can also be made in Setup Mode from the User Preferences menu.

• Output Device: This allows you to specify the Audio Output device you have installed. Options are: HDS Internal Speakers, External Speakers, HDS Headset, and CallPort Spkr / Mic. This selection can also be made in Setup Mode from the User Preferences menu.

The Network Controls - HDS Conference Menu



The Network Controls menu reports the current state of the HDS Conference session, with data on the frame rate and network usage for both the incoming and outgoing video images. These values report actual network usage and reflect the working of the HDS Conference session - how much data is transmitted (based on the content of the image and the compression selected) and the network load.

These values can be adjusted with the "Quality" slider bar. This lets you select the level of compression that the ViewStation uses on the video image.

Quality set at 100% selects the maximum quality of the video image, which means that the ViewStation will send every frame in its entirety, which in turn means

that the frame rate will probably decrease. The consequence will be a high quality image, but less smoothness of motion.

Quality set at 0% selects minimum quality of the video image, which usually means that the ViewStation sends only the parts of each frame that have changed significantly from the previous frame. The result is that the frame rate is high, with good motion but somewhat less quality of the images themselves. Refer to the HDS Conference manual pages for a more detailed discussion.

This Quality setting is a highly subjective choice and will depend on the objectives of your conference session. The actual frame rate is also greatly affected by the motion within the image and other network usage. You will probably want to experiment a little to see the consequences of these selections.

2.7.4.4 Using the HDS Tuner Client

The HDS Tuner allows you to display a live television or cable image on the ViewStation screen. This might be useful for monitoring news feeds or weather information, or closed circuit TV for security monitoring, and so on.

The HDS Tuner consists of two components: a small tuner box, with cables, and a ViewStation local client to control channel selection and tuning.

Connecting the HDS Tuner to the ViewStation

Unpack the HDS Tuner box. There is a separate installation guide for the HDS Tuner. There should be a tuner box, a power transformer and cable, an audio and a video cable, and a special cable to connect to the ViewStation's Serial 1 port.

• Plug in the power from the AC transformer to the tuner (on the side).

• Connect the HDS Tuner's controls to the serial port, "Serial 1", so you can control the tuner from the ViewStation's display. All connectors and adapters are supplied. Connect the other end of the cable to the tuner's special connector.

• Connect the tuner's audio cable to the ViewStation's AUDIO IN connector.

• Connect the tuner's video cable to the ViewStation's VIDEO connector.

• Connect a TV antenna (not supplied) or cable input to the tuner (coaxial connector).

The HDS Tuner Client



Using the HDS Tuner Client

This is the ViewStation client you use to control the TV/cable input to the ViewStation's video. It looks like a hand-held remote control for your television. Start the HDS Tuner client from the window manager menu. Hold down the right mouse button on the root window and select the HDSperx sub-menu. Click on HDS Tuner.

The HDS Tuner client is a graphical means to control the tuner. It allows you to select channels, scan the channels, etc. just like a hand-held remote control for your TV. The ViewStation controls the HDS Tuner through its serial port connection according to your selections on the ViewStation screen.

When you connect the HDS Tuner for the first time, you should scan the available channels (the available channels depend largely on your geographical location and your antenna or cable connection) so the tuner client can find the available channels. You can use the fine tuning on the tuner client to enhance your reception (these settings are retained by the tuner). Refer to the HDS Tuner man page for information on preconfiguring the available channel and fine tuning information.

Open the HDS Video client and bring up a video window, as described in the section above. With the tuner's output connected to the ViewStation's video port, the ViewStation displays the signal selected by the tuner. You can change the channel with the HDS Tuner client on the ViewStation's display.

2.7.4.5 HDS netTV Simulcast Client

The HDS netTV Simulcast client allows you to broadcast a video signal on your network using IP multicasting to other HDS ViewStations equipped with HDS netVideo software.

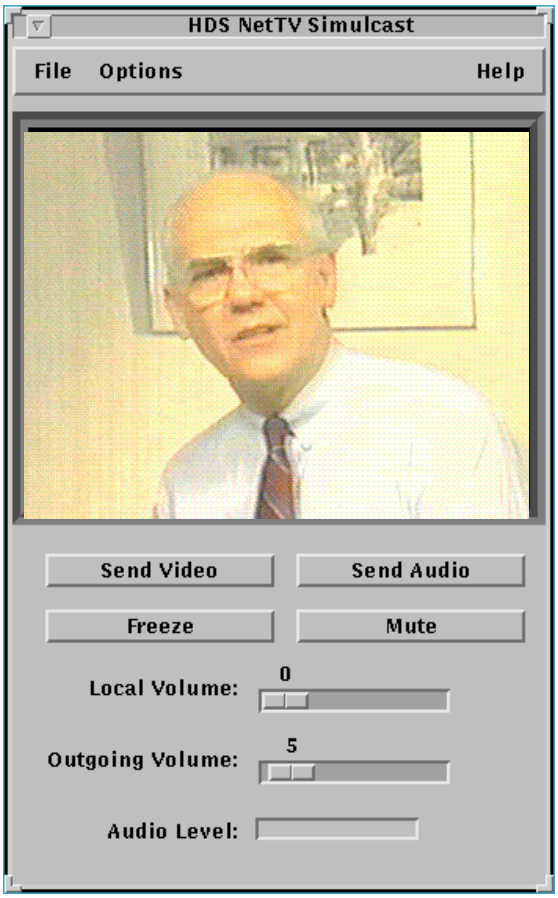
Start the HDS netTV Simulcast client from the window manager's pull-down menu. Hold down the right mouse button on the root window and select the HDSperx sub-menu. Click on HDS netTV Simulcast.

HDS netTV Simulcast displays the current video input in its window, typically from your camera or VCR or from the tuner.

Click on the buttons to "Send Video" and "Send Audio" to broadcast the signals. This sends the audio and video signals to a multicast IP address (the default is 224.2.1.1) where it can be received by any ViewStation on the network. (The values for the multicast address and the video/audio ports are selectable from the "Network Addressing" option of the Simulcast client.)

Note that the Simulcast can be sent in two sizes, 160x120 or 320x240.

HDS netTV Simulcast Client



The HDS netTV Simulcast client offers a number of options:

- The "File" menu offers a "Quit" option.
- The "Options" menu offers four submenus (discussed below)

  Video Controls - to control the local video image

  Audio Controls - to control the local audio signals

  Network Controls - to control the impact of simulcasting on the network

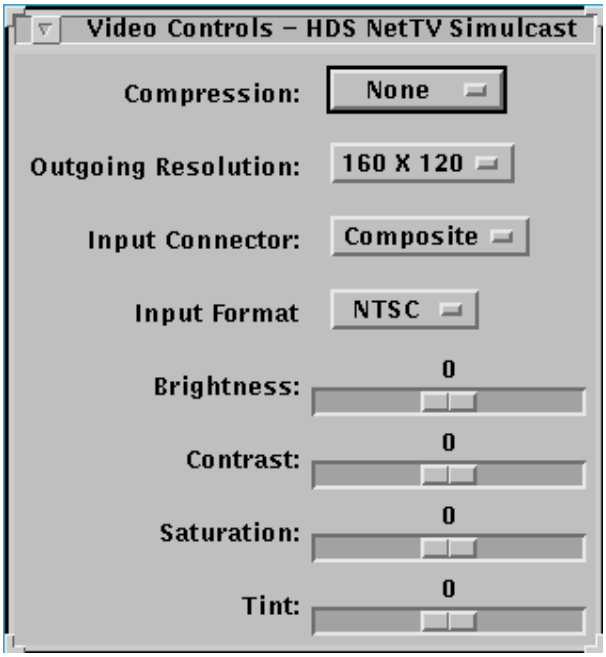
  Network Addresses - to select address and port variables for multicasting

There are also buttons for "Send Video" to broadcast the video or toggle to stop the transmission. You can also "Freeze" the video image. There are also buttons to "Send Audio" to broadcast the audio or toggle to stop the transmission. You can also "Mute" the local audio. All of these buttons are toggle buttons.

There are slider bars for Local Volume and Outgoing Volume.

The "Audio level:" bar is a signal strength meter that continuously monitors the audio level, with green levels indicating good audio and red levels indicating too much volume.

The Video Controls - HDS netTV Simulcast Menu



The Video Controls menu offers a number of selections.

- Compression - This controls the quality of the video image. A slider bar for this variable is available on the "Network Controls" menu. There are three compression selections available: None, CellB, and HDSGrey. None indicates no compression of the images. CellB indicates that the ViewStation uses CellB compression (adjustable with the "Quality" slider on the Network Controls menu). HDSGrey indicates that the image color is discarded in order to improve the transmission frame rate.

• Outgoing Resolution - 160x120 or 320x240. This is the actual pixel size of the transmitted image. Note that the larger image has 4 times as many pixels, with corresponding impact on the network. (Note also that the HDS netTV viewer client has options to enlarge the received image. This is frequently the better choice as it reduces the network traffic.)

• Input Connector - Composite or SVideo. You would use this only if you have an SVideo input connector.

• Input Format - NTSC, PAL, or SECAM.

• There are four slider bars for adjustment of the video image:

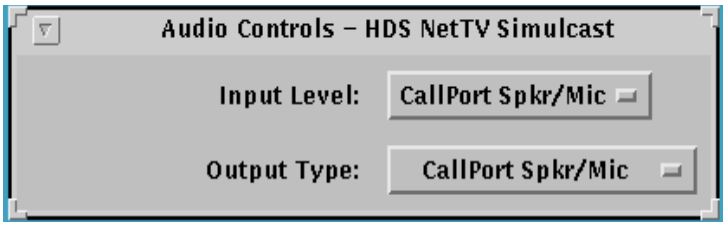
Brightness

Contrast

Saturation

Tint

The Audio Controls - HDS netTV Simulcast Menu

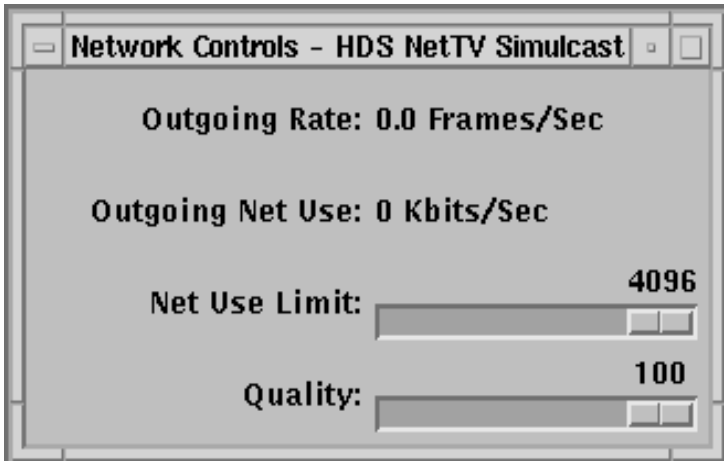


The Audio Controls menu offers two selections:

• Input Level - Unspecified, HDS ViewCam, HDS Headset, CallPort Spkr / Mic, HDS Microphone, HDS Tuner, VCR, and Line Level Audio.

• Output Type - HDS Internal Speaker, External Speakers, HDS Headset, and CallPort Spkr / Mic.

The Network Controls - HDS netTV Simulcast Menu

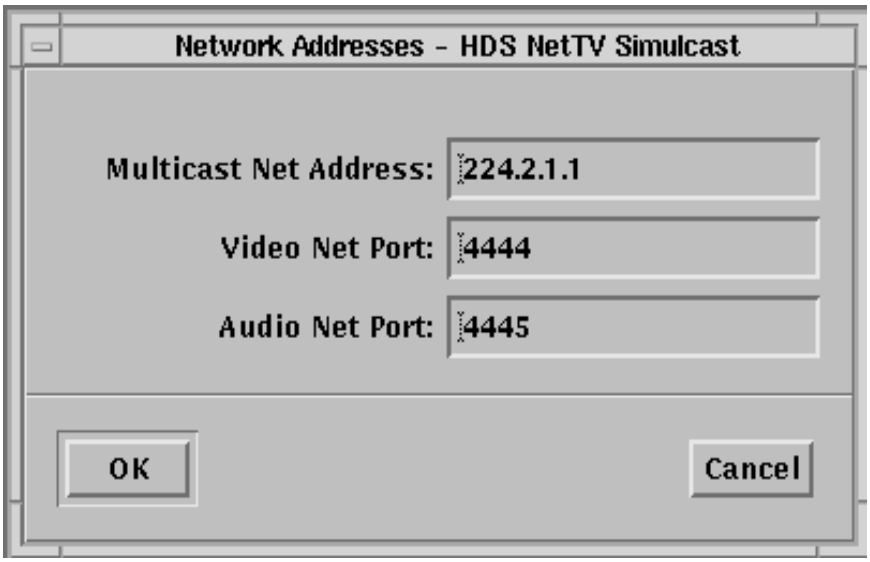


The Network Controls menu reports network usage (frame rate and network load) and allows you to set some variables to control the impact of multicasting on your network.

• Net Use Limit slider bar allows you to set a maximum network usage value. This variable sets an upper boundary for network use by the simulcast. Decreasing this value can result in reduced frame rates. The setting appropriate for your network is dependent on your network environment.

• Quality slider bar selects the amount of compression done to the video images by the ViewStation. Quality of 100% is maximum video quality (and probably a reduced frame rate). Quality of 0% means maximum compression, so you will have a relatively high frame rate but somewhat less video image quality.

The Network Addresses - HDS netTV Simulcast Menu



The Network Addresses menu allows you to set several network variables.

• Multicast Net Address - the default is 224.2.1.1; other multicast addresses are permitted with the IP protocol. You can also enter an address for a specific ViewStation if you want to restrict the viewers to a single ViewStation.

• Video Net Port - the default port is 4444; you can select another port number.

• Audio Net Port - the default is 4445; you can select another port.

These addresses and ports allow selection of video signals like TV channels, the sender and receiver must have the same settings to receive the signal.

2.7.4.6 HDS netTV Viewer Client

This is the client to receive multicast video signals sent by the Simulcast client.

The netTV Viewer client can be used on any ViewStation FX, Dual or Ultra equipped with HDS netVideo Playback software. It need not be a ViewStation with the video hardware installed but must be running HDSware video software.

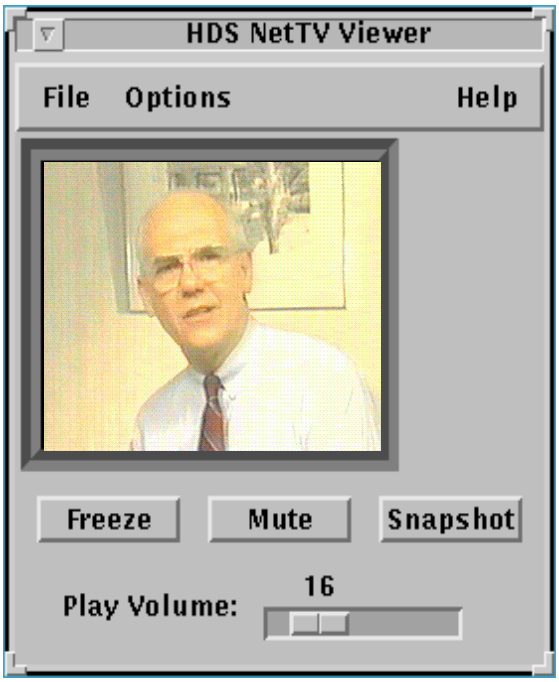
To hear the broadcast audio, the ViewStation must have the Audio option installed.

Start the HDS netTV viewer client from the window manager's pull-down menu. Hold down the right mouse button on the root window and select the HDSperx sub-menu. Click on HDS netTV Viewer.

The HDS netTV Viewer client opens a connection to the multicast IP address and will display the HDSnetTV signal being broadcast.

Running the HDS netTV Viewer on the same ViewStation which is broadcasting via the HDS netTV Simulcast client is strongly discouraged, as this will greatly diminish the broadcast frame rate.

HDS netTV Viewer Client



The HDS netTV Viewer offers several options:

- The "File" menu offers a "Quit" option.
- The "Options" menu offers four submenus (discussed below)

  Video Controls - to control the local video image

  Audio Controls - to control the local audio signals

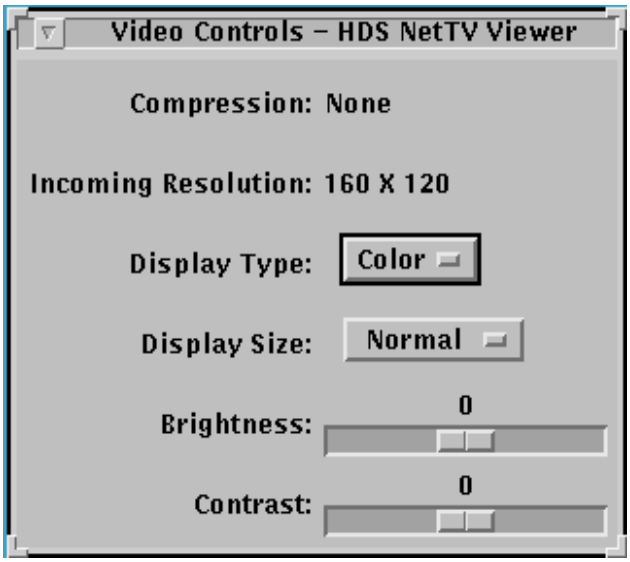
  Network Controls - to control the impact of simulcasting on the network

  Network Addresses - to select address and port variables for multicasting

There are also buttons for "Freeze" to stop the video transmission temporarily. There are also buttons to "Mute" the audio signal. These are both toggle buttons.

There is also a button to take a "Snapshot" of the video image. There is a slider bar for Play Volume.

The Video Controls - HDS NetTV Viewer



The Video Controls menu offers a number of selections.

• Compression - This reports the compression of the video image as sent by the HDS netTV Simulcast client. There are three compression selections available: None, CellB, and HDSGrey. None indicates no compression of the images. CellB indicates that the ViewStation uses CellB compression (adjustable with the "Quality" slider on the Network Controls menu). HDSGrey indicates that the image color is discarded in order to improve the transmission frame rate.

• Incoming Resolution - 160x120 or 320x240. This reports the pixel size of the received image.

• Display Type - Color or Grey. The Grey setting uses fewer colors in the colormap and may display at a higher frame rate.

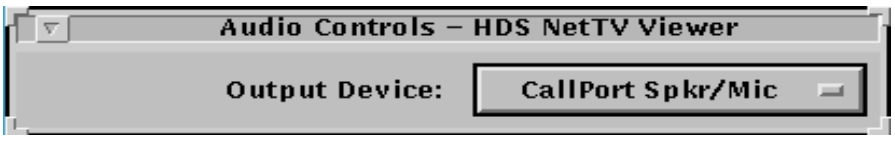
• Display Size - Normal, Reduced, or Enlarged. This lets you expand the received image on your local display. Note that the number of pixels transmitted remains the same, but the displayed image size is changed.

• There are two slider bars for adjustment of the video image:

Brightness

Contrast

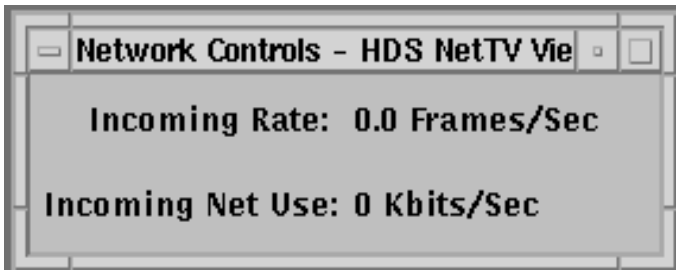
The Audio Controls - HDS netTV Viewer Menu



The Audio Controls menu offers one selection:

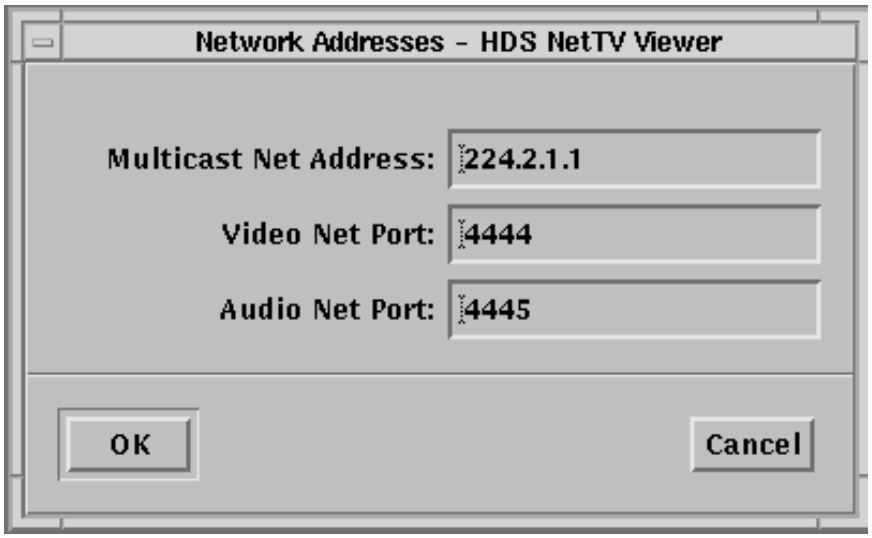
• Output Device - HDS Internal Speaker, External Speakers, HDS Headset, or Callport Spkr / Mic. This lets you select the type of audio output device you are using.

The Network Controls - HDS netTV Viewer Menu



The Network Controls menu reports network usage: Incoming frame rate and Incoming network use. These are adjustable on the multicasting ViewStation.

The Network Addresses - HDS netTV Viewer Menu



The Network Addresses menu allows you to set several network variables. All of these variables should be set the same for both the Simulcast and Viewer clients on all ViewStations using these options.

• Multicast Net Address - the default is 224.2.1.1; other multicast addresses are permitted with the IP protocol.

• Video Net Port - the default port is 4444; you can select another port number.

• Audio Net Port - the default is 4445; you can select another port.

These addresses and ports allow selection of video signals like TV channels, the sender and receiver must have the same settings to receive the signal.

2.7.5 Frequently Asked Questions:

These are questions often asked about HDS ViewStation video operations.

2.7.5.1 How does Video affect my network?

The HDS netTV Simulcast client does several things:

- Receives video (and audio) from its analog source
- Converts the analog data (video and audio) to digital format
- Compresses the video data using Sun's CELL-B compression algorithm (Alternately, HDSGrey discards the color data of the image to reduce traffic)
- Transmits the compressed data over the Ethernet using IP Multicasting

The HDS netTV Simulcast client has two slider bars that let you adjust the impact of the multicasting load on the network. These two slider bars are "Quality" and "Net Use Limit" on the Network Controls menu. They control the amount of compression used by the Simulcast client.

The HDS netTV viewer client reports the frame rate it is receiving and the traffic generated on the network. These are affected both by the compression setting and also by the content of the video image (CELL-B compression doesn't send the entire video image with each frame, but only the pixels changed since the last frame. Therefore, an image of rapid movement has much more data to be compressed than an image of a "talking head" against a static background).

The examples shown below illustrate several possibilities. In these cases, the "Quality" slider was set to 0%, 50%, and 100% with the results as recorded. The "Net Use Limit" was not restricted in these tests. Compression was turned On.

Case 1: Maximum Compression (maximum frame rate)

Capture size: 160x120 pixels

Quality: 0

Frame Rate: 34 frames per second

Bits per second: 200,000 average

% of Ethernet: 200,000 / 5,000,000 = 4%

% of switched Ethernet: 200,000 / 10,000,000 = 2%

Case 2: Average Compression

Capture size: 160x120 pixels

Quality: 50

Frame Rate: 30 frames per second

Bits per second: 400,000 average

% of Ethernet: 400,000 / 5,000,000 = 8%

% of switched Ethernet: 400,000 / 10,000,000 = 4%

Case 3: Maximum Quality (minimum compression)

Capture size: 160x120 pixels

Quality: 100

Frame Rate: 32.5 frames per second

Bits per second: 1,200,000 average

% of Ethernet: 1,200,000 / 5,000,000 = 24%

% of switched Ethernet: 1.200,000 / 10,000,000 = 12%

2.7.5.2 How large are digital video files in different formats?

There are two principal variables to consider: first, the resolution size of the image (either 160x120 or 320x240 pixels); second, the audio component of the files.

We've examined some .avi files which use Indeo 3.1 compression. The audio tracks in the files were recorded in mono at 11 KB / sec. The 160x120 image files occupy approximately 3 MB of file space per minute of video (the audio portion of the file represents about 22% of the file size). The 320x240 image files occupy approximately 9 MB of file space per minute of video (the audio portion of the file represents about 7% of the file size).

2.7.5.3 What do I need for my in-house video training?

A typical training application in a university setting uses HDS ViewStations with video features. A laboratory classroom uses a ViewStation with netVideo and a video camera and microphone. This ViewStation captures the professor with his lectures and exhibits, then uses the netTV Simulcast client to send this video and audio to other ViewStations on the network, in this case in several different classrooms across the campus.

The equipment needed for this simple installation is one ViewStation with HDS netVideo (along with appropriate cameras, microphones, etc.), and as many other ViewStation FX models with HDS netTV Viewer software and audio capability as needed.

Other training techniques include preparation of instructional video tapes which are broadcast at scheduled times on an HDS netVideo unit to other ViewStations on the network. This is an easy way to create an internal broadcast network for your company.

2.7.5.4 What are some other applications for netVideo ViewStations?

Two of HDS's customers use their ViewStations for emergency response services, in one case for a municipal emergency response center, and in another case for a disaster control center for a regional communications company.

In the emergency response center, the netVideo ViewStation receives TV signals from a remote camera at the emergency site via a private cable channel broadcast. This is then multicast to other ViewStations (along with news and weather channel information). Since the ViewStations are full-featured X terminals, the operators have access to multiple host computers for database information and dispatch to support their on-site teams.

The disaster control center is a centralized regional group that relies on netVideo ViewStations to monitor problems via TV news and weather channels, with internal distribution at the center via the ViewStation's multicasting client. They also use extensive teleconferencing to local offices with the ViewStation, which is especially efficient because of their high-speed networks and routing.

2.7.6 HDS Multimedia Client Manual Pages

2.7.6.1 HDS Audio Local Client Manual Page

NAME

hdsaudio - HDS ViewStation Audio Control Panel

SYNOPSIS

hdsaudio [-d dev] [-Xt / Motif options]

DESCRIPTION

The hdsaudio downloadable Motif client runs locally on the HDS ViewStation. It provides a control panel which allows the user to easily set and monitor various audio settings.

These settings include:

Playback Volume

Playback Stereo Balance

Record Gain

Record Stereo Balance

Input Monitoring Volume

Audio Out Device Type

Audio In Device Type

The client also displays useful statistics about current audio activity (sample counts, sampling rates, etc).

Changes made with hdsaudio stay effective even after the client is exited. If a -display argument is used, the controls can appear on a remote X Window display but the audio device affected is the one on the ViewStation executing hdsaudio. This client can be used with or without the HDS NetAudio server, auhds.

This client can only be used on ViewStations equipped with the optional HDS MultiMedia Stereo Sound Card.

OPTIONS

The hdsaudio client accepts all of the standard X Toolkit command line options as well as the following:

-d dev

Specify an alternate audio device to which output should be directed. If the -d option is not specified, / dev / audio is the default audio device.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose HDS Audio. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

HDSAudio hdsaudio

XmForm BaseFrame

XmFrame

XmRowColumn GainPanel

XmLabel title

XmSeparator sep

XmForm slider

XmLabel play

XmScale scale

XmForm slider

XmLabel playbalance

XmScale scale

XmForm slider

XmLabel record

XmScale scale

XmForm slider

XmLabel recordbalance

XmScale scale

XmForm slider

XmLabel monitor

XmScale scale

XmForm pause

XmPushButton pause

XmPushButton pause

XmCascadeButtonGadget speaker

XmCascadeButtonGadget source

XmForm spform

XmMenuShell

XmRowColumn spkr_menu

XmPushButton internal

XmPushButton external

XmForm sform
XmMenuShell
XmRowColumn src_menu
XmPushButton line
XmPushButton min
XmPushButton dyn
XmPushButton boost
XmForm StatusPanel
XmFrame
XmRowColumn play
XmLabel title
XmSeparator sep
XmForm open
XmToggleButton toggle
XmLabel label
XmForm paused
XmToggleButton toggle
XmLabel label
XmSeparator sep
XmForm active
XmToggleButton toggle
XmLabel label
XmForm error
XmToggleButton toggle
XmLabel label
XmForm waiting
XmToggleButton toggle
XmLabel label
XmSeparator sep
XmForm eof
XmLabel label
XmLabel val
XmForm sam
XmLabel label
XmLabel val

XmSeparator sep
XmForm rate
XmLabel label
XmLabel val
XmForm chan
XmLabel label
XmLabel val
XmForm prec
XmLabel label
XmLabel val
XmForm enc
XmLabel label
XmLabel val
XmFrame
XmRowColumn record
XmLabel title
XmSeparator sep
XmForm open
XmToggleButton toggle
XmLabel label
XmForm paused
XmToggleButton toggle
XmLabel label
XmSeparator sep
XmForm active
XmToggleButton toggle
XmLabel label
XmForm error
XmToggleButton toggle
XmLabel label
XmForm waiting
XmToggleButton toggle
XmLabel label
XmSeparator sep
XmLabel

XmForm sam

XmLabel label

XmLabel val

XmSeparator sep

XmForm rate

XmLabel label

XmLabel val

XmForm chan

XmLabel label

XmLabel val

XmForm prec

XmLabel label

XmLabel val

XmForm enc

XmLabel label

XmLabel val

XmDialogShell audio_popup

XmMessageBox audio

SEE ALSO

auhds(1) hdsplay(1) netaudio(1)

COPYRIGHT

Copyright 1995, HDS Network Systems, Inc.

2.7.6.2 HDS Play Audio Client Manual Page

NAME

hdsplay - play audio file on HDS ViewStation

SYNOPSIS

hdsplay [-V] [-v #] [-m #] [-d dev] [-b #] [-s #] [-c #] [-e ULAW | ALAW | LINEAR] [-p #] [-r] [file...]

DESCRIPTION

hdsplay will play audio files on a Human Designed Sytems ViewStation. The files can be in standard Sun / Next format or can be raw binary digital audio files. For raw data files, the data format and sampling rate can be specified as command line options. If no filenames are specified, the standard input is used.

The hdsplay program is run locally by the ViewStation. Any input files specified as arguments should be ViewStation paths using the ViewStation's Remote Hosts Setup configuration. See the EXAMPLES section below.

Note that TFTP is generally not a fast enough protocol to transfer sound files with a high sample rate. Instead, use NFS or FTP for files with high sampling rates.

The hdsplay program can only be used on ViewStations equipped with the optional HDS MultiMedia Audio Card.

OPTIONS

-V

Verbose. Print informational messages to the standard error.

-v vol

Set the output volume to vol before playing begins. vol is an integer value between 0 and 100, inclusive. If this argument is not specified, the output volume remains at the level most recently set by any process.

-m vol

Set the monitor volume to vol before playing begins. vol is an integer value between 0 and 100, inclusive. Changes to the monitor volume will persist after the 'hdsplay' program has exited.

-d dev

Specify an alternate audio device to which output should be directed. If the -d option is not specified, /dev/audio is the default audio device.

-b size

Specify the blocksize for file reads and device writes, in kilobytes. The default value is 64.

-s rate

Specify the date sampling rate, in samples per second.

The rate is usually determined from the input file header, so this option is only needed for raw input data. The default value is 8000. Other acceptable values are: 5512 6615 9600 11025 16000 18900 22050 27428 32000 33075 37800 44100 and 48000.

-e encoding

Specify the data sample encoding type. The encoding is usually determined from the input file header, so this option is only needed for raw input data. The default value is ULAW. Other acceptable values are ALAW and LINEAR.

-c channels

Specify the number of interleaved channels. The number of channels is usually determined from the input file header, so this option is only needed for raw input data. Acceptable values are 1 for mono and 2 for stereo. The default value is 1.

-p precision

Specify the sample precision for each channel, measured in bits. The precision is usually determined from the input file header, so this option is only needed for raw input data. Acceptable values are 8 and 16 bits. The default is 8 bits.

A 16 bit precision can only be used for LINEAR encodings and 16 bit data must be in most-significant-byte-first order.

-r

Treat files as raw input data. No attempt will be made to read a file header.

-?

Help. Print a command line usage message.

EXAMPLES

To play a sound file from a host computer, the following commands can be issued in the host's shell:

example% rsh fxhost hdsplay - < file.au

example% rsh fxhost hdsplay -r -s 44100 -p 16 -e linear -c 2 - < cdquality.raw

example% rsh fxhost hdsplay /myhost/sounds/demo.au

From a ViewStation 'HDS Console' shell:

hds% hdsplay /mysun/usr/demo/SOUND/sounds/spacemusic.au

SEE ALSO

hdsaudio(1)

2.7.6.3 HDS Au Audio Client Manual Page

NAME

auhds - HDS NetAudio server

SYNOPSIS

auhds [:listen port offset] [-option ...]

DESCRIPTION

auhds is the name of the NetAudio server running as a down- loaded client on the HDS ViewStation. The auhds NetAudio server provides ViewStation audio services to applications on the network.

See netaudio(1) for more information.

STARTING THE SERVER

If frequent use is expected, auhds should be started in the background from the ViewStation's Startup Command File. If only occasional use is expected, auhds can be started when needed via the HDS Console or by using the Remote Shell protocol.

When the NetAudio server starts up, it takes over the ViewStation's local /dev/audio. Note that other local applications that attempt to access /dev/audio (such as hdsplay) will fail while it is running.

NOTES

The HDS ViewStation local client hdsaudio can be used in conjunction with auhds to control output speaker selection, volume, etc.

The auhds server can only be used on ViewStations equipped with the optional HDS MultiMedia Audio Card.

NETWORK CONNECTIONS

The NetAudio server supports connections made using the following reliable byte-streams:

TCP/IP

The server listens on port 8000+n, where n is the listen port offset.

Unix Domain

The NetAudio server uses /dev/netaudion as the filename for the socket, where n is the display number.

OPTIONS

The auhds server accepts the following command line options:

-aa

Allows clients on any host to connect. By default, access is allowed only to clients on the local host. This option should almost always be specified.

SIGNALS

The NetAudio server attaches special meaning to the following signals:

SIGHUP

This signal causes the server to close all existing connections, free all resources, and restore all defaults.

SIGTERM

This signal causes the server to exit cleanly.

SIGUSR1

This signal is used quite differently from either of the above. When the server starts, it checks to see if it has inherited SIGUSR1 as SIG_IGN instead of the usual SIG_DFL. In this case, the server sends a SIGUSR1 to its parent process after it has set up the various connection schemes.

DIAGNOSTICS

Too numerous to list them all. If started in the background, messages are usually logged through the ViewStation's syslog facility.

HDS VIEWSTATION FILES

/dev/netaudio* Unix domain socket

/dev/audio Audio device

SEE ALSO

hdsaudio(1), netaudio(1), auinfo(1), auplay(1), auctl(1)

COPYRIGHT

Copyright 1993, Network Computing Devices, Inc. Copyright 1994, Human Designed Systems, Inc.

AUTHORS

The HDS version of the NetAudio server was based on the publically-available NetAudio server source originally written by Greg Renda and Dave Lemke, which has large amounts of code borrowed from the sample X server.

The sample X server was originally written by Susan Angebranndt, Raymond Drewry, Philip Karlton, and Todd Newman, from Digital Equipment Corporation, with support from a large cast. It has since been extensively rewritten by Keith Packard and Bob Scheifler, from MIT.

2.7.6.4 NetAudio Audio Client Manual Page

This is the man page for NetAudio. More extensive man pages are available in the ./hds-fx/contrib/Netaudio directory.

NAME

NetAudio - a portable, network-transparent audio system

SYNOPSIS

The NetAudio service is a network-transparent system for playing, recording, and manipulating audio data over a network. It uses the client/server model to separate application code from the software drivers needed to control specific audio input and output devices.

DESCRIPTION

The NetAudio service provides applications with the ability to send and receive audio data such as voice, sound effects, and music in a network environment. A audio server inside the desktop terminal or personal computer controls the various input and output devices in response to messages sent from client programs running on other hosts.

A variety of utility programs are provided that can be used to play or record audio using shell scripts or command-line procedures. More ambitious applications can communicate directly with the audio server using a C-language programming library.

The NetAudio service supports a variety of the common for-mats used to store sound data. Audio inputs and outputs can run at a any of a range of sampling rates. The audio server automatically converts all data to the designed format or rate.

Streams of data from multiple inputs can be mixed together and directed to any attached output device. Sounds that are used many times can be stored in the server so that they do not need to be transmitted over the network each time they are played. Furthermore, inputs can be hooked directly to outputs (for example, a CD player can be connected to a set of speakers) so that data can be played without requiring any client intervention or network traffic.

AUDIO SERVER NAMES

Each audio server has a name that is used by applications in establishing a connection. The format of the name depends upon the type of network being used. In TCP/IP environments, audio server names look like:

tcp/hostname:portnum

where hostname is the name or numeric IP address of the desktop machine to be contacted and portnum is the port number on which the server is listening. If the "tcp/" prefix is not given, 8000 is automatically added to portnum so that the audio server name is compatible with the X Window System display name.

In DECnet environments, audio server names look like:

decnet/nodename::num

where nodename is the name or numeric DECnet address of the desktop machine to be contacted and num is the DECnet task AUDIO$num. If the "decnet/" prefix is omitted, it is automatically added so that the audio server name is compatible with the X Window System display name.

Most audio utilities accept a command line argument -audio servername for specifying the audio server to be contacted.

% auinfo -audio tcp/unit:8000

If no server is specified on the command line, the AUDIOSERVER environment variable is checked to see if a name has been stored there. Otherwise, the DISPLAY environment variable used by X Window System applications is checked and the corresponding audio server is used.

SAMPLE PROGRAMS

Several sample programs that use the NetAudio service are provided:

auplay

This program simply plays the files listed on its command line. It is typically used from shell scripts, command-line procedures, or other programs that just need to "dump" sound data to the audio server.

audemo

This program allows the user to select and play pre-recorded sound files or to record new sound files. It provides a simple control panel for dynamically changing the volume while a sound is being played and for aborting a sound.

audial

This program generates North American telephone touch tones correspond to a given string. audial can also recognize touch tones and produce the corresponding string.

autool

This program is compatible with the Sun audiotool program and is typically used to play prerecorded sound files. Unlike auplay it provides a graphical interface to allow the user to replay or cancel the data.

auinfo

This program lists information describing the capabilities of an audio server.

auctl

This program can be used to change various device parameters.

auscope

This program is an audio protocol filter that can be used to view the network packets being sent between an application and an audio server.

aupanel

This program provides an X-based window-oriented interface allowing the user to adjust the attributes of the devices provided by the NetAudio service.

auedit

This program provides an X-based window-oriented interface allowing the user to edit audio files.

auconvert

This program allows the user to perform various conversion operations on sound files.

AUDIO FILES

The audio programming library and utility programs can be used to read and write files in any of the following file formats:

Sun/NeXT

These files typically have a '.au' or '.snd' extension. They are usually produced by Sun or NeXT workstations.

Creative Labs

These files typically have a '.voc' extension. They are usually produced by by Soundblaster audio cards on IBM PC compatible computers.

Microsoft WAVE

These files typically have a '.wav' extension. They are usually produced by IBM PC compatible computers.

AIFF

These files typically have a '.aiff' or '.aif' extension. They are usually produced by SGI workstations or Apple computers.

Due to the nature of the file formats, only the Sun / NeXT file format can be piped through standard input and / or standard output. The convention for specifying standard input or standard output is to use a hyphen ("-") as the file name.

PROGRAMMING

Application developers may create their own audio clients using the NetAudio programming library.

ENVIRONMENT VARIABLES

The following environment variables are used by NetAudio programs:

AUDIOSERVER

This specifies the name of default audio server to contact if no name is explicitly given on the command line.

DISPLAY

This specifies the name of an X Window System display that should be assumed to also have a corresponding NetAudio server.

SEE ALSO

auctl(1), audemo(1), audial(1), auinfo(1), auplay(1), auscope(1), autool(1)

COPYRIGHT

The following copyright and permission notice applies to all files in the NCDaudio application programming libraries and example programs provided in source form by Network Computing Devices:

Copyright 1993 Network Computing Devices, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name Network Computing Devices, Inc. not be used in advertising or publicity pertaining to distribution of this software without specific, written prior permission.

THIS SOFTWARE IS PROVIDED 'AS-IS'. NETWORK COMPUTING DEVICES, INC., DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING WITHOUT LIMITATION ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT SHALL NETWORK COMPUTING DEVICES, INC., BE LIABLE FOR ANY DAMAGES WHATSOEVER, INCLUDING SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING LOSS OF USE, DATA, OR PROFITS, EVEN IF ADVISED OF THE POSSIBILITY THEREOF, AND REGARDLESS OF WHETHER IN AN ACTION IN CONTRACT, TORT OR NEGLIGENCE, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

TRADEMARKS

NCD, NCDaudio, NCD Display Station, and NCDware are all trademarks of Network Computing Devices, Inc.

X Window System is a trademark of the Massachusetts Institute of Technology.

AUTHORS

Greg Renda, Jim Fulton, David Lemke

2.7.6.5 HDS Conference Manual Page

NAME

hdsconf - ViewStation video conferencing application

SYNOPSIS

hdsconf [-addressfile filename] [-dest connecthost] [-spkrvol (0 to 100)] [-micvol (1 to 100)] [-compress] [-nocompress] [-connector connector] [-informat format] [-brightness (-30 to 30)] [-contrast (-30 to 30)] [-saturation (-30 to 30)] [-tint (-30 to 30)] [-indev audioinputdevice] [-outdev audiooutputdevice] [-quality (0 to 100)] [-vidport networkportnum] [-audport networkportnum] [-noaudio] [-nosendvid] [-nosendaud] [-nosend] [-noscreech] [-toolkitOption ...]

DESCRIPTION

hdsconf is a powerful video teleconferencing application which allows multiple HDS ViewStation users to communicate using video and audio.

The HDS ViewCam should be connected to the ViewStation's VIDEO connector. An audio input device should be connected to the ViewStation's AUDIO IN connector. This could be the HDS ViewCam's audio cable, the HDS Headset, or the echo-suppressing CallPort Speaker / Microphone.

Once the audio devices are attached, the ViewStation must be configured to recognize them. The audio device types should be set in the HDS Setup menu. See the descriptions of the audio Input Device and Output Device below for more information.

Assuming the devices are properly attached and configured, a network connection can be made to another ViewStation using the Connect button. You can specify another host by IP address or by hostname (if the ViewStation Domain Name Services or Host Resolver File is properly configured).

Instead of using IP addresses or hostnames, you can arrange for a list of descriptive names to be presented. These descriptive names are placed in an address file. The default address file is /FXUtilities/hds-fx/hdsvideo/lib/HDSconf.list, where /FXUtilities refers to the current HDSware base directory. You can change the location of this file using the HDSconf.addressFileName X resource or the -addressfile command line option. Each line of the file consists of an IP address (or hostname) followed by a descriptive name, such as the ViewStation user's name or ViewStation's office location. Comment lines beginning with the '#' character are permitted.

The Disconnect button can be used to terminate a session.

The Freeze button can be used to freeze the outgoing video image. Pressing the button again clears the freeze.

The Send Video button must be used to actually transmit video. If the Send Video button is not highlighted, no video is being transmitted.

Depending upon the type of audio devices being used, either the Hold To Talk button or the Send Audio button will be available at any given time.

The Hold To Talk button is used to transmit audio. To avoid audio feedback, the speaker audio is temporarily suspended while using Hold To Talk. Releasing the

Hold To Talk button again stops audio transmitting, and restores the speaker audio. The Hold To Talk button is only presented if the current combination of Audio In Device and Audio Out Device might cause local feedback. This generally means that the HDS Internal Speaker or External Speakers are being used with an open microphone.

The Send Audio button must be used to transmit audio. Unlike the Hold To Talk button, the Send Audio button does not disable the incoming audio. However, the Send Audio button is only available when the Audio In Device and Audio Out Device do not allow local feedback. This generally means that the CallPort Spkr/Mic or the HDS Headset is being used.

The Speaker Volume slider controls the volume of the current Audio Out Device.

The Mic Volume slider controls the amount of gain applied to the Audio In Device. This volume should not normally need to be adjusted, except by very small amounts. When it is adjusted properly, the Audio meter level should generally stay in the safe green level, occasionally straying into the red zone for louder sounds. It should not be consistently in the red zone, or audio quality may suffer considerably.

The Audio meter indicates the level of the current Audio Input Device attached to the AUDIO IN connector. The meter is continuously updated, even when not transmitting audio. If the level is not registering green for most sounds, make sure the Audio In Device is set properly, the device is properly connected, and the Mic Volume slider is properly adjusted.

The Options menu contains entries for the Video Controls, Audio Controls, and Network Controls panels.

The Video Controls panel contains the following items.

The Compression panel item controls whether or not image compression should be used for the outgoing image. The compression lowers the amount of network usage, but can decrease the overall image quality.

The Input Connector item selects which type of video signal is attached to the ViewStation. The default is the Composite signal which is a single-plug jack. Another option is the S-Video connector which has multiple pins and which separates color information from luminance information. If set incorrectly, the ViewStation video display may appear black-and-white or otherwise incorrect.

The Input Format item indicates which type of video signal format is being used. The default is NTSC which is the standard for North America and several other regions. Other options are PAL which is dominant in Europe, and SECAM which is dominant in France. Once you determine the standard for your equipment, you will usually use one setting exclusively.

The Brightness, Contrast, Saturation, and Tint sliders control the video characteristics of the outgoing video window.

The Audio Controls panel contains the following items.

The Input Device item indicates the type of device attached to the ViewStation's AUDIO IN connector. This value must be set properly or the ViewStation's audio will not operate correctly. The initial value for the input device comes from the ViewStation's Setup client. It is better to make any changes to this value in the Setup client menu "Setup Options / User Preferences / Audio Device Settings / Audio In Device" and save the value for the future. NOTE: It is important to change this value whenever you change the audio device attached to the ViewStation. Otherwise, the ViewStation will not know what type of device is attached and severe audio quality problems will result. The input device can be set to "Unspecified", "HDS ViewCam", "HDS Headset", "CallPort Spkr / Mic", "HDS Microphone", "HDS Tuner", "VCR", or "Line Level Audio". The "Line Level Audio" category refers to the output level of many common home audio devices, such as CD players and cassette decks.

The Output Device item indicates the type of device attached to the ViewStation's AUDIO OUT connector. This value must be set properly or the ViewStation's audio will not operate correctly. As with the Input Device, the initial value for the output device comes from the ViewStation's Setup client. It is better to make any changes to this value in the Setup menu "Setup Options / User Preferences / Audio Device Settings / Audio Out Device" and save the value for the future. NOTE: It is important to change this value whenever you change the audio device attached to the ViewStation. Otherwise, the ViewStation will not know what type of device is attached and severe audio quality problems will result. The output device can be set to "HDS Internal Speaker", "External Speakers", "HDS Headset", or "CallPort Spkr / Mic".

The Network Controls panel contains the following items.

The Incoming Rate item shows the number of incoming video frames per second. This can be affected by the Quality and Compression settings on the other conferencing ViewStation.

The Incoming Net Use item shows the amount of network use for the incoming audio and video connections. This can be affected by the Quality and Compression controls on the other conferencing ViewStation.

The Outgoing Rate item shows the number of outgoing video frames per second. This can be affected by the local Quality and Compression controls.

The Outgoing Net Use item shows the amount of network use for the outgoing audio and video connections. This can be affected by the local Quality and Compression controls.

The Quality slider controls the general quality of the out- going image. It controls how much of a change must occur in each small area of the screen before the area must be retransmitted to the receiver. The default is 100, which means each area on the screen is always sent for each new video frame. Lower values mean that less of the screen must be sent for each new frame. This decreases the outgoing network usage, but can result in a poorer quality image. Values between 80 and 90 work well for typical cases, although lower values can be used when network bandwidth is at a premium.

OPTIONS

In addition to the usual Xt and Motif options, hdsconf program supports the following application-specific options.

-addressfile filename

Specify the file listing addresses and textual descriptions of ViewStations capable of conferencing. The default value is /FXUtilities/hds-fx/hdsvideo/lib/HDSconf.list.

-dest connecthost

Specify the host for initial connection.

-spkrvol (0 to 100)

Set the audio speaker output volume.

-micvol (1 to 100)

Set the audio input volume.

-compress

Use compression for outgoing video images.

-nocompress

Do not use compression for outgoing video images.

-connector connector

Select the video input connector. connector can be either Composite or SVideo.

-informat format

Select the video input signal format. format can be NTSC, PAL, or SECAM.

-brightness (-30 to 30)

Set the initial video window brightness.

-contrast (-30 to 30)

Set the initial video window contrast.

-saturation (-30 to 30)

Set the initial video window saturation.

-tint (-30 to 30)

Set the initial video window tint.

-indev audioinputdevice

Indicate the type of audio device attached to the ViewStation's AUDIO IN connector. It is better to save this value in the ViewStation's Setup client or to set it in the ViewStation's Configuration File. The audioinputdevice can be "Unspecified", "HDS ViewCam", "HDS Headset", "CallPort Spkr/Mic", "HDS Microphone", "HDS Tuner", "VCR", or "Line Level Audio". See the DESCRIPTION section for more information.

-outdev audiooutputdevice

Indicate the type of audio device attached to the ViewStation's AUDIO OUT connector. It is better to save this value in the ViewStation's Setup client or to set it in the ViewStation's Configuration File. The audiooutputdevice can be "HDS Internal Speaker", "External Speakers", "HDS Headset", or "CallPort Spkr/Mic". See the DESCRIPTION section for more information.

-quality (0 to 100)

See the DESCRIPTION section for more information.

-vidport networkportnum

Specify the Internet TCP network port number to use for the video connection. The default is 5100.

-audport networkportnum

Specify the Internet TCP network port number to use for the audio connection. The default is 5101.

-noaudio

Indicates that the ViewStation audio device should not be used. This is typically useful only if you wish to use the audio device for another purpose while using the video portion of the conference.

-nosendvid

Indicates that video should not be sent automatically as soon as the connection is established. The "Send Vid" button will still be available to start the video manually.

-nosendaud

Indicates that audio should not be sent automatically as soon as the connection is established. The "Send Aud" or "Hold to Talk" button will still be available to start the audio manually.

-nosend

Indicates that neither video nor audio should be sent automatically as soon as the connection is established. This is the sames as -nosendvid and -nosendaud.

-noscreech

Disables the use of screech detection. Screech detection attempts to determine when the speaker output signal level is too high and responds by lowering the Speaker Volume. Use this option to circumvent this feature.

X RESOURCES

In addition to the usual Xt and Motif resources, the hdsconf supports the following application-specific resources. The values and defaults are the same as the corresponding command line options.

HDSconf.addressListFileName: filename

Specify the file listing addresses and descriptions of conference-capable ViewStations. The default value is /FXUtilities/hds-fx/hdsvideo/lib/HDSconf.list.

HDSconf.addressList: addresses

As an alternative to using a file with addresses and descriptions, the addressList resource can be set directly. If set, this overrides the use of the addressListFile-Name. The resource should consist of one line for each ViewStation. Each line should contain the hostname (or host IP address) one or more spaces followed by a description. Lines should be separated by a newline character.

HDSconf.destAddress: connecthost

HDSconf.sendVid: Yes | No

HDSconf.sendAud: Yes | No

HDSconf.send: Yes | No

HDSconf.speakerVolume: (0 to 100)

HDSconf.micVolume: (1 to 100)

HDSconf.compression: Yes | No

HDSconf.inputConnector: connector

HDSconf.inputFormat: format

HDSconf.brightness: (-30 to 30)

HDSconf.contrast: (-30 to 30)

HDSconf.saturation: (-30 to 30)

HDSconf.quality: (0 to 100)

HDSconf.tint: (-30 to 30)

HDSconf.inputDevice: audioinputdevice

HDSconf.outputDevice: audiooutputdevice

HDSconf.doAudio: Yes | No

HDSconf.vidPort: networkportnum

HDSconf.audPort: networkportnum

HDSconf.screechDetection: Yes | No

HDSconf.meterSafeColor: Pixel

HDSconf.meterWarningColor: Pixel

COPYRIGHT

Copyright 1995, HDS Network Systems, Inc.

2.7.6.6 HDS Movie Manual Page

NAME

hdsmovie - view audio/video interleaved files

SYNOPSIS

hdsmovie [-f filename] [-loop] [-playvol (0 to 100)] [-invol (1 to 100)] [-brightness (-30 to 30)] [-contrast (-30 to 30)] [-saturation (-30 to 30)] [-audio (Mono or

Stereo)] [-outdev audiooutputdevice] [-nobuttons] [-noaudio] [-noloop] [-toolkitOption ...]

DESCRIPTION

hdsmovie plays audio/video interleaved files. The video can be raw or compressed with the Indeo compression algorithm. Version 3.2 of Indeo is supported.

By default, the program will create a main driver panel called HDS Movie.

The Movie File item should be filled with a ViewStation file path. This should usually begin with the /hostname prefix. Note that video files usually require a fast Access Protocol such as FTP in order to read the file quickly enough for rapid display. The usual default of TFTP is not sufficient. You may need to update the Remote Hosts entries for the host in the ViewStation Setup screen.

The Play Movie button causes the movie file to be played once.

The Loop Movie button causes the movie file to played continuously in a loop.

The Clear File button simply clears the current filename entry to allow a new one to be entered. It does not affect the file contents in any way.

When the video file playing is started, a second window is created. This window has the video file name as its title. The movie window offers several buttons and a volume slider.

The Freeze button can be used to freeze the current image. It affects only the video window in which it is pressed. Pressing the button again cancels the freeze.

The Mute button will mute the ViewStation audio output. Pressing the button again cancels the mute.

The SnapShot button will create a new window containing a still of the current image. The snapshot window contents can be easily captured to a file using common X image mani- pulations tools such as xwd(1) and xv. Note that if there are not enough free color entries available in the ViewStation's colormap, the snapshot window may not appear to have the correct colors. If this occurs, select the snapshot window with the window manager to see the correct colors. This can generally be avoided if at least 128 colors are available when the first snapshot is taken.

The Restart button causes the current movie to restart from the beginning.

The Done button terminates the movie window.

The Play Volume slider controls the output volume of the ViewStation. Note that the settings of other items in the Audio Controls panel can also have an important affect on the overall audio volume. Changes to this slider are reflected in all the hdsmovie windows.

The main HDS Movie panel also contains an Options menu which offers access to two other control dialogs, Video Controls and Audio Controls.

The Brightness, Contrast, and Saturation sliders control the video characteristics of the video window.

The Output Device item indicates the type of device attached to the ViewStation's AUDIO OUT connector. This value must be set properly or the ViewStation's audio will not operate correctly. The initial value for the output device comes from the ViewStation's Setup client. It is better to make any changes to this value in the Setup menu "Setup Options / User Preferences / Audio Device Settings / Audio Out Device" and save the value for the future. NOTE: It is important to change this value whenever you change the audio device attached to the ViewStation. Otherwise, the ViewStation will not know what type of device is attached and severe audio quality problems will result. The output device can be set to "HDS Internal Speaker", "External Speakers", "HDS Headset", or "CallPort Spkr / Mic".

OPTIONS

In addition to the usual Xt and Motif options, hdsmovie supports the following application-specific options.

-f filename

Gives the ViewStation path to the movie file to play. If specified, this file will be played automatically when the hdsmovie program is started.

-loop

Indicates that the file playback should be restarted when finished.

-playvol (0 to 100)

Set the initial play volume.

-brightness (-30 to 30)

Set the initial video window brightness.

-contrast (-30 to 30)

Set the initial video window contrast.

-saturation (-30 to 30)

Set the initial video window saturation.

-outdev audiooutputdevice

Indicate the type of audio device attached to the ViewStation's AUDIO OUT connector. It is better to save this value in the ViewStation's Setup client or to set it in the ViewStation's Configuration File. The audiooutputdevice can be "HDS Internal Speaker", "External Speakers", "HDS Headset", or "CallPort Spkr / Mic". See the DESCRIPTION section for more information.

-nobuttons

This option removes the control panels from the movie window.

-noaudio

This option brings up hdsmovie without using the ViewStation audio capabilities. This allows other programs to control the audio and avoids conflicts with other audio programs.

X RESOURCES

In addition to the usual Xt and Motif resources, the hdsmovie supports the following application specific resources. The values and defaults are the same as the corresponding command line options.

HDSvid.filename: (0 to 4)

HDSvid.loop: Yes | No

HDSvid.brightness: (-30 to 30)

HDSvid.contrast: (-30 to 30)

HDSvid.saturation: (-30 to 30)

HDSvid.outputType: audiooutputtype

HDSvid.showButtons: Yes | No

HDSvid.doAudio: Yes | No

COPYRIGHT

Copyright 1995, HDS Network Systems, Inc.

2.7.6.7 HDS NetTV Simulcast Manual Page

NAME

hdssimulcast - Multicast video & audio to other ViewStations

SYNOPSIS

hdssimulcast [-sendvid] [-sendaud] [-send] [-localvol (0 to 100)] [-outvol (1 to 100)] [-compressioncompressiontype] [-large] [-connector connector] [-informat format] [-brightness (-30 to 30)] [-contrast (-30 to 30)] [-saturation (-30 to30)] [-tint (-30 to 30)] [-indev audioinputdevice] [-outdev audiooutputdevice] [-netlimit (0 to 4096)] [-quality (0 to 100)] [-mcast address] [-vidport networkportnum] [-audport networkportnum] [-ttl multicasttimetolive] [-noaudio] [-toolkitOption ...]

DESCRIPTION

hdssimulcast uses IP Multicasting technology to efficiently transmit video and audio to a potentially large number of HDS NetTV Viewer applications running on other networked HDS ViewStations. It is similar to a real-life TV broadcasting-station, except that the network is being used to carry the television signals instead of the airwaves or special-purpose cable.

To view the multicast transmissions on other ViewStations, you must use the HDS NetTV Viewer (hdsnettv) program. The HDS NetTV Viewer program can be run on any ViewStation and does not require the optional HDS netVideo hardware. As discussed below, if you change the Simulcast Network Addresses, make sure to set the HDS NetTV Viewer Network Addresses to match.

The live video sources can include the output of a typical video camera (such as the HDS ViewCam), a VCR, the optional HDS Tuner output, or some other video-producing device. The video source should be connected to the VIDEO connector

on the ViewStation. If using the optional HDS Tuner device, be sure to connect the Video Out from the Tuner to the VIDEO connector on the ViewStation.

To transmit audio, an audio input device should be connected to the ViewStation's AUDIO IN connector. This could be the HDS ViewCam's audio cable, the HDS Headset, the echo- suppressing CallPort Speaker / Microphone, the HDS Tuner, or VCR audio output. An alternate audio output device can also be used.

Once the audio devices are attached, the ViewStation must be configured to recognize them. The audio device types should be set in the HDS Setup menu. See the descriptions of the audio Input Device and Output Device below for more information.

Before sending video or audio on the network, you may need to configure the Network Addresses under the Options menu. The network addresses correspond (roughly) to the broadcast "channel" carrying your HDS Simulcast program. You should choose addresses which do not conflict with any other simulcasts currently being used on your networks. See below for more details.

The Local Volume controls the volume of the audio speaker on the local machine. However, it does not affect the volume of the signal which is multicast over the network.

The Outgoing Volume slider controls the level of the audio signal which is sent out over the network. It adjusts the amount of gain applied to the Audio In Device. This volume should not normally need to be adjusted, except by very small amounts. When it is adjusted properly, the Audio meter level should generally stay in the safe green level, occasionally straying into the red zone for louder sounds. It should not be consistently in the red zone, or audio quality may suffer considerably. The Outgoing Volume slider will also affect the local volume.

The Audio meter indicates the level of the current Audio Input Device attached to the AUDIO IN connector. The meter is continuously updated, even when not transmitting audio. If the level is not registering green for most sounds, make sure the Audio In Device is set properly, the device is properly connected, and the Outgoing Volume slider is properly adjusted.

The Send Video button must be used to actually begin transmitting video over the network. Pressing the button again stops the video sending. Video is only transmitted if this button is highlighted.

The Send Audio button must be used to actually begin transmitting audio over the network. Pressing the button again stops the audio sending. Audio is only transmitted if this button is highlighted.

The Freeze button can be used to freeze the outgoing video image. Pressing the button again clears the freeze.

The Mute button can be used to mute the local audio temporarily. It does not affect the outgoing volume. To mute the outgoing volume, disable the Send Audio toggle.

The Options menu contains entries for the Video Controls, Audio Controls, Network Controls, and Network Addresses panels.

The Video Controls panel contains the following items.

The Compression panel item controls the type of video compression to use for the outgoing video transmission. The compression lowers the outgoing network usage, but can decrease the overall image quality. The compressions available are: None, CellB, and HDSGrey. The CellB compression scheme provides good image compression for color images. The HDSGrey compression scheme provides only black-and-white transmission.

The Outgoing Resolution item controls the size of the local and outgoing image. The two available values are 160 x 120 and 320 x 240. Using the smaller image size usually provides smoother motion and higher frame rates.

The Input Connector item selects which type of video signal is attached to the ViewStation. The default is the Composite signal which is a single-plug jack. Another option is the S-Video connector which has multiple pins and which separates color information from luminance information. If set incorrectly, the ViewStation video display may appear black-and-white or otherwise incorrect.

The Input Format item indicates which type of video signal format is being used. The default is NTSC which is the standard for North America and several other regions. Other options are PAL which is dominant in Europe, and SECAM which is dominant in France. Once you determine the standard for your equipment, you will usually use one setting exclusively.

The Brightness, Contrast, Saturation, and Tint sliders control the video characteristics of the video window.

The Audio Controls panel contains the following items.

The Input Device item indicates the type of device attached to the ViewStation's AUDIO IN connector. This value must be set properly or the ViewStation's audio will not operate correctly. The initial value for the input device comes from the ViewStation's Setup client. It is better to make any changes to this value in the Setup client menu "Setup Options / User Preferences / Audio Device Settings / Audio In Device" and save the value for the future. NOTE: It is important to change this value whenever you change the audio device attached to the ViewStation. Otherwise, the ViewStation will not know what type of device is attached and severe audio quality problems will result. The input device can be set to "Unspecified", "HDS ViewCam", "HDS Headset", "CallPort Spkr / Mic", "HDS Microphone", "HDS Tuner", "VCR", or "Line Level Audio". The "Line Level Audio" category refers to the output level of many common home audio devices, such as CD players and cassette decks.

The Output Device item indicates the type of device attached to the ViewStation's AUDIO OUT connector. This value must be set properly or the ViewStation's audio will not operate correctly. As with the Input Device, the initial value for the output device comes from the ViewStation's Setup client. It is better to make any changes to this value in the Setup menu "Setup Options / User Preferences / Audio

Device Settings / Audio Out Device" and save the value for the future. NOTE: It is important to change this value whenever you change the audio device attached to the ViewStation. Otherwise, the ViewStation will not know what type of device is attached and severe audio quality problems will result. The output device can be set to "HDS Internal Speaker", "External Speakers", "HDS Headset", or "CallPort Spkr / Mic".

The Network Controls panel contains the following items.

The Outgoing Rate item shows the number of outgoing video frames per second. This can be affected by the Quality and Compression settings.The Outgoing Net Use item shows the amount of network use for the outgoing audio and video connections. This can be affected by the Quality and Compression settings.

The Net Use Limit slider sets the maximum allowed network use. The default is 4096. Decreasing the value can result in decreased frame rates and in motion which is not as smooth. You can adjust this value to suit your needs.

The Quality slider controls the general quality of the out- going image. It generally controls how much of a change must occur in an area of the screen before it must be retransmitted to the receiver. The default is 100, which means the entire image is sent each frame. Lower values reduce the amount of outgoing network usage, but also reduce the quality of the image.

The Network Addresses panel contains the following items.

The Multicast Net Address item gives the destination IP address for the hdssimulcast output. The IP address is a multicast address if it is in the form aaa.bbb.ccc.ddd where aaa is between 224 and 239. If a multicast address is specified, then all receivers listening to that multicast group "channel" can receive the simulcast. If you want to specify a single receiver, you can use a non-multicast (uni-cast) address. The default value is 224.2.1.1. The hdsnettv viewer program's Multicast Net Address must be set to match in order to receive the transmissions.

The Video Net Port can be used to specify the UDP network port number for the outgoing video packets. The default value is 4444. Note that the the hdsnettv viewer program's Video Net Port must be set to match in order to receive the transmissions.

The Audio Net Port can be used to specify the UDP network port number for the outgoing video packets. The default value is 4445. Note that the the hdsnettv viewer program's Audio Net Port must be set to match in order to receive the transmissions.

OPTIONS

In addition to the usual Xt and Motif options, hdssimulcast supports the following application-specific options.

-sendvid

Indicates that video should be sent automatically.

-sendaud

Indicates that audio should be sent automatically.

-send

Indicates that both video and audio should be sent automatically.

-localvol (0 to 100)

Set the local audio speaker output volume.

-outvol (1 to 100)

Set the audio outgoing volume.

-compression compressiontype

Specify the type of compression to use. compressiontype can be None, CellB, or HDSGrey.

-large

Set the initial Outgoing Image Resolution to 320x240.

-connector connector

Select the video input connector. connector can be either Composite or SVideo.

-informat format

Select the video input signal format. format can be NTSC, PAL, or SECAM.

-brightness (-30 to 30)

Set the initial video window brightness.

-contrast (-30 to 30)

Set the initial video window contrast.

-saturation (-30 to 30)

Set the initial video window saturation.

-tint (-30 to 30)

Set the initial video window tint.

-indev audioinputdevice

Indicate the type of audio device attached to the ViewStation's AUDIO IN connector. It is better to save this value in the ViewStation's Setup client or to set it in the ViewStation's Configuration File. The audioinputdevice can be "Unspecified", "HDS ViewCam", "HDS Headset", "CallPort Spkr/Mic", "HDS Microphone", "HDS Tuner", "VCR", or "Line Level Audio". See the DESCRIPTION section for more information.

-outdev audiooutputdevice

Indicate the type of audio device attached to the ViewStation's AUDIO OUT connector. It is better to save this value in the ViewStation's Setup client or to set it in the ViewStation's Configuration File. The audiooutputdevice can be "HDS Internal Speaker", "External Speakers", "HDS Headset", or "CallPort Spkr/Mic". See the DESCRIPTION section for more information.

-quality (0 to 100)

See the DESCRIPTION section for more information.

-mcast address

Set the destination IP address. This can be a multicast or a standard unicast IP address.

-vidport networkportnum

Specify the Internet TCP network port number to use for the video connection. The default is 4444.

-audport networkportnum

Specify the Internet TCP network port number to use for the audio connection. The default is 4445.

-ttl multicasttimetolive

Set the time-to-live for multicast packets.

-noaudio

Don't use the ViewStation audio facilities. This can be used to prevent conflicts with other audio programs.

X RESOURCES

In addition to the usual Xt and Motif resources, the hdssimulcast program supports the following application- specific resources. The values and defaults are the same as the corresponding command line options.

HDSSimulcast.sendVid: Yes | No

HDSSimulcast.sendAud: Yes | No

HDSSimulcast.send: Yes | No

HDSSimulcast.localVolume: (0 to 100)

HDSSimulcast.outgoingVolume: (1 to 100)

HDSSimulcast.compression: None | CellB | HDSGrey

HDSSimulcast.inputConnector: connector

HDSSimulcast.inputFormat: format

HDSSimulcast.brightness: (-30 to 30)

HDSSimulcast.contrast: (-30 to 30)

HDSSimulcast.saturation: (-30 to 30)

HDSSimulcast.quality: (0 to 100)

HDSSimulcast.tint: (-30 to 30)

HDSSimulcast.inputDevice: audioinputdevice

HDSSimulcast.outputDevice: audiooutputdevice

HDSSimulcast.mcastAddress: networkportnum

HDSSimulcast.vidPort: networkportnum

HDSSimulcast.audPort: networkportnum

HDSSimulcast.ttl: timetolive

HDSSimulcast.doAudio: Yes | No

HDSconf.meterSafeColor: Pixel

HDSconf.meterWarningColor: Pixel

COPYRIGHT

Copyright 1995, HDS Network Systems, Inc.

2.7.6.8 HDS NetTV Viewer Manual Page

NAME

hdsnettv - Receive Multicast video & audio from hdssimulcast.

SYNOPSIS

hdsnettv [-volume (0 to 100)] [-color] [-grey] [-reduced] [-normal] [-enlarged] [-brightness (-30 to 30)] [-contrast (-30 to 30)] [-outdev audiooutputdevice] [-mcast address] [-vidport networkportnum] [-audport networkportnum] [-noaudio] [-toolkitOption ...]

DESCRIPTION

hdsnettv uses IP Multicasting technology to receive video and audio sent by another networked HDS ViewStation running the hdssimulcast program.

Use of the hdsnettv program does not require any special video adapter hardware. It can be used on any HDS ViewStation. The Network Address act as a kind of "channel- selector." The entries used by the HDS NetTV Viewer must match those used by the corresponding HDS NetTV Simulcast program.

The Volume slider controls the volume of the received audio. Volume is also affected by the Output Type setting of the Audio Controls. Note that if there is no audio, the hdssimulcast program might not be sending audio.

The Freeze button can be used to freeze the video image. Pressing the button again clears the freeze.

The Mute button can be used to mute the audio temporarily. Pressing the button again cancels the mute.

The SnapShot button will create a new window containing a still of the current image. The snapshot window contents can be easily captured to a file using common X image manipulations tools such as xwd(1) and xv. Note that if there are not enough free color entries available in the ViewStation's colormap, the snapshot window may not appear to have the correct colors. If this occurs, select the snapshot window with the window manager to see the correct colors. This can generally be avoided if at least 128 colors are available when the first snapshot is taken.

The Options menu contains entries for the Video Controls, Audio Controls, Network Controls, and Network Addresses panels.

The Video Controls panel contains the following items.

The Compression item indicates what kind of image compression was used for the incoming image. Compression lowers the amount of network usage, but can decrease the overall image quality. This must be adjusted by hdssimulcast which is sending the transmission. If HDSGrey image compression is being used, performance may be improved by setting the Display Type control to Grey.

The Incoming Resolution item indicates the size of the incoming image. The two available values are 160 x 120 and 320 x 240. Using the smaller image size usually provides smoother motion and higher frame rates. This must be adjusted in the sending hdssimulcast program.

The Display Type item indicates whether color or greyscale video should be displayed.

The Display Size item indicates whether the incoming image should be enlarged or reduced. Enlarging the image results in a larger image, but can reduce the performance and the frame rate. Reducing the image can improve frame rate if a large image is transmitted.

The Brightness and Contrast sliders control the video characteristics of the video window.

The Audio Controls panel contains the following items.

The Output Device item indicates the type of device attached to the ViewStation's AUDIO OUT connector. This value must be set properly or the ViewStation's audio will not operate correctly. The initial value for the output device comes from the ViewStation's Setup client. It is better to make any changes to this value in the Setup menu "Setup Options / User Preferences / Audio Device Settings / Audio Out Device" and save the value for the future. NOTE: It is important to change this value whenever you change the audio device attached to the ViewStation. Otherwise, the ViewStation will not know what type of device is attached and severe audio quality problems will result. The output device can be set to "HDS Internal Speaker", "External Speakers", "HDS Headset", or "CallPort Spkr / Mic".

The Network Controls panel contains the following items.

The Incoming Rate item shows the number of incoming video frames per second. This can be affected by the Quality and Compression settings on the hdssimulcast program.

The Incoming Net Use item shows the amount of network use for the incoming audio and video transmissions. This can be affected by the Quality and Compression settings on the hdssimulcast program.

The Network Addresses panel contains the following items.

The Multicast Net Address item gives the multicast IP address for listening on the network. The default is 224.2.1.1. The hdsnettv viewer program's Multicast Net Address must be set to match the multicast address used by the hdssimulcast program.

The Video Net Port can be used to specify the UDP network port number for the outgoing video packets. The default value is 4444. Note that the the hdsnettv viewer program's Video Net Port must be set to match the address used by the hdssimulcast program.

The Audio Net Port can be used to specify the UDP network port number for the outgoing video packets. The default value is 4445. Note that the the hdsnettv viewer program's Audio Net Port must be set to match the address used by the hdssimulcast program.

OPTIONS

In addition to the usual Xt and Motif options, hdssimulcast supports the following application-specific options.

-volume (0 to 100)

Set the local audio speaker output volume.

-color

Display color images.

-gray

Display grayscale video instead of color.

-reduced

Indicate that the displayed image should be one quarter of the transmitted size. This can result in faster performance for larger images.

-normal

Indicate that the displayed image should be the normal transmitted size.

-enlarged

Indicate that the displayed image should be four times larger than the transmitted size. This can result in a larger displayed image, but a lower frame rate.

-brightness (-30 to 30)

Set the initial video window brightness.

-contrast (-30 to 30)

Set the initial video window contrast.

-outdev audiooutputdevice

Indicate the type of audio device attached to the ViewStation's AUDIO OUT connector. It is better to save this value in the ViewStation's Setup client or to set it in the ViewStation's Configuration File. The audiooutputdevice can be "HDS Internal Speaker", "External Speakers", "HDS Headset", or "CallPort Spkr/Mic". See the DESCRIPTION section for more information.

-mcast address

Set the multicast listen IP address.

-vidport networkportnum

Specify the Internet TCP network port number to use for the video connection. The default is 4444.

-audport networkportnum

Specify the Internet TCP network port number to use for the audio connection. The default is 4445.

-noaudio

Don't use the ViewStation audio facilities. This can be used to prevent conflicts with other audio programs.

X RESOURCES

In addition to the usual Xt and Motif resources, the hdsnettv program supports the following application-specific resources. The values and defaults are the same as the corresponding command line options.

HDSNetTV.volume: (0 to 100)

HDSNetTV.displayType: Color | Grey

HDSNetTV.displaySize: Reduced | Normal | Enlarged

HDSNetTV.brightness: (-30 to 30)

HDSNetTV.contrast: (-30 to 30)

HDSNetTV.outputType: audiooutputtype

HDSNetTV.mcastAddress: networkportnum

HDSNetTV.vidPort: networkportnum

HDSNetTV.audPort: networkportnum

HDSNetTV.doAudio: Yes | No

COPYRIGHT

Copyright 1995, HDS Network Systems, Inc.

2.7.6.9 HDS Tuner Manual Page

NAME

hdstuner - control the optional HDS Tuner television tuner / receiver device.

SYNOPSIS

hdstuner [-device / dev / tty1(or2)] [-attenuate] [-noattenuate] [-mute] [unmute] [-antenna] [-cable] [-hrc_cable] [-icc_cable] [-finetune value (-30 to 30)] [-toolkitOption ...] [channel]

DESCRIPTION

hdstuner can be used to control the optional HDS Tuner television tuner / receiver. The HDS Tuner device must be attached to one of the ViewStation's serial ports. By default, Serial Port 1 is assumed. However, if Serial Port 2 is used instead, this can be specified using the -device option or the HDSTuner.device X resource.

Note that the hdstuner program does not directly display the video output. Use other programs such as hdsvid and hdssimulcast in conjuction with hdstuner to actually display the tuner output.

The hdstuner program allows the user to scan for available channels, select channels, and fine tune reception. The type of signal can be selected (antenna, cable, etc) as well as the audio output level of the HDS Tuner device. Audio mute control is also available.

The Scan For Channels button will scan through all channels looking for an acceptable signal level. All channels found will be added to the current list of presets. The list of presets can also be initially configured using the HDSTuner.channelPresets X resource. In the absence of a pre-configured list of channel presets, the user will usually want to Scan For Channels after bringing up the HDStuner application. Make sure the TV Input has been set appropriately before scanning.

The Add/Delete Channel button is used to add or delete the current channel in the preset list.

The Mute button disables the audio output of the HDS Tuner device. The HDSTuner.mute X resource can be used to set the initial state of this button.

The Exit button exits the entire HDStuner application. The state of the HDS Tuner device remains the same. The audio mute state and selected channel remain the same. If the HDStuner application is restarted, it will by default display the settings which had been established previously. This default behavior can be overridden with command line options and X resources to force a particular initial configuration. If power is removed from the HDS Tuner device, the previous settings are lost.

The Up Arrow and Down Arrow buttons are used to skip through the list of preset channels. Channels which have not been scanned or specified in the initial list of channel presets cannot be reached via the arrow buttons. The Scan For Channels button or the direct keypad should be used to reach such channels.

The Numeric Keypad is used to go directly to a channel, regardless of whether it is in the current channel preset list.

The Channel Fine Tuning slider can be used to make fine tuning adjustments. Each channel is tuned independently of the others.

The TV Input option controls which type of signal is being provided on the Tuner's Antenna In connector. Antenna refers to standard over-the-air frequencies. Cable refers to common CATV channels. HRC Cable refers to Harmonically Related Carrier encoded cable signals. ICC Cable refers to Incremental Coherent Carrier encoded cable signals.

OPTIONS

-device /dev/tty1(or2)

Indicate the ViewStation device to which the HDS Tuner is attached. The default is /dev/tty1 for Serial Port 1. Serial Port 2 may also be used.

-attenuate

-noattenuate

Indicate whether the Audio Out level of the HDS Tuner device should be line level (noattentuate) or microphone level (attenuate). By default, the hdstuner program will automatically set for the attenuated level.

-mute

-unmute

Indicate whether the Tuner's audio output should be initially muted. The default is unmute.

-antenna

-cable

-hrc_cable

-icc_cable

Set the TV Input for the tuner.

-finetune value (-30 to 30)

Set the initial finetune value for tuner. Value ranges from -30 to 30.

-toolkitOption ...

Set any standard Xt Intrinsics toolkit option. These include the -display and -xrm options.

channel

Set the initial startup channel for the tuner.

X RESOURCES

In addition to the usual Xt and Motif resources, the hdstuner supports the following application specific resources:

HDSTuner.device: /dev/tty1 | /dev/tty2

Specify the serial port to which the tuner is attached. The default is Serial Port 1, /dev/tty1.

HDSTuner.tvInput: antenna | cable | hrc_cable | icc_cable

Specify the type of the video signal attached to the Antenna In. The default is the previous setting or antenna.

HDSTuner.audioLevel: microphone | line

Specify the level of the Tuner device's Audio Out connector. The default is microphone.

HDSTuner.channelPresets: channel:finetune ...

Give an initial list of channels the the tuner to use a presets. This avoids the need to scan for channels and also allows the user to have the most-viewed channels available at startup. Each entry has a channel number and a finetune value separated by a colon. The finetune value must be between -30 and 30.

Example: HDSTuner.channelPresets: 3:0 6:0 10:0

HDSTuner.channel: channel_number

Specify the initial channel for the tuner.

HDSTuner.fineTune: value

Specify the initial finetune value for the tuner.

HDSTuner.mute: True | False

Specify whether the tuner's audio output should initially be muted.

SEE ALSO

hdsaudio(1) hdssimulcast(1) hdsvid(1)

COPYRIGHT

Copyright 1995, HDS Network Systems, Inc.

2.7.6.10 HDS Video Manual Page

NAME

hdsvid - view multiple live video windows.

SYNOPSIS

hdsvid [-windows (0 to 4)] [-playvol (0 to 100)] [-invol (1 to 100)] [-inport connector] [-informat format] [-hires] [-lores] [-resolution resolution] [-brightness (-30 to 30)] [-contrast (-30 to 30)] [-saturation (-30 to 30)] [-tint (-30 to 30)] [-audio (Mono or Stereo)] [-indev audioinputdevice] [-outdev audiooutputdevice] [-forceaspect] [-noaspect] [-nobuttons] [-noaudio] [-toolkitOption ...]

DESCRIPTION

hdsvid can display high-quality live video in up to four simultaneous windows. Windows can be independently resized and overlapped. Still-frame snapshots of each window can be taken and the contents of these snapshot windows can be easily saved to a file using common X Window System tools.

The live video sources can include the line-level output of a video camera, a VCR, television receiver monitor output, optional HDS Tuner TV Receiver output, or other video- producing device. The video source should be connected to the VIDEO connector on the ViewStation.

To use audio, an audio input device should be connected to the ViewStation's AUDIO IN connector. This could be the HDS ViewCam's audio cable, the HDS Headset, the echo-suppressing CallPort Speaker / Microphone, the HDS Tuner, VCR audio output, or some other audio device. An alternate audio output device can also be used.

Once the audio devices are attached, the ViewStation must be configured to recognize them. The audio device types should be set in the HDS Setup menu. See the descriptions of the audio Input Device and Output Device below for more information.

By default, the client will create two windows. The first is the main driver panel, HDS Video. The second is the initial video window, Window 1. The initial video window offers several buttons and a volume slider.

The Freeze button can be used to freeze the current image. It affects only the video window in which it is pressed. Other video windows will not be affected. Pressing the button again cancels the freeze.

The Mute button will mute the ViewStation audio output. Pressing the Mute button on one video window affects the others. Pressing the button again cancels the mute.

The SnapShot button will create a new window containing a still of the current image. The snapshot window contents can be easily captured to a file using common X image manipulations tools such as xwd(1) and xv. Note that if there are not enough free color entries available in the ViewStation's colormap, the snapshot window may not appear to have the correct colors. If this occurs, select the snapshot window with the window manager to see the correct colors. This can generally be avoided if at least 128 colors are available when the first snapshot is taken.

The Done button terminates the video window. Other video windows (and the main HDS Video panel) are not affected. (To terminate the hdsvid session, exit the main HDS Video window.

The Play Volume slider controls the output volume of the ViewStation. Note that the settings of other items in the Audio Controls panel can also have an important affect on the overall audio volume. Changes to this slider are reflected in all the hdsvid windows.

The HDS Video panel contains a Start New Window button and two volume control sliders.

The Start New Window button creates a new video window. The window will be either HighResolution or LowResolution based on the setting of the New Window Type item in the Video Controls panel. Two HighResolution windows can be displayed at one time. Four LowResolution windows can be displayed at one time. When the maximum number of windows are in use, the Start New Window button will be grayed out. It will be active again when a video window has been closed.

The Input Volume slider adjusts the amount of gain applied to the Audio In Device. This volume should not normally need to be adjusted, except by very small amounts. When it is adjusted properly, the Audio meter level should generally stay in the safe green level, occasionally straying into the red zone for louder sounds.

The Audio meter indicates the level of the current Audio Input Device attached to the AUDIO IN connector. If the level is not registering green for most sounds, make sure the Audio In Device is set properly, the device is properly connected, and the Input Volume slider is properly adjusted.

The HDS Video panel contains an Options menu which offers access to two other control dialogs, Video Controls and Audio Controls.

The Video Controls panel contains the following items.

The Input Connector item selects which type of video signal is attached to the ViewStation. The default is the Composite signal which is a single-plug jack. Another option is the S-Video connector which has multiple pins and which separates color information from luminance information. If set incorrectly, the ViewStation video display may appear black-and-white or otherwise incorrect.

The Input Format item indicates which type of video signal format is being used. The default is NTSC which is the standard for North America and several other regions. Other options are PAL which is dominant in Europe, and SECAM which is dominant in France. Once you determine the standard for your equipment, you will usually use one setting exclusively.

The New Window Type can be set to HighResolution or LowResolution. This affects the amount of vertical resolution for the image. If HighResolution is specified the video signal is deinterlaced and both fields are displayed.

The Brightness, Contrast, Saturation, and Tint sliders control the video characteristics of all the video windows.

The Audio Controls panel contains the following items.

The Input Device item indicates the type of device attached to the ViewStation's AUDIO IN connector. This value must be set properly or the ViewStation's audio will not operate correctly. The initial value for the input device comes from the ViewStation's Setup client. It is better to make any changes to this value in the Setup client menu "Setup Options / User Preferences / Audio Device Settings / Audio In Device" and save the value for the future. NOTE: It is important to change this value whenever you change the audio device attached to the ViewStation. Otherwise, the ViewStation will not know what type of device is attached and severe audio quality problems will result. The input device can be set to "Unspecified", "HDS ViewCam", "HDS Headset", "CallPort Spkr / Mic", "HDS Microphone", "HDS Tuner", "VCR", or "Line Level Audio". The "Line Level Audio" category refers to the output level of many common home audio devices, such as CD players and cassette decks.

The Audio Type item indicates whether the input is Mono or Stereo. If the default value of Mono is in effect, the left channel is used.

The Output Device item indicates the type of device attached to the ViewStation's AUDIO OUT connector. This value must be set properly or the ViewStation's audio will not operate correctly. As with the Input Device, the initial value for the output device comes from the ViewStation's Setup client. It is better to make any changes to this value in the Setup menu "Setup Options / User Preferences / Audio Device Settings / Audio Out Device" and save the value for the future. NOTE: It is important to change this value whenever you change the audio device attached to the ViewStation. Otherwise, the ViewStation will not know what type of device is attached and severe audio quality problems will result. The output device can be set to "HDS Internal Speaker", "External Speakers", "HDS Headset", or "CallPort Spkr / Mic".

The Play Volume and Input Volume sliders are duplicates of those found on other control panels. Modifications to these sliders automatically adjust the other sliders.

OPTIONS

In addition to the usual Xt and Motif options, hdsvid supports the following application-specific options.

-windows (0 to 4)

Indicate the number of video windows displayed at startup. The default is 1. Only two widows can be displayed initially unless -lores is used.

-playvol (0 to 100)

Set the initial play volume.

-invol (1 to 100)

Set the initial amplification volume for the AUDIO IN source.

-inport connector

Select the video input connector. connector can be either Composite or SVideo.

-informat format

Select the video input signal format. format can be NTSC, PAL, or SECAM.

-hires

Indicate that the default window type should be HighResolution.

-lores

Indicate that the default window type should be LowResolution.

-resolution resolution

Set the default resolution. resolution can be either HighResolution (the default) or LowResolution.

-brightness (-30 to 30)

Set the initial video window brightness.

-contrast (-30 to 30)

Set the initial video window contrast.

-saturation (-30 to 30)

Set the initial video window saturation.

-tint (-30 to 30)

Set the initial video window tint.

-audio Mono | Stereo

Select either Mono or Stereo audio processing.

-indev audioinputdevice

Indicate the type of audio device attached to the ViewStation's AUDIO IN connector. It is better to save this value in the ViewStation's Setup client or to set it in the ViewStation's Configuration File. The audioinputdevice can be "Unspecified", "HDS ViewCam", "HDS Headset", "CallPort Spkr/Mic", "HDS Microphone", "HDS Tuner", "VCR", or "Line Level Audio". See the DESCRIPTION section for more information.

-outdev audiooutputdevice

Indicate the type of audio device attached to the ViewStation's AUDIO OUT connector. It is better to save this value in the ViewStation's Setup client or to set it in the ViewStation's Configuration File. The audiooutputdevice can be "HDS Internal Speaker", "External Speakers", "HDS Headset", or "CallPort Spkr/Mic". See the DESCRIPTION section for more information.

-forceaspect

This option requires that video windows maintain the normal 4x3 aspect ratio. This is the default.

-noaspect

This option allows video windows to have any aspect ratio selected by the user. This will cause the video contents to stretch & shrink in the vertical and horizontal directions. This can distort the image, but will fill the window.

-nobuttons

This option removes the control panels from the video windows.

-noaudio

This option brings up hdsvid without using the ViewStation audio capabilities. This allows other programs to control the audio and avoids conflicts with other audio programs.

X RESOURCES

In addition to the usual Xt and Motif resources, the hdsvid supports the following application specific resources. The values and defaults are the same as the corresponding command line options.

HDSvid.windows: (0 to 4)

HDSvid.playVolume: (0 to 100)

HDSvid.inputVolume: (1 to 100)

HDSvid.inputPort: connector

HDSvid.inputFormat: format

HDSvid.resolution: resolution

HDSvid.brightness: (-30 to 30)

HDSvid.contrast: (-30 to 30)

HDSvid.saturation: (-30 to 30)

HDSvid.tint: (-30 to 30)

HDSvid.audioType: Mono | Stereo

HDSSimulcast.inputDevice: audioinputdevice

HDSSimulcast.outputDevice: audiooutputdevice

HDSvid.forceAspect: Yes | No

HDSvid.showButtons: Yes | No

HDSvid.doAudio: Yes | No

HDSvid*Win1*width: width

HDSvid*Win1*height: height

HDSvid*Win2*width: width

HDSvid*Win2*height: height

HDSvid*Win3*width: width

HDSvid*Win3*height: height

HDSvid*Win4*width: width

HDSvid*Win4*height: height

Sets the width and height of the actual video image. The surrounding window will size itself to fit around it. The default width is 640 pixels and the default height is 480 pixels.

HDSvid*Window 1*geometry: geometry

HDSvid*Window 2*geometry: geometry

HDSvid*Window 3*geometry: geometry

HDSvid*Window 4*geometry: geometry

Sets the location and size of the video window. The size includes the control buttons and other framing. The default size is based on the size of the incorporated video image.

COPYRIGHT

Copyright 1995, HDS Network Systems, Inc.

2.7.6.11 HDS Record Client Man Page

NAME

hdsrecord - record an audio file from the HDS ViewStation

SYNOPSIS

hdsrecord [-s #] [-p #] [-e ULAW | ALAW | LINEAR] [-c #] [-r] [-w] [-t [[hh:]mm:]ss[.ddd]] [-i info] [-a] [-f] [-v #] [-d dev] [[file]]

DESCRIPTION

hdsrecord will record audio files from an HDS ViewStation. The files can be recorded in standard Sun / Next format or can be raw binary digital audio format. The data format and sampling rate can be specified as command line options. If no filenames are specified, the standard output is used.

Typically the program will be invoked via the Remote Shell facility from a host computer. See the examples below.

Recording is usually terminated by using the interrupt character (usually Control-C). In order to provide more control over the recording start time, hdsrecord will optionally wait for the user to press the enter key before recording is actually started.

The hdsrecord program can only be used on ViewStations equipped with the optional HDS MultiMedia Audio Card.

The hdsplay program can be used to playback recorded audio files.

OPTIONS

-s rate

Specify the date sampling rate, in samples per second. The default value is 8000. Acceptable values are: 5512 6615 8000 9600 11025 16000 18900 22050 27428 32000 33075 37800 44100 and 48000.

-p precision

Specify the sample precision for each channel, measured in bits. Acceptable values are 8 and 16 bits. The default is 8 bits. A 16 bit precision can only be used for LINEAR encodings and 16 bit data will be in most-significant-byte-first order.

-e encoding

Specify the data sample encoding type. The default value is ULAW. LINEAR may also be used.

-c channels

Specify the number of interleaved channels. Acceptable values are 1 for mono and 2 for stereo. The default value is 1.

-r

Treat files as raw input data. No attempt will be made to write a file header.

-w

Wait for the user to press the enter key before starting to record. This allows finer control over the start of recording.

-v vol

Set the recording volume to vol before recording begins. vol is an integer value between 0 and 100, inclusive. If this argument is not specified, the recording volume remains at the level most recently set by any process.

-t [[hh:]mm:]ss[.ddd]

Set the maximum recording time.

-i info

Set an information string describing the file. This string will be saved in the audio file header.

-a

Append to an already existing audio file.

-f

Ignore sample rate differences on append.

-d dev

Specify an alternate audio device to which output should be directed. If the -d option is not specified, /dev/audio is the default audio device.

-?

Print a command line usage help message.

file

The name of the output file. This must be a ViewStation local path name. If the file name is omitted (or is "-") the file is written to the standard output.

EXAMPLES

To record a sound file from a host computer, the following commands can be issued in the host's shell. These examples assume that the remote shell command on your host computer is rsh. Your computer may use a different command, such as rcmd.

example% rsh fxhost hdsrecord > file.au

example% rsh fxhost hdsrecord -s 44100 -p 16 -e linear -c 2 - > cdquality.au

example% rsh fxhost hdsrecord -t 1:00 -r - > oneminute.raw

SEE ALSO

hdsaudio(1) hdsplay(1)

FVWM - Virtual Window Manager

This chapter describes the FVWM Virtual Window Manager and its associated utility modules. These all run as ViewStation local clients.

Section 2.0 - HDS ViewStation Local Clients

2.8 FVWM Virtual Window Manager

This chapter contains the manual pages for the FVWM Virtual Window Manager version 1.2 and its associated utility modules:

- FvwmAudio
- FvwmAuto
- FvwmBacker
- FvwmBanner
- FvwmClean
- FvwmDebug
- Fvwm Goodstuff
- FvwmIconBox
- FvwmIdent
- FvwmPager
- FvwmScroll
- FvwmWinlist
- xpmroot (often used by FVWM) is described in Section 2.6.19.

2.8.1 FVWM Manual Page

NAME

fvwm - F(?) Virtual Window Manager for X11

SYNOPSIS

fvwm [options]

DESCRIPTION

Fvwm is a window manager for X11. It is a derivative of twm, redesigned to minimize memory consumption, provide a 3-D look to window frames, and provide a simple virtual desktop. Memory consumption is estimated at about one-half to one- third the memory consumption of twm, due primarily to a redesign of twm's method of storing mouse bindings. In addition, many of the configurable options of twm have been removed.

The name "FVWM" used to stand for something, but I forgot what. (Feeble, famous, foobar? It doesn't really matter, this is an acronym based society anyway.)

SPECIAL NOTE FOR XFREE86 USERS

XFree86 provides a virtual screen whose operation can be confusing when used in conjunction with fvwm. With XFree86 all windows which appear on the virtual screen actually get drawn into video memory (whether or not they appear on the physical screen), so the virtual screen size is limited by available video memory.

With fvwm's virtual desktop, windows which do not appear on the screen do not actually get drawn into video RAM. The size of the virtual desktop is limited to about 32,000 by 32,000 pixels, but it is probably impractical to use a virtual desktop more than about 5 times the visible screen in each direction. Note that memory usage is a function of the number of windows which exist - the size of the desktop makes no difference.

When becoming familiar with fvwm it is recommended that you disable XFree86's virtual screen by setting the virtual screen size to the physical screen size. After you become familiar with fvwm you may want to re-enable XFree86's virtual screen.

COPYRIGHTS

Since fvwm is derived from twm code it shares twm's copy- rights.

fvwm is copyright 1988 by Evans and Sutherland Computer Corporation, Salt Lake City, Utah, and 1989 by the Massachusetts Institute of Technology, Cambridge, Massachusetts, All rights reserved. It is also copyright 1993 and 1994 by Robert Nation.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the names of Evans & Sutherland and M.I.T. not be used in advertising in publicity pertaining to distribution of the software without specific, written prior permission.

ROBERT NATION, EVANS & SUTHERLAND, AND M.I.T. DISCLAIM ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL EVANS & SUTHERLAND OR M.I.T. BE LIABLE FOR ANY SPE-

CIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTUOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

ANATOMY OF A WINDOW

Fvwm puts a decorative border around most windows. This border consists of a bar on each side and a small "L" shaped section on each corner. There is an additional top bar called the title bar which is used to display the name of the window. In addition, there are up to 10 title-bar buttons. The top, side, and bottom bars are collectively known as the side-bars. The corner pieces are called the frame.

Unless the standard defaults files are modified, pressing mouse button 1 in the title or side-bars will begin a move operation on the window. Pressing button 1 in the corner frame pieces will begin a resize operation. Pressing button 2 anywhere in the border brings up an extensive list of window operations.

Up to ten title-bar buttons may exist. Their use is completely user definable. The default configuration has a title-bar button on each side of the title-bar. The one on the left is used to bring up a list of window options, regardless of which mouse button is used. The one on the right is used to iconify the window. The number of title-bar buttons used depends on which ones have mouse actions bound to them. See the section on the "Mouse" configuration parameter below.

THE VIRTUAL DESKTOP

Fvwm provides multiple virtual desktops for users who wish to use them. The screen is a viewport onto a desktop which may be larger than the screen. Several distinct desktops can be accessed (concept: one desktop for each project, or one desktop for each application, when view applications are distinct). Since each desktop can be larger than the physical screen, windows which are larger than the screen or large groups of related windows can easily be viewed.

The size of the virtual desktops can be specified at startup. All virtual desktops must be the same size. The total number of distinct desktops need not be specified, but is limited to approximately 4 billion total. All windows on the current desktop can be displayed in a Pager, a miniature view of the current desktop. Windows which are not on the current desktop can be listed, along with their geometries, in a window list, accessible as a pop-up menu.

"Sticky" windows are windows which transcend the virtual desktop by "Sticking to the screen's glass." They always stay put on the screen. This is convenient for things like clocks and xbiff's, so you only need to run one such gadget and it always stays with you.

Window geometries are specified relative to the current viewport. That is:

xterm -geometry +0+0

will always show up in the upper-left hand corner of the visible portion of the screen. It is permissible to specify geometries which place windows on the virtual

desktop, but off the screen. For example, if the visible screen is 1000 by 1000 pixels, and the desktop size is 3x3, and the current viewport is at the upper left hand corner of the desktop, then invoking:

xterm -geometry +1000+1000

will place the window just off of the lower right hand corner of the screen. It can be found by moving the mouse to the lower right hand corner of the screen and waiting for it to scroll into view. There is currently no way to cause a window to map onto a desktop other than the currently active desk.

A geometry specified as something like:

xterm -geometry -5-5

will generally place the window's lower right hand corner 5 pixels from the lower right corner of the visible portion of the screen. Not all applications support window geometries with negative offsets.

Some applications, like xterm and xfontsel, allow the user to specify the start-up desk on the command line:

xterm -xrm "*Desk:1"

will start an xterm on desk number 1. Not all applications understand this option, however.

INITIALIZATION

During initialization, fvwm will search for a configuration file which describes key and button bindings, and a few other things. The format of these files will be described later. First, fvwm will search for a file named .fvwmrc in the users home directory. Failing that, it will look for /usr/lib/X11/fvwm/system.fvwmrc for system-wide defaults. If that file is not found, fvwm will exit.

Fvwm will set two environment variables which will be inherited by its children. These are $DISPLAY which describes the display on which fvwm is running. $DISPLAY may be unix:0.0 or :0.0, which doesn't work too well when passed through rsh to another machine, so $HOSTDISPLAY will also be set and will use a network-ready description of the display. $HOST- DISPLAY will always use the TCP/IP transport protocol (even for a local connection) so $DISPLAY should be used for local connections, as it may use Unix-domain sockets, which are faster.

SHAPED WINDOWS

If you typically use shaped windows such as xeyes or oclock, you have several options. You can make them all undecorated (NoBorder oclock and NoTitle oclock, for example) or you can use the default configuration and leave them decorated, in which case a decorative border and a solid-color backdrop are shown. Alternately, you can compile in the SHAPE extensions by changing a flag in the Makefile, in which case you get the shaped window with no backdrop, and a title bar floats above the window. The shaped window extensions increase the window manager's memory consumption by about 60 Kbytes when no shaped windows are present but have little effect when shaped windows are present.

ICONS

The basic Fvwm configuration uses monochrome bitmap icons, similar to twm. If XPM extensions are compiled in, then color icons similar to ctwm, MS-Windows, or the Macintosh icons can be used. In order to use these options you will need the XPM package, as described in the Makefile.noImake and the Imakefile.

If both the SHAPE and XPM options are compiled in you will get shaped color icons, which are very spiffy.

MODULES

A module is a separate program which runs as a separate Unix process but transmits commands to fvwm to execute. Future releases are expected to provide a means for these modules to extract window information from fvwm. Users can write their own modules to do any weird or bizarre manipulations without affecting the integrity of fvwm itself.

Modules MUST be spawned by fvwm so that it can set up two pipes for fvwm and the module to communicate with. The pipes will already be open for the module when it starts and the file descriptors for the pipes are provided as command line arguments.

Modules can be spawned during fvwm initialization via the Module option, or at any time during the X session by use of the Module built-in. Modules can exist for the duration of the X session, or can perform a single task and exit. If the module is still active when fvwm is told to quit, then fvwm will close the communication pipes and wait to receive a SIGCHLD from the module, indicating that it has detected the pipe closure and has exited. If modules fail to detect the pipe closure fvwm will exit after approximately 30 seconds anyway. The number of simultaneously executing modules is limited by the operating system's maximum number of simultaneously open files, usually between 60 and 256.

Modules simply transmit text commands to the fvwm built-in command engine. Text commands are formatted just as in the case of a mouse binding in the .fvwmrc setup file. Certain auxiliary information is also transmitted, as in the sample module GoodStuff. The GoodStuff module is documented in its own man page.

ICCCM COMPLIANCE

Fvwm attempts to be ICCCM 1.1 compliant. As of this (1.20l) colormap handling is not completely ICCCM compliant. In addition, ICCCM states that it should be possible for applications to receive ANY keystroke, which is not consistent with the keyboard shortcut approach used in fvwm and most other window managers.

M4 PREPROCESSING

If fvwm is compiled with the M4 option, fvwm uses m4(1) to preprocess its setup files before parsing. This way you can use m4 macros to perform operations at runtime. This makes it very easy to work with different displays with different characteristics.

For example, depending on your mood, you might want different color schemes. One way of doing this is by using the -m4opt to specify your mood. For a sunny

mood use -m4opt -DSunny; for a dark mood use -m4opt -DDark. Your .fvwmrc file might then contain:

```
ifdef('Sunny',`
StdForeColor Black
StdBackColor LightSkyBlue
HiForeColor yellow
HiBackColor PeachPuff1
PagerBackColor BlanchedAlmond ')

ifdef('Dark',`
StdForeColor Black
StdBackColor #60a0c0
HiForeColor black
HiBackColor #c06077
PagerBackColor #5c54c0
PagerForeColor orchid
StickyForeColor Black
StickyBackColor #60c0a0 ')
```

The following m4 symbols are predefined by fvwm:

BITS_PER_RGB The number of significant bits in an RGB color. (log base 2 of the number of distinct colors that can be created. This is often different from the number of colors that can be displayed at once.)

CLASS Your visual class. Will return one of StaticGray, GrayScale, StaticColor, PseudoColor, TrueColor, DirectColor, or, if it cannot determine what you have, NonStandard.

CLIENTHOST The machine that is running the clients.

COLOR This will be either 'Yes' or 'No'. This is just a wrapper around the CLASS definition. Returns 'Yes' on *Color and 'No' on StaticGray and GrayScale.

FVWMDIR This is set to the path where the modules were configured to be installed.

FVWM_VERSION This is a string containing the version of fvwm.

HEIGHT The height of your display in pixels.

HOME The user's home directory. Obtained from the environment.

HOSTNAME The canonical hostname running the clients (ie. a fully-qualified version of CLIENTHOST).

OPTIONS This is a string of compile time options used. Each option is separated from the other by a space.

PLANES The number of bit planes your display supports in the default root window.

RELEASE The release number of your X server. For MIT X11R5 this is 5.

REVISION The X minor protocol revision. As seen by ProtocolRevision(3).

SERVERHOST This variable is set to the name of the machine that is running the X server.

TWM_TYPE Tells which twm offshoot is running. It will always be set to the string "fvwm" in this program. This is useful for protecting parts of your .twmrc file that fvwm proper won't understand (like WorkSpaces) so that it is still usable with other twm programs.

USER The name of the user running the program. Obtained from the environment.

VENDOR The vendor of your X server. For example: MIT X Consortium.

VERSION The X major protocol version. As seen by ProtocolVersion(3).

WIDTH The width of your display in pixels.

X_RESOLUTION The X resolution of your display in pixels per meter.

Y_RESOLUTION The Y resolution of your display in pixels per meter.

You may well find that if you research the m4(1) manual well and understand the power of m4, this will be a very useful and powerful tool. But if you use any of the symbols which are predefined by m4, you are in severe danger! For example, Sun's m4 predefines include, so if you use that name in your .fvwmrc, you are out of luck. The correct solution to this problem is to put a set of quotes around the troublesome word: 'include'.

To help alleviate this problem, the following options may be useful. To change the quoting characters used by m4, use the options -m4-squote and -m4-equote. Be sure to specify both options otherwise m4 will be confused. When these are given, a changequote macro is given before the users fvwmrc file is processed.

NOTE: Some versions of m4 are broken with respect to changing quoting characters and included files. When the quoting strings are longer than one character, the macro "include(<<file>>)", where "<<" and ">>" are the quoting characters, contains extra characters around the contents of the included file. This will confuse fvwm. SunOS 4.1.3 is known to have this problem.

If you are using GNU m4 an additional option is available.

By specifying -m4-prefix when starting fvwm, m4 is instructed to prefix all builtin macros with m4_. Thus, include becomes m4_include.

The availability of the m4 preprocessing is subject to the compilation define M4.

OPTIONS

These are the command line options that are recoginzed by fvwm:

-f config_file

Causes fvwm to use config_file in the user's home directory instead of .fvwmrc as the window manager configuration file.

-debug

Puts X transactions in synchronous mode, which dramatically slows things down, but guarantees that fvwm's internal error messages are correct.

-d displayname

Manage the display called "displayname" instead of the name obtained from the environment variable $DISPLAY.

-s

On a multi-screen display, run fvwm only on the screen named in the $DISPLAY environment variable or provided through the -d option. Normally, fvwm will attempt to start up on all screens of a multi-screen display.

-version

Print the version of fvwm to stderr.

The following options are available only if fvwm is compiled with the M4 option.

-no-m4

Do not use m4 to preprocess the .fvwmrc. The default is to preprocess the startup file using m4(1).

-m4-prefix

If GNU m4 is available, cause m4 to prefix all builtin commands with m4_.

-m4opt option

Pass this option to m4. The option can be any string of characters without spaces. This option can occur multiple times. If GNU m4 is available, DO NOT pass the -P option here. Use -m4-prefix instead.

-m4-squote string

Use this given string as the starting quote characters. You must also specify -m4-equote.

-m4-equote string

Use this given string as the ending quote characters. You must also specify -m4-squote.

-m4prog path

Use path as the location of the desired m4 processor. By default, m4prog is set to "m4" which must exist somewhere on the user's path. This option allows the user to explicitly choose the version of m4 to use.

CONFIGURATION FILES

The configuration file is used to describe mouse and button bindings, colors, the virtual display size, and related items. This section describes the configuration

options. Lines beginning with '#' will be ignored by fvwm. Lines starting with '*' are expected to contain module configuration commands (rather than configuration commands for fvwm itself).

AppsBackingStore

Causes application windows to request backing store. This option compromises the ICCCM compliance of the window manager. While this option can speed things up in an X-terminal, where redraws of windows are expensive, it may not help much on regular workstations.

AutoRaise delay

Enables auto-raising of windows and specifies the time delay (in milliseconds) between when a window acquires the input focus and when it is automatically raised. This option works in focus-follows-mouse mode, and in click-to-focus mode if the focus is changed by clicking in the application window instead of a decoration window. In click-to-focus mode, you can suppress the raise-on-focus behavior by specifying a negative delay value.

BackingStore

Causes fvwm decorations to request backing store. See the discussion on Apps-BackingStore.

BoundaryWidth Width

Changes the boundary width on decorated windows to the specified value. The default is 6 pixels.

The Style command provides another (more general) method for specifying BoundaryWidth.

ButtonStyle button# WidthxHeight

Defines the rectangular decoration shape to be used in a title-bar button. button# is the title-bar button number, and is between 0 and 9. A description of title-bar button numbers is given in the Mouse section below. Width is the percentage of the full button width which is to be used. Height is the percentage of the full height to be used. Negative numbers cause the shading to be inverted.

And that's not all! If you use a line like:

ButtonStyle : 2 4 50x30@1 70x70@0 30x70@0 50x30@1

then the button 2 decoration will use a 4-point pattern consisting of a line from (x=50,y=30) to (70,70) in the shadow color (@0), and then to (30,70) in the shadow color, and finally to (50,30) in the highlight color (@1). Is that too confusing? See the sample system.fvwmrc.

CenterOnCirculate

When circulating, the desktop page containing the window which the pointer is moving to is automatically selected. If CenterOnCirculate is selected then fvwm will do its best to center the target window in the desktop viewport, rather than just lining up to the closest page.

CirculateSkip windowname

Causes windows with the indicated name to be skipped over when the circulate-up or circulate-down functions are invoked. windowname can be a window's name or its class.

The Style command provides another (more general) method for specifying CirculateSkip.

CirculateSkipIcons

Causes circulate and warp operations to skip over iconified windows.

ClickTime delay

Specifies the maximum delay (in milliseconds) between a button press and a button release for the Function built-in to consider the action a mouse click. The default delay is 150 milliseconds.

ClickToFocus

Normally keyboard input goes to the window the mouse pointer is in. If this option is set the keyboard input stays with one window until the mouse is clicked with the pointer positioned in a new window.

Cursor cursor_num cursor_type

This provides a very awkward way of changing cursor styles. Cursor_num tells which cursor you are changing, and is a number between 0 and 12, as follows:

0 POSITION - used when initially placing windows.

1 TITLE - used in a window title-bar.

2 DEFAULT - used in windows that don't set their cursor.

3 SYS - used in one of the title-bar buttons.

4 MOVE - used when moving or resizing windows.

5 WAIT - used during an EXEC builtin command.

6 MENU - used in menus.

7 SELECT - used for various builtin commands such as iconify.

8 DESTROY - used for DESTROY and DELETE built-ins.

9 TOP - used in the top side-bar of a window.

10 RIGHT - used in the right side-bar of a window.

11 BOTTOM - used in the bottom side-bar of a window.

12 LEFT - used in the left side-bar of a window.

13 TOP_LEFT - used in the top left corner of a window.

14 TOP_RIGHT - used in the top right corner of a window.

15 BOTTOM_LEFT - used in the bottom left corner of a window.

16 BOTTOM_RIGHT - used in the bottom right corner of a window.

The cursor_type argument is a number which tells the cursor shape to use. The available numbers can be found in /usr/include/X11/cursorfont.h and are currently even numbers between 0 and 152. At the current time, the following cursor types are available:

0 X_cursor 2 arrow

4 based_arrow_down 6 based_arrow_up

8 boat 10 bogosity

12 bottom_left_corner 14 bottom_right_corner

16 bottom_side 18 bottom_tee

20 box_spiral 22 center_ptr

24 circle 26 clock

28 coffee_mug 30 cross

32 cross_reverse 34 crosshair

36 diamond_cross 38 dot

40 dotbox 42 double_arrow

44 draft_large 46 draft_small

48 draped_box 50 exchange

52 fleur 54 gobbler

56 gumby 58 hand1

60 hand2 62 heart

64 icon 66 iron_cross

68 left_ptr 70 left_side

72 left_tee 74 leftbutton

76 ll_angle 78 lr_angle

80 man 82 middlebutton

84 mouse 86 pencil

88 pirate 90 plus

92 question_arrow 94 right_ptr

96 right_side 98 right_tee

100 rightbutton 102 rtl_logo

104 sailboat 106 sb_down_arrow

108 sb_h_double_arrow 110 sb_left_arrow

112 sb_right_arrow 114 sb_up_arrow

116 sb_v_double_arrow 118 shuttle

120 sizing 122 spider

124 spraycan 126 star

128 target 130 tcross

132 top_left_arrow 134 top_left_corner

136 top_right_corner 138 top_side

140 top_tee 142 trek

144 ul_angle 146 umbrella

148 ur_angle 150 watch

152 xterm

DecorateTransients

Causes transient windows, which are normally left undecorated, to be given the usual fvwm decorations. Note that some pop-up windows, such as the xterm menus, are not managed by the window manager and still do not receive decorations.

DeskTopScale Scale

Defines the virtual desktop scale with respect to the screen.

DeskTopSize HorizontalxVertical

Defines the virtual desktop size in units of the physical screen size.

DontMoveOff

Prevents windows from being moved off or initially placed off of the desktop. A few programs will not work correctly if you use this option. This only keeps windows from being completely lost off the edge of the desktop. It insists on keeping 16 pixels on the desktop but doesn't care a bit about keeping the whole window on the desk. See EdgeResistance if you don't like having windows partially off the screen.

EdgeResistance scrolling moving

Tells how hard it should be to change the desktop viewport by moving the mouse over the edge of the screen and how hard it should be to move a window over the edge of the screen.

The first parameter tells how milliseconds the pointer must spend on the screen edge before fvwm will move the viewport. This is intended for people who use "EdgeScroll 100 100" but find themselves accidentally flipping pages when they don't want to.

The second parameter tells how many pixels over the edge of the screen a window's edge must move before it actually moves partially off the screen.

Note that, with "EdgeScroll 0 0", it is still possible to move or resize windows across the edge of the current screen. By making the first parameter to EdgeResistance 10000 this type of motion is impossible. With EdgeResistance less than 10000 but greater than 0 moving over pages becomes difficult but not impossible.

EdgeScroll horizontal vertical

Specifies the percentage of a page to scroll when the cursor hits the edge of a page. If you don't want any paging or scrolling when you hit the edge of a page include "EdgeScroll 0 0" in your .fvwmrc file. If you want whole pages, use "EdgeScroll 100 100". Both horizontal and vertical should be positive numbers.

If the horizontal and vertical percentages are multiplied by 1000 then scrolling will wrap around at the edge of the desktop. If "EdgeScroll 100000 100000" is used fvwm will scroll by whole pages, wrapping around at the edge of the desktop.

Font fontname

Makes fvwm use font fontname instead of "fixed" for menus, the resize indicators, and icon labels (if Icon- Font is not specified).

Function FunctionName

Starts the definition of a complex function, composed of the fvwm built-in functions, which will later be bound to a mouse button or key. FunctionName must be enclosed in quotes. Function entries are included on lines following the Function keyword. The definition ends with the key word EndFunction. Function entries are specified as shown in the following example. The first word on each line is the built-in function which will be performed, followed the type of event which should trigger the action (enclosed in quotes), followed by any additional arguments needed by the built- in function. Menus can be specified by using the Popup built-in as long as the menu was defined earlier in the configuration file.

The trigger actions which are recognized are Immediate, Motion, Click, and DoubleClick. Immediate actions are executed as soon as the function is activated, even if a window has not been selected. If there are actions other than immediate ones, fvwm will wait to see if the user is clicking, double-clicking, or dragging the mouse. After the decision is made, fvwm will execute only the built-ins from the function definition whose trigger action matches the action performed by the user.

If the following example were bound to button 1 in a window title-bar, then, when button 1 is pressed, fvwm would wait 150 msec to see if the button is released. If the button is not released fvwm will start a move operation. When the move operation is complete a raise operation will be performed. If a button release is detected then fvwm will wait another 150 msec for a second click. If only one click is detected then the window will be raised. If two clicks are detected the window will be alternately raised and lowered. The 150 msec wait duration can be altered using the ClickTime option.

Function "Move-or-Raise"

Move "Motion"

Raise "Motion"

Raise "Click"

RaiseLower "DoubleClick"

EndFunction

The clicking and double clicking concepts do not carry through to using keyboard shortcuts.

Two special functions exist: InitFunction and RestartFunction. The InitFunction will be called when fvwm is started for the first time in any X session and can be used to start modules, set background patterns, and begin programs. The restart function will be called when fvwm is restarted. It can be used to start modules and set background patterns but probably should not be used to start programs.

HiBackColor colorname

Sets the background color of the selected window to colorname. When using a monochrome screen this option is ignored and white is used.

HiForeColor colorname

Sets the color of the selected window's title to color-name. When using a monochrome screen this option is ignored and black is used.

Icon windowname bitmap-file

Specifies the bitmap to be used for a window when it is iconified. The windowname can be an application's window name or class name and must be enclosed in quotes. The bitmap-file is either the full path name to a standard X11 bitmap file or a file in the IconPath or PixmapPath. The specified bitmap / pixmap is used in preference to any icon supplied by the window itself.

If fvwm is compiled with XPM support for color icons then bitmap can be an XPM pixmap file.

windowname should be enclosed in double quotes but bitmap-file should not. Environment variables should not be used in the bitmap-file specification.

If windowname is an empty string then the specified file is the default icon, and will be used if no other icon bitmap or pixmap can be found:

Icon "" my-favorite-icon

The Style command provides another (more general) method for specifying Icon.

IconBox left top right bottom

Defines regions of the screen in which to place icons. Up to four icon boxes can be defined. If an IconBox line is provided then icons will automatically be placed in them, if possible. Each time a window is iconified a new place is found for it. Icon boxes are searched for space going left to right, then top to bottom. Icons will not be auto-placed on top of other icons but they may be placed underneath application windows. If left or right is negative, then fvwm will add the screen width to it. If top or bottom is negative, then fvwm will add the screen height to it. NOTE: -0 is not parsed as the right or bottom pixel on the screen. You have to use -1 instead.

If no IconBox line is provided or all icon boxes are full, then fvwm will place icons near the current pointer location.

IconFont fontname

Makes fvwm use font fontname for icon labels. If omitted, the menu font (specified by the Font configuration parameter) will be used instead.

IconPath path

Specifies a colon separated list of full path names of directories where bitmap (monochrome) icons can be found. Each path should start with a slash. Note: if the M4 patches are included when fvwm is built, then m4 will want to mangle the word "include" which will frequently show up in the IconPath or PixmapPath command. To fix this add undefine('include') prior to the IconPath command.

Key keyname Context Modifiers Function

Binds a keyboard key to a specified fvwm built-in function. Definition is the same as for a mouse binding except that the mouse button number is replaced with a key name. The keyname is one of the entries from /usr/include/X11/keysymdef.h, with the leading XK_ omitted. The Context and Modifiers fields are defined as in the mouse binding.

Binding a key to a title-bar button will not cause that button to appear unless a mouse binding also exists.

Lenience

The ICCCM states that if an application sets the input field of the wm_hints structure to False, then it never wants the window manager to give it the input focus. The only application that I know of which needs this is sxpm, and that is a silly bug with a trivial fix and has no overall effect on the program anyway. Rumor is that some older applications have problems too.

If this parameter is set then fvwm will ignore this ICCCM convention.

MenuBackColor colorname

Sets the menu background color. When using monochrome this option is ignored. This option is only available if fvwm is compiled with MENUCOLOR defined.

MenuForeColor colorname

Sets the menu foreground color. When using monochrome this option is ignored. This option is only available if fvwm is compiled with MENUCOLOR defined.

MenuStippleColor colorname

Sets the color for shaded out entries in menus (for functions which are not allowed on the currently selected window). When using monochrome this option is ignored and a stipple pattern is used. This option is only available if fvwm is compiled with MENUCOLOR defined.

Module ModuleName

Specifies a module which should be spawned during initialization. At the current time the available modules are FvwmAudio, FvwmBacker, FvwmBanner, FvwmClean, FvwmDebug, FvwmIconBox, FvwmIdent, FvwmPager, FvwmSave, FvwmSaveDesk, FvwmScroll, FvwmWinList, and GoodStuff. These modules

have their own man pages. Module can also be used as a built-in. Modules can be short lived transient programs or, like GoodStuff, can remain for the duration of the X session. Modules will be terminated by the window manager prior to restarts and quits, if possible. See the introductory section on modules.

ModulePath

Specifies a colon separated list of paths for fvwm to search when looking for a module to load. Individual directories do not need trailing slashes.

Mouse Button Context Modifiers Function

Defines a mouse binding. Button is the mouse button number. If Button is zero then any button will perform the specified function. Context describes where the binding applies. Valid contexts are R for the root window, W for an application window, T for a window title bar, S for a window side, top, or bottom bar, F for a window frame (the corners), I for an Icon window, or 0 through 9 for title-bar buttons, or any combination of these letters. A is for any context except for title- bar buttons. For instance, a context of FST will apply when the mouse is anywhere in a window's border except the title-bar buttons.

Modifiers is any combination of N for no modifiers, C for control, S for shift, M for Meta, or A for any modifier. For example, a modifier of SM will apply when both the Meta and shift keys are down. X11 modifiers mod1 through mod5 are represented as the digits 1 through 5.

Function is one of fvwm's built-in functions.

The title bar buttons are numbered with odd numbered buttons on the left side of the title bar and even numbers on the right. Smaller-numbered buttons are displayed toward the outside of the window while larger-numbered buttons appear toward the middle of the window (0 is short for 10). In summary, the buttons are numbered:

1 3 5 7 9 0 8 6 4 2

The highest odd numbered button which has an action bound to it determines the number of buttons drawn on the left side of the title bar. The highest even number determines the number or right side buttons which are drawn. Actions can be bound to either mouse buttons or keyboard keys.

MWMBorders

Substitutes MWM style 1 pixel wide relief lines instead of fvwm's 2 pixel borders.

MWMButtons

Disables button press feedback for all decorations except the title bar and title-bar buttons, as in MWM.

MWMDecorHints

Causes fvwm to read the MOTIF_WM_HINTS atom from application windows and to parse and attempt to replicate the Motif behavior with regard to window decorations. Note that mwm allows function hints to affect window decorations but these effects are not replicated by this option.

MWMFunctionHints

Causes fvwm to read the MOTIF_WM_HINTS atom from application windows and to parse and attempt to replicate the Motif behavior with regard to allowed window functions. Unlike mwm, which simply removes prohibited functions from the window's menus, fvwm simply shades out the prohibited functions. Also, because fvwm implements some functions in user defined macros that mwm implements internally, the mapping of prohibited functions is partially based on the menu item label.

MWMHintOverride

If MWMFunctionHints is used then maximization and iconfication are prohibited for transients. Also, windows can specify that the window manager should not destroy or delete them. Since these MWM rules are kind of stupid, especially with regard to the transient windows, I provide this MWMHintOverride option. When it is used menu items will be shaded out if MWM would prohibit their use, but the user can go ahead and select that item and it will operate as expected.

The override should be used cautiously because some applications will break if you override their mwm hints.

MWMMenus

Substitutes MWM look and feel menus in place of the standard fvwm versions. This option also triggers a few other mwm-style options, such as centering the size / resize window on the screen, instead of leaving it in the upper left, and switches the resize-on-initial-placement trigger action to shift-button-1 instead of the twm style press-button-2

NoBorder windowname

Keeps fvwm from putting decorative borders on windows named windowname. This command has no effect on the title-bar. This is handy for clocks and similar gadgets that you don't want to take up too much space. windowname can be a window's name or its class.

If you specify both NoBorder windowname and NoTitle windowname for the same window in your .fvwmrc file the window will be completely undecorated.

Windowname can contain the wildcards "*" and "?" which match window names in the normal Unix filename matching manner. Actual "*", "?", and "\" characters in a window name can be entered by preceding the character with a "\".

The Style command provides another (more general) method for specifying NoBorder.

NoBoundaryWidth Width

Changes the width of the decorations for windows with no titles and no borders. The default is 1. Any positive or zero value is acceptable. Decorations for these undecorated windows have the same context as the side-bars on normally decorated windows.

The Style command provides another (more general) method for specifying NoBoundaryWidth.

NoPPosition

Instructs fvwm to ignore the PPosition field when adding new windows. Adherence to the PPosition field is required for some applications, but if you don't have one of those its a real headache.

NoTitle windowname

Keeps fvwm from putting a title-bar in the decorations for windows named windowname. This is handy for clocks and similar gadgets that you don't want to take up too much space. windowname can be a window's name or its class.

Windowname can contain the wildcards "*" and "?" which match window names in the normal Unix filename matching manner. Actual "*", "?", and "\" characters in a window name can be entered by preceding the character with a "\".

The Style command provides another (more general) method for specifying NoTitle.

OpaqueMove percentage

Tells fvwm the maximum size window with which opaque window movement should be used. The percentage is percent of the total screen area. With "OpaqueMove 0" all windows will be moved using the traditional rubber-band outline. With "OpaqueMove 100" all windows will be move as solid windows. The default is "OpaqueMove 5", which allows small windows to be moved in an opaque manner but large windows are moved as rubber-bands.

OpaqueResize

Causes resize operations to be done with the window itself instead of an outline.

Pager X_Location Y_Location

Enables a paging style of moving across the desktop. A Pager window (not a popup) will appear at (X_Location, Y_Location). Miniature versions of all the nonsticky windows on the virtual desktop are shown in the pager. The color of the miniature version is the same as the color of the full-size window's border.

In the Pager window, pressing mouse button 1 will move the desktop viewport to the selected page (in click-to-focus mode; it will also move the keyboard focus to the window whose miniature you click on). Pressing button 2 on a window in the pager will begin a window move, using the miniature to quickly move the window anywhere on the desktop. Pressing button 3 will move the top-left corner of the viewport to the location of the button press, even if it does not line up with a page. Dragging button 3 will cause the selected viewport to scroll as you move the pointer. The Pager is automatically sticky but does not automatically stay on top.

PagerForeColor colorname

Causes the pager foreground color to be colorname instead of black. This is the color used to highlight the current viewport in the pager window. On a mono-

chrome screen this option is ignored. If the NO_PAGER option is set when building fvwm this option is unavailable.

PagerBackColor colorname

Causes the pager background color to be colorname instead of white. On a monochrome screen this option is ignored. If the NO_PAGER option is set when building fvwm this option is unavailable.

PagerFont fontname

Makes fvwm use font fontname for writing window icon names in the window's representation in the pager. If this option is omitted no names are written in the pager windows.

PagingDefault pagingdefaultvalue

Tells fvwm if it should start up with paging enabled or disabled. "PagingDefault 0" will start fvwm with paging disabled; "PagingDefault 1" will start fvwm with paging enabled by default.

PixmapPath path

Specifies a colon separated list of full path names of directories where pixmap (color) icons can be found. Each path should start with a slash.

Popup PopupName

Starts the definition of a pop-up menu which will later be bound to a mouse button or key. PopupName must be enclosed in quotes. Menu entries are included on lines following the Popup keyword. The menu definition ends with the key word EndPopup. Menu entries are specified as shown in the following example. The first word on each line is the built-in function which will be performed, followed by the caption (enclosed in quotes) which will be shown in the menu, followed by any additional arguments needed by the built-in function. Sub- menus can be specified by using the Popup built-in as long as the sub-menu was defined earlier in the configuration file.

Popup "Window Ops"

```
Title "Window Ops"
Move "Move"
Resize "Resize"
Raise "Raise"
Lower "Lower"
Iconify "(De)Iconify"
Nop " "
Destroy "Destroy"
Title "HARDCOPY"
Exec "Hardcopy" exec xdpr &
Exec "Hardcopy RV" exec xdpr -rv &
```

EndMenu

Note that if a tab character is embedded in the caption of a menu entry then the text following the tab will be entered into a second column in the menu and the entire menu will be left-adjusted. This is intended for shortcut labeling. The tab character must really be a tab. If it is expanded into spaces it will not work!

For example:

Popup "Window Ops"

Title "Window Ops Alt-F1"

.

.

.

Is the start of a left adjusted menu. Alt-F1 will be placed toward the right side of the menu.

Shortcut keys may be specified in the menu definition by preceding the character with an ampersand. The ampersand will not be displayed but the character after it will be displayed underlined, and if the user presses the corresponding key then that item will be activated as if the user had clicked on it with the mouse. Only alphabetic and numeric characters may be used as shortcut keys. The shift state of the keyboard is ignored when testing shortcut characters. For example:

Popup "Window Ops"

Maximize "Ma&ximise" 100 100

EndMenu

When this menu is popped up the 'x' will be underlined and pressing the 'x' key will cause the current window to be maximized. Shortcut keys are not operative unless MENU_HOTKEYS was defined when building fvwm. If WINDOWLIST_HOTKEYS was also defined then hot keys are automatically added to the WindowList when it is displayed.

RandomPlacement

Causes windows which would normally require user placement to be automatically placed in ever-so-slightly random locations. For the best of all possible worlds use both RandomPlacement and SmartPlacement.

SaveUnders

Causes the fvwm decoration frames to request save-unders. This can significantly improve the performance during opaque moves but it causes a significant increase in memory usage.

SloppyFocus

This focusing mode is like focus-follows-mouse (the default) except that the focus will not be removed from a window until your mouse enters a new window. Exiting a window to enter the root window will leave the focus unchanged.

SmartPlacement

Causes windows which would normally require user placement to be automatically placed in a smart location - a location in which they do not overlap any other windows on the screen. If no such position can be found user placement or random placement will be used as a fall-back method. For the best of all possible worlds use both RandomPlacement and SmartPlacement.

StartsOnDesk windowname desk-number

This command causes windows whose name or class is windowname to be initially placed on desktop number desk-number. windowname should be enclosed in double quotes. If the window requires interactive placement, an outline will be displayed on the current desk but the window will appear on the specified desk.

Windowname can contain the wildcards "*" and "?" which match window names in the normal Unix filename matching manner. Actual "*", "?", and "\" characters in a window name can be entered by preceding the character with a "\".

The Style command provides another (more general) method for specifying StartsOnDesk.

StaysOnTop windowname

These windows always try to stay on top of the other windows. This might be handy for clocks or mailboxes that you would always like to be visible. If the window is explicitly lowered it will not try to force its way back to the top until it is explicitly raised. windowname can be a window's name or its class.

Windowname can contain the wildcards "*" and "?" which match window names in the normal Unix filename matching manner. Actual "*", "?", and "\" characters in a window name can be entered by preceding the character with a "\".

The Style command provides another (more general) method for specifying StaysOnTop.

StdBackColor colorname

Sets the background color for menus and non-selected windows to colorname. When using a monochrome screen this option is ignored and white is used.

The Style command provides another (more general) method for specifying StdBackColor.

StdForeColor colorname

Sets the foreground color for menus and non-selected window titles to colorname. When using a monochrome screen this option is ignored and black is used.

The Style command provides another (more general) method for specifying StdForeColor.

StickyBackColor colorname

Sets the background color for non-selected sticky windows to colorname. When using a monochrome screen this option is ignored and white is used. Only available if -DMORE_COLORS is used when compiling.

StickyForeColor colorname

Sets the foreground color for non-selected sticky window titles to colorname. When using a monochrome screen this option is ignored and black is used. Only available if -DMORE_COLORS is used when compiling.

Sticky windowname

Sticky windows "stick to the screen's glass." That is, they don't move the the viewport into the virtual desk- top changes. windowname can be a window's name or its class.

Windowname can contain the wildcards "*" and "?" which match window names in the normal Unix filename matching manner. Actual "*", "?", and "\" characters in a window name can be entered by preceding the character with a "\".

The Style command provides another (more general) method for specifying Sticky.

StickyIcons

Causes icons to always stick to the screen's glass. That is, icons always follow you around the desktop. When a window is de-iconified it gets un-stuck. Some people find this a useful way of moving windows around.

StubbornIcons

Changes de-iconification behavior a bit. Instead of having windows always de-iconify themselves on the current page they de-iconify into their original position.

StubbornIconPlacement

When used with IconBoxes, causes icons to avoid placing themselves underneath existing windows.

StubbornPlacement

When using SmartPlacement, causes new windows to avoid placing themselves over icons.

Style windowname options

This command is intended to replace the commands NoBorder, NoTitle, StartsOnDesk, Sticky, StaysOnTop, Icon, WindowListSkip, CirculateSkip, SuppressIcons, BoundaryWidth, NoBoundaryWidth, StdForeColor, and StdBackColor with a single flexible and comprehensive command. This command is used to set attributes of a window to values other than the default or to set the window manager default styles.

windowname can be a window's name, class, or resource string. It can contain the wildcards * and / or ?, which are matched in the usual Unix filename manner.

options is a comma separated list containing some or all of the keywords BorderWidth, HandleWidth,NoIcon / Icon, NoTitle / Title, NoHandles / Handles, WindowListSkip / WindowListHit, CirculateSkip / CirculateHit, StaysOnTop /

StaysPut, Sticky / Slippery, StartIconic / StartNormal, Color, ForeColor, BackColor, StartsOnDesk / StartsAnyWhere, IconTitle / NoIconTitle, and NoButton / Button.

In the above list some options are listed as style- option / opposite-style-option. The opposite-style-option for entries that have them describes the fvwm default behavior and can be used if you want to change the fvwm default behavior.

Icon takes an (optional) unquoted string argument which is the icon bitmap or pixmap to use.

StartsOnDesk takes a numeric argument which is the desktop number on which the window should be initially placed.

BorderWidth takes a numeric argument which is the width of the border to place the window if it does not have resize-handles.

HandleWidth takes a numeric argument which is the width of the border to place the window if it does have resize-handles.

Button and NoButton take a numeric argument which is the number of the title-bar button which is to be included / omitted.

Color takes two arguments. The first is the window- label text color and the second is the window decoration's normal background color. The two colors are separated with a slash. If the use of a slash causes problems then the seperate ForeColor and BackColor options can be used.

An example:

```
# Change default fvwm behavior to no title-bars on windows!
# Also define a default icon.
Style "*" NoTitle,Icon unknown1.xpm, BorderWidth 4,HandleWidth 5
# now, window specific changes:
Style "Fvwm*" NoHandles,Sticky,WindowListSkip,BorderWidth 0
Style "Fvwm Pager" StaysOnTop, BorderWidth 0
Style "*lock" NoHandles,Sticky,StaysOnTop,WindowListSkip
Style "xbiff" Sticky, WindowListSkip
Style "GoodStuff" NoHandles,Sticky,WindowListSkip
Style "sxpm" NoHandles
Style "makerkit"
# Put title-bars back on xterms only!
Style "xterm" Title, Color black / grey
Style "rxvt" Icon term.xpm
Style "xterm" Icon rterm.xpm
Style "xcalc" Icon xcalc.xpm
Style "xbiff" Icon mail1.xpm
Style "xmh" Icon mail1.xpm, StartsOnDesk 2
```

Style "xman" Icon xman.xpm

Style "matlab" Icon math4.xpm, StartsOnDesk 3

Style "xmag" Icon magnifying_glass2.xpm

Style "xgraph" Icon graphs.xpm

Style "GoodStuff" Icon toolbox.xpm

Style "Maker" StartsOnDesk 1

Style "signal" StartsOnDesk 3

Note that all properties for a window will be OR'ed together. In the above example "FvwmPager" gets the property StaysOnTop via an exact window name match but also gets NoHandles, Sticky, and WindowListSkip by a match to "Fvwm*". It will get NoTitle by virtue of a match to "*". If conflicting styles are specified for a window, then the last style specified will be used.

If the NoIcon attribute is set then the specified window will simply disappear when it is iconified. The window can be recovered through the window-list. If Icon is set without an argument then the NoIcon attrbute is cleared but no icon is specified. An example which allows only the FvwmPager module icon to exist:

Style "*" NoIcon

Style "Fvwm Pager" Icon

SuppressIcons

Prevents icon windows from being created or drawn. When used with the window-list this provides a sort of icon manager.

The Style command provides another (more general) method for specifying SuppressIcons.

WindowFont fontname

Makes fvwm use font fontname instead of "fixed" for the window title bar.

WindowListSkip windowname

Causes windows with the indicated name to be left out of the window list.

Windowname can contain the wildcards "*" and "?" which match window names in the normal Unix filename matching manner. Actual "*", "?", and "\" characters in a window name can be entered by preceding the character with a "\".

The Style command provides another (more general) method for specifying WindowListSkip.

XORvalue number

Changes the value with which bits are XOR'ed when doing rubber-band window moving or resizing. Setting this value is a trial-and-error process.

BUILT IN FUNCTIONS

Fvwm supports a set of built-in functions which can be bound to keyboard or mouse buttons:

Beep

Makes the computer beep.

CirculateDown [name window_name]

Causes the pointer to move to the next window in the list of windows for which CirculateSkip has not not been specified.

If the optional arguments are supplied then the focus will move to the first window whose name (or icon name or class) matches window_name. The optional argument name is required if window_name is supplied and is enclosed in quotes. This argument is the name which appears in menus if the function is called from a menu, but serves no purpose if the function is not called from a menu.

CirculateUp [name window_name]

Causes the pointer to move to the previous window in the list of windows for which CirculateSkip has not not been specified.

If the optional arguments are supplied then the focus will move to the first window whose name (or icon name or class) matches window_name. The optional argument name is required if window_name is supplied and is enclosed in quotes. This argument is the name which appears in menus if the function is called from a menu, but serves no purpose if the function is not called from a menu

Here's an example that move the focus to an xterm window when Alt-F1 is pressed:

```
Key F1 A M CirculateUp "whatever" xterm
```

Close

If the window accepts the delete window protocol a message is sent to the window asking it to gracefully remove itself. If the window does not understand the delete window protocol then the window is destroyed.

CursorMove horizonal vertical

Moves the mouse pointer by horizontal pages in the X direction and vertical pages in the Y direction. Either or both entries may be negative. Both horizontal and vertical values are expressed in percent of pages, so "CursorMove 100 100" means to move down and left by one full page. "CursorMove 50 25" means to move left half a page and down a quarter of a page. The CursorMove function should not be called from pop-up menus.

Delete

Sends a message to a window asking that it remove itself, frequently causing the application to exit.

Desk arg1 arg2

Changes to another desktop (workspace, room).

If arg1 is non zero then the next desktop number will be the current desktop number plus arg1. Desktop numbers can be negative.

If arg1 is zero then the new desktop number will be arg2.

The number of active desktops is determined dynamically. Only desktops which contain windows or are currently being displayed are active. Desktop numbers must be between 2147483647 and -2147483648 (is that enough?).

Destroy

Destroys a window. Guaranteed to get rid of the window, but is a fairly violent way to terminate an application.

Exec name command

Executes command. command is not quoted but name is. name is the name that appears in a menu, if that is where the function is called from. name is required even if the function is not called from a menu.

The following example binds function key F1 in the root window, with no modifiers, to the exec function. The program rxvt will be started with an assortment of options.

```
Key F1 R N Exec "rxvt" exec rxvt -fg yellow -bg blue -e /bin/tcsh &
```

Focus

Moves the viewport or window as needed to make the selected window visible. Sets the keyboard focus to the selected window. Raises the window if needed to make it visible. Warps the pointer into the selected window in focus-follows-mouse mode. Does not de-iconify. This function is primarily for use with a module such as FvwmWinList.

Function

Used to bind a previously defined function to a key or mouse button.

The following example binds mouse button 1 to a function called "Move-or-Raise", whose definition was provided as an example earlier in this man page. After performing this binding fvwm will execute to move-or- raise function whenever button 1 is pressed in a window title-bar.

```
Mouse 1 T A Function "Move-or-Raise"
```

GotoPage x y

Moves the desktop viewport to page (x,y). The upper left page is (0,0), the upper right is (N,0), where N is one less than the current number of horizontal pages specified in the DeskTopSize command. The lower left page is (0,M), and the lower right page is (N,M), where M is the desktop's vertical size as specified in the DeskTopSize command. The GotoPage function should not be used in a pop-up menu.

Iconify [value]

Iconifies a window if it is not already iconified or de-iconifies it if it is already iconified. If the optional argument value is positive the only iconification will be

allowed. It the optional argument is negative only de-iconification will be allowed.

Lower

Allows the user to lower a window.

Maximize [horizontal vertical]

Without its optional arguments Maximize causes the window to alternately switch from a full-screen size to its normal size.

With the optional arguments horizontal and vertical, which are expressed as percentage of a full screen, the user can control the new size of the window. If horizontal is greater than 0 then the horizontal dimension of the window will be set to horizontal*screen_width / 100. The vertical resizing is similar. For example, the following will add a title-bar button to switch a window to the full vertical size of the screen:

Mouse 0 4 A Maximize 0 100

The following causes windows to be stretched to the full width:

Mouse 0 4 A Maximize 100 0

This makes a window that is half the screen size in each direction:

Mouse 0 4 A Maximize 50 50

Values larger than 100 can be used with caution.

If the letter "p" is appended to each coordinate (horizontal and / or vertical), then the scroll amount will be measured in pixels.

Module name ModuleName

Specifies a module which should be spawned. Modules can be short lived transient programs or can remain for the duration of the X session. Modules will be terminated by the window manager prior to restarts and quits, if possible. name is a double-qouted string which has absolutely no significance, but must exist.

Move [x y]

Allows the user to move a window. If called from some- where in a window or its border, then that window will be moved. If called from the root window then the user will be allowed to select the target window.

If the optional arguments x and y are provided, then the window will be moved so that its upper left corner is at location (x,y). The units of x and y are percent-of-screen, unless a letter "p" is appended to each coordinate, in which case the location is specified in pixels.

Examples:

Mouse 1 T A Move

Mouse 2 T A Move 10 10

Mouse 3 T A Move 10p 10p

In the first example, an interactive move is indicated. In the second, the window whose title-bar is selected will be moved so that its upper left hand corner is 10 percent of the screen width in from the left of the screen, and 10 percent down from the top. The final example moves the window to coordinate (10,10) pixels.

Nop Does nothing. This is used to insert a blank line or separator in a menu. If the menu item specification is Nop " ", then a blank line is inserted. If it looks like Nop "", then a separator line is inserted.

Popup

This built-in has two purposes: to bind a menu to a key or mouse button, and to bind a sub-menu into a menu. The formats for the two purposes differ slightly.

To bind a previously defined pop-up menu to a key or mouse button:

The following example binds mouse buttons 2 and 3 to a pop-up called "Window Ops", whose definition was provided as an example earlier in this man page. The menu will pop up if the buttons 2 or 3 are pressed in the window frame, side-bar, or title-bar, with no modifiers (none of shift, control, or meta).

```
Mouse 2 FST N Popup "Window Ops"
Mouse 3 FST N Popup "Window Ops"
```

Pop-ups can be bound to keys through the use of the key modifier. Pop-ups can be operated without using the mouse by binding to keys and operating via the up arrow, down arrow, and enter keys.

To bind a previously defined pop-up menu to another menu, for use as a sub-menu:

The following example defines a sub menu, "QuitVerify" and binds it into a main menu, called "Utilities":

```
Popup "Quit-Verify"
Title "Really Quit Fvwm?"
Quit "Yes, Really Quit"
Restart "Restart Fvwm" fvwm
Nop ""
Nop "No, Don't Quit"
EndPopup

Popup "Utilities"
Title "Utilities"
Exec "Xterm" exec xterm &
Exec "Rxvt" exec rxvt &
Exec "Top" exec rxvt -T Top -n Top -e top &
Exec "Calculator" exec xcalc &
```

```
Exec "Xman" exec xman &
Exec "Xmag" exec xmag &
Nop ""
Popup "Exit Fvwm" Quit-Verify
EndPopup
```

Sub-menus must be defined prior to the main menu in which they are bound. Sub-menu nesting can be arbitrarily deep.

Quit

Exits fvwm, generally causing X to exit too.

Raise

Allows the user to raise a window.

RaiseLower

Alternately raises and lowers a window.

Refresh

Causes all windows on the screen to redraw themselves.

Resize [x y]

Allows the user to resize a window.

If the optional arguments x and y are provided, then the window will be moved so that its upper left corner is at location (x,y). The units of x and y are percent-of-screen, unless a letter "p" is appended to each coordinate, in which case the location is specified in pixels.

Restart name WindowManagerName

Causes fvwm to restart itself if WindowManagerName is "fvwm", or to switch to an alternate window manager if WindowManagerName is other than "fvwm". If the window manager is not in your default search path, then you should use the full path name for WindowManagerName.

WindowManagerName is not quoted but name is. name is the name that appears in a menu, if that is where the function is called from. name is required even if the function is not called from a menu.

This command should not have a trailing ampersand or any command line arguments and should not make use of any environmental variables. Of the following examples, the first three are sure losers, but the third is OK:

```
Key F1 R N Restart " " fvwm &
Key F1 R N Restart " " $(HOME)/bin/fvwm
Key F1 R N Restart " " twm -f .mystartupfile
Key F1 R N Restart " " /home/nation/bin/fvwm
```

Stick

Makes a window sticky if it is not already sticky, or non-sticky if it is already sticky.

Scroll horizonal vertical

Scrolls the virtual desktop's viewport by horizontal pages in the x-direction and vertical pages in the y- direction. Either or both entries may be negative. Both horizontal and vertical values are expressed in percent of pages, so "Scroll 100 100" means to scroll down and left by one full page. "Scroll 50 25" means to scroll left half a page and down a quarter of a page. The scroll function should not be called from pop-up menus. Normally, scrolling stops at the edge of the desktop.

If the horizontal and vertical percentages are multiplied by 1000 then scrolling will wrap around at the edge of the desktop. If "Scroll 100000 0" is executed over and over fvwm will move to the next desktop page on each execution and will wrap around at the edge of the desktop, so that every page is hit in turn.

If the letter "p" is appended to each coordinate (horizontal and / or vertical), then the scroll amount will be measured in pixels.

Title

Does nothing. This is used to insert a title line in a popup or menu.

TogglePage

Temporarily disables edge scrolling. Edge scrolling can be re-enabled by calling this again.

Wait name

This built-in is intended to be used in fvwm functions only. It causes execution of a function to pause until a new window name name appears. Fvwm remains fully functional during a wait. This is particularly useful in the InitFunction if you are trying to start windows on specific desktops:

```
Function "InitFunction"
Exec "I" exec xterm -geometry 80x64+0+0
Wait "I" xterm
Desk "I" 0 2
Exec "I" exec xmh -font fixed -geometry 507x750+0+0 &
Wait "I" xmh
Desk "I" 0 0
EndFunction
```

The above function starts an xterm on the current desk, waits for it to map itself, then switches to desk 2 and starts an xmh. After the xmh window appears control moves to desk 0.

Warp [name window_name]

Same as CirculateDown but de-iconifies any iconified windows as it focuses on them.

WindowsDesk new_desk

Moves the selected window the the desktop specified as new_desk.

WindowList arg1 arg2

Generates a pop-up menu (and pops it up) in which the title and geometry of each of the windows currently on the desk top are shown. The geometry of iconified windows is shown in brackets. Selecting an item from the window list pop-up menu will cause that window to be moved onto the desktop if it is currently not on it, will move the desktop viewport to the page containing the upper left hand corner of the window, will de-iconify the window if it is iconified, and will raise the window.

If arg1 is an even number then the windows will be listed using the window name (the name that shows up in the title-bar). If it is odd then the window's icon name is used.

If arg1 is less than 2 then all windows on all desktops (except those listed in WindowListSkip directives) will be shown.

If arg1 is 2 or 3 then only windows on the current desktop will be shown.

If arg1 is 4 or 5 then only windows on desktop number arg2 will be shown.

KEYBOARD SHORTCUTS

All (I think) window manager operations can be performed from the keyboard so mouseless operation should be possible. In addition to scrolling around the virtual desktop by binding the Scroll built-in to appropriate keys, pop-ups, move, resize, and most other built-ins can be bound to keys. Once a built-in function is started the pointer is moved by using the up, down, left, and right arrows, and the action is terminated by pressing return. Holding down the shift key will cause the pointer movement to go in larger steps and holding down the control key will cause the cursor movement to go in smaller steps. Standard emacs and vi cursor movement controls (^n, ^p, ^f, ^b, and ^j, ^k, ^h, ^l) can be used instead of the arrow keys.

SUPPLIED CONFIGURATION

A sample configuration file, system.fvwmrc, is supplied with the fvwm distribution. It is well commented and can be used as a source of examples for fvwm configuration.

USE ON MULTI-SCREEN DISPLAYS

If the -s command line argument is not given, fvwm will automatically start up on every screen on the specified display. After fvwm starts each screen is treated indepen dently. Restarts of fvwm need to be performed separately on each screen. The use of EdgeScroll 0 0 is strongly recommended for multi-screen displays.

You may need to quit on each screen to quit from the X session completely.

Multi-screen support is only available if fvwm is compiled with -DMULTIPLE_SCREENS.

BUGS

As of fvwm 0.99 there were exactly 39.342 unidentified bugs. Identified bugs have mostly been fixed, though. Since then 9.34 bugs have been fixed. Assuming that there are at least 10 unidentified bugs for every identified one, that leaves us with 39.342 - 9.32 + 10 * 9.34 = 123.402 unidentified bugs. If we follow this to its logical conclusion we will have an infinite number of unidentified bugs before the number of bugs can start to diminish, at which point the program will be bug-free. Since this is a computer program infinity = 3.4028e+38 if you don't insist on double- precision. At the current rate of bug discovery we should expect to achieve this point in 3.37e+27 years. I guess I better plan on passing this thing on to my children....

Binding a key to a window decoration but not to the window itself is discouraged because when the key-press event finally gets to the window it will be marked as SYNTHETIC and will be ignored by many applications.

Bug reports can be sent to fvwm@wonderland.org.

AUTHOR

Robert Nation with help from many people, based on twm code, which was written by Thomas LaStrange.

2.8.2 FvwmAudio Manual Page

NAME

FvwmAudio - the FVWM Audio module

SYNOPSIS

FvwmAudio is spawned by fvwm, so no command line invocation will work. From within the .fvwmrc file, FvwmAudio is spawned as follows:

Module FvwmAudio

or from within an fvwm pop-up menu:

```
Popup "Module-Popup"
Title "Modules"
Module "GoodStuff" GoodStuff
Module "NoClutter" FvwmClean
Module "FvwmIdentify" FvwmIdent
Module "Banner" FvwmBanner
Module "Debug" FvwmDebug
Module "Audio" FvwmAudio
Module "Pager" FvwmPager 0 3
EndPopup
```

DESCRIPTION

The FvwmAudio module communicates with the Fvwm window manager to bind audio sounds to window manager actions. Different audio queues may be assigned to any window manager action. FvwmAudio can be used with any independent audio player, and therefore requires no special audio library APIs of its own. Simple in its design, it merely binds the audio filenames to particular actions and forks off the audio player program with the appropriate filename when that action occurs.

FvwmAudio can also have builtin support for the rplay library.

CONFIGURATION OPTIONS

FvwmAudio reads the same .fvwmrc file as fvwm(1) reads when it starts up, and looks for certain configuration options:

***FvwmAudioPlayCmd /usr/bin/X11/demos/auplay**

This determines the independent audio player program that will actually play the sounds. If the play command is set to builtin-rplay then the builtin rplay support will be used. For example: *FvwmAudioPlayCmd builtin-rplay

***FvwmAudioDir /usr/lib/sounds**

Specifies the directory to look for the audio files. This option is ignored when rplay is used.

***FvwmAudioDelay 5**

Specifies that sound events will only be played if they occur at least 5 seconds after the previous event. Sounds events that occur during the delay period are ignored. This option is useful if you don't want several sounds playing at the same time. The default delay is 0 which disables the audio delay.

***FvwmAudioRplayHost hostname**

Specifies what host the sounds will play on. The host- name can also be an environment variable such as $HOST-DISPLAY. This option is only valid with builtin rplay support.

***FvwmAudioRplayPriority 0**

Specifies what priority will be assigned to the sounds when they are played. This option is only valid with builtin rplay support.

***FvwmAudioRplayVolume 127**

Specifies what volume will be assigned to the sounds when they are played. This option is only valid with builtin rplay support.

***FvwmAudio window-manager_action audio_filename**

Binds particular window manager actions to sound queues.

***FvwmAudio startup TaDa.au**

***FvwmAudio shutdown Elvis_Left.au**

***FvwmAudio unknown doh.au**

***FvwmAudio add_window drip.au**

***FvwmAudio raise_window swoosh.au**

***FvwmAudio lower_window swoosh.au**

***FvwmAudio focus_change boing.au**

***FvwmAudio destroy_window explosion.au**

***FvwmAudio iconify ploop.au**

***FvwmAudio deiconify ploop.au**

***FvwmAudio toggle_paging fwop.au**

***FvwmAudio new_page beam_trek.au**

***FvwmAudio new_desk beam_trek.au**

***FvwmAudio configure_window huh.au**

***FvwmAudio window_name beep.au**

***FvwmAudio icon_name beep.au**

***FvwmAudio res_class beep.au**

***FvwmAudio res_name beep.au**

***FvwmAudio end_windowlist twang.au**

INVOCATION

The invocation method was shown in the synopsis section. No command line invocation is possible. FvwmAudio must be invoked by the fvwm window manager.

BUGS

It's REALLY noisy when fvwm starts and restarts.

COPYRIGHTS

This module is heavily based on a similar Fvwm module called FvwmSound by Mark Boyns. FvwmAudio simply takes Mark's original program and extends it to make it generic enough to work with any audio player. The concept for interfacing this module to the Window Manager, is original work by Robert Nation.

Copyright 1994, Mark Boyns and Mark Scott. No guarantees or warranties or anything are provided or implied in any way whatsoever. Use this program at your own risk. Permission to use this program for any purpose is given, as long as the copyright is kept intact.

AUTHORS

Mark Boyns (boyns@sdsu.edu)

Mark Scott (mscott@mcd.mot.com)

2.8.3 FvwmAuto Manual Page

NAME

FvwmAuto - the FVWM auto-raise module

SYNOPSIS

FvwmAuto is spawned by fvwm, so no command line invocation will work. An appropriate invocation from fvwm looks like:

Module FvwmAuto 200 or, from a menu: Popup "Module-Popup"

Title "Modules"

Module "GoodStuff" GoodStuff

Module "NoClutter" FvwmClean

Module "FvwmIdentify" FvwmIdent

Module "Banner" FvwmBanner

Module "Debug" FvwmDebug

Module "SaveDesktop" FvwmSave

Module "AutoRaise" FvwmAuto 200 EndPopup

The numeric argument is required. It specified how long a window must retain the keyboard input focus before it is raised. The delay is measured in milliseconds, and any integer 0 or greater is acceptable.

AUTHOR

FvwmAuto just appeared one day, nobody knows how.

2.8.4 FvwmBacker Manual Page

NAME

FvwmBacker - the FVWM background changer module

SYNOPSIS

FvwmBacker is spawned by fvwm, so no command line invocation will work.

DESCRIPTION

The FvwmBacker module provides functionality to change the background when changing desktops. Any command can be executed to change the backgrounds. The commands used to change the backgrounds must exit after changing the root window, (i.e. 'xv -root zzz.gif' does not, without the -quit option) otherwise FvwmBacker will hang.

COPYRIGHTS

The FvwmBacker module is the original work of Mike Finger.

Copyright 1994, Mike Finger. The author makes no guarantees or warranties of any kind about the use of this module. Use this modules at your own risk. You may freely use this module or any portion of it for any purpose as long as the copyright is kept intact.

INITIALIZATION

During initialization, FvwmBacker will scan the same configuration file that FVWM used during startup to find the options that pertain to it. These options are discussed in a later section.

INVOCATION

FvwmBacker can be invoked by fvwm during initialization by inserting the line 'Module FvwmBacker' in the .fvwmrc file.

FvwmBacker must reside in a directory that is listed in the ModulePath option of FVWM for it to be executed by FVWM.

CONFIGURATION OPTIONS

The follow option is the only option supported at present

***FvwmWinListDesk DeskNumber command**

Specifies the command to execute when the specified DeskNumber becomes active.

If the command begins with -solid FvwmBacker uses the next argument as a color in the X database and sets the background to that color without generating a system call to xsetroot. Only single word color names may be used.

SAMPLE CONFIGURATION

The following are excepts from a .fvwmrc file which describe FvwmBacker initialization commands:

```
####
# Set Up Backgrounds for different desktops.
####
*FvwmBackerDesk 0 -solid steelblue
*FvwmBackerDesk 1 -solid midnightblue
*FvwmBackerDesk 2 -solid yellow
*FvwmBackerDesk 3 xpmroot /usr/include/X11/pixmaps/
background2.xpm
```

AUTHOR

Mike Finger (mfinger@mermaid.micro.umn.edu) (Mike_Finger@atk.com) (doodman on IRC, check the #linux channel)

Modified by Andrew Davison (davison@cs.monash.edu.au)

2.8.5 FvwmBanner Manual Page

NAME

FvwmBanner - the FVWM Banner

SYNOPSIS

FvwmBanner is intended to be spawned by fvwm, but it will work if run from the command line.

DESCRIPTION

The FvwmInitBanner displays an Fvwm Logo in the center of the screen for 5 seconds.

COPYRIGHTS

None.

INITIALIZATION

Nothing interesting.

INVOCATION

FvwmBanner can be invoked by binding the action 'Module FvwmBanner' to a menu or key-stroke in the .fvwmrc file. Fvwm will search directory specified in the ModulePath configuration option to attempt to locate FvwmBanner. Although nothing keeps you from launching FvwmBanner at start-up time, you probably don't want to.

CONFIGURATION OPTIONS

There are no configuration options.

AUTHOR

Robert Nation

2.8.6 FvwmClean Manual Page

NAME

FvwmClean - the FVWM desktop clutter reduction module

SYNOPSIS

FvwmClean is spawned by fvwm, so no command line invocation will work.

DESCRIPTION

The FvwmClean module performs actions on windows which have not had the keyboard focus for a specific length of time. This is intended to help alleviate the problem of leaving programs running in unused portions of your desktop.

FvwmClean reads the same .fvwmrc file as fvwm reads when it starts up, and looks for lines similar to "*FvwmNoClutter 3600 Iconify".

COPYRIGHTS

The FvwmClean program, and the concept for interfacing this module to the Window Manager, are all original work by Robert Nation

Copyright 1994, Robert Nation. No guarantees or warranties or anything are provided or implied in any way whatsoever. Use this program at your own risk. Permission to use this program for any purpose is given, as long as the copyright is kept intact.

INITIALIZATION

During initialization, FvwmClean will eventually search a configuration file which describes the time-outs and actions to take. The configuration file is the same file that fvwm used during initialization.

If the FvwmClean executable is linked to another name, ie ln -s FvwmClean OtherClutter, then another module called Other-Clutter can be started, with a com-

pletely different configuration than FvwmClean, simply by changing the keyword FvwmClean to OtherClutter. This way multiple clutter- reduction programs can be used.

INVOCATION

FvwmClean can be invoked by inserting the line 'Module FvwmClean' in the .fvwmrc file. This can be placed on a line by itself, if FvwmClean is to be spawned during fvwm's initialization, or can be bound to a menu or mouse button or keystroke to invoke it later. Fvwm will search directory specified in the ModulePath configuration option to attempt to locate FvwmClean.

CONFIGURATION OPTIONS

FvwmClean reads the same .fvwmrc file as fvwm reads when it starts up, and looks for lines similar to "*FvwmClean 3600 Iconify". The format of these lines is "*FvwmClean [time] [command]", where command is any fvwm built-in command, and time is the time in seconds between when a window looses focus and when the command is executed. At most 3 actions can be specified.

AUTHOR

Robert Nation

2.8.7 FvwmDebug Manual Page

NAME

FvwmDebug - the FVWM debugger module

SYNOPSIS

FvwmDebug is spawned by fvwm, so no command line invocation will work.

DESCRIPTION

The FvwmDebug captures the stream of module packets coming from fvwm, and displays them on stderr.

COPYRIGHTS

The FvwmDebug program, and the concept for interfacing this module to the Window Manager, are all original work by Robert Nation

Copyright 1994, Robert Nation. No guarantees or warranties or anything are provided or implied in any way whatsoever. Use this program at your own risk. Permission to use this program for any purpose is given, as long as the copyright is kept intact.

INITIALIZATION

During initialization, the FvwmDebug requests that fvwm dump a complete window list into its pipe. These packets are terminated with and "end Send_WindowList" message.

INVOCATION

FvwmDebug can be invoked by inserting the line 'Module FvwmDebug' in the .fvwmrc file. This can be placed on a line by itself, if FvwmDebug is to be

spawned during fvwm's initialization, or can be bound to a menu or mouse button or keystroke to invoke it later. Fvwm will search directory specified in the ModulePath configuration option to attempt to locate FvwmDebug.

CONFIGURATION OPTIONS

There are no configuration options.

AUTHOR

Robert Nation

2.8.8 Fvwm Goodstuff Manual Page

NAME

GoodStuff - the FVWM button panel module

SYNOPSIS

GoodStuff is spawned by fvwm, so no command line invocation will work.

DESCRIPTION

The GoodStuff module provides a button panel which sits on the X terminal's root window. The user can press the buttons at any time, and trigger invocation of a user-specified command by the window manager. GoodStuff only works when fvwm is used as the window manager.

The button panel can be of any configuration or geometry, and can have monochrome or color icons to represent the actions which would be invoked.

COPYRIGHTS

The GoodStuff program, and the concept for interfacing this module to the Window Manager, are all original work by Robert Nation

Copyright 1993, Robert Nation. No guarantees or warranties or anything are provided or implied in any way whatsoever. Use this program at your own risk. Permission to use this program for any purpose is given, as long as the copyright is kept intact.

INITIALIZATION

During initialization, GoodStuff will search for a configuration file which describes the button panel geometry, color, icons, and actions. The format of this files will be described later. The configuration file will be the one which fvwm used during its initialization.

If the GoodStuff executable is linked to another name, ie ln -s GoodStuff TermStuff, then another module called TermStuff can be started, with a completely different configuration than GoodStuff, simply by changing the keywords Good??? to Term???. This way multiple button-bars can be used.

INVOCATION

GoodStuff can be invoked by inserting the line 'Module GoodStuff' in the .fvwmrc file. This can be placed on a line by itself, if GoodStuff is to be spawned during fvwm's initialization, or can be bound to a menu or mouse button or

keystroke to invoke it later. Fvwm will search directory specified in the ModulePath configuration option to attempt to locate GoodStuff.

CONFIGURATION OPTIONS

The following options can be placed in the .fvwmrc file

***GoodStuffGeometry geometry**

Specifies the GoodStuff window location and/or size. If the size is not specified, GoodStuff will auto-size itself in a reasonable manner. The geometry is a standard X11 window geometry specification.

***GoodStuffFont font**

Specifies the font to be used for labeling the buttons.

***GoodStuffRows rows**

Specifies the number of rows of buttons to be created. If unspecified, 2 rows will be used.

***GoodStuffColumns columns**

Specifies the number of columns of buttons to be created. If unspecified, the number of columns will be set to the number of buttons requested, divided by the number of rows. If both the rows and columns are specified, but do not specify as many buttons as are defined, then the users columns specification will be ignored.

***GoodStuffFore color**

Specifies the color for button label text and monochrome icons.

***GoodStuffBack color**

Specifies the color for the buttons.

***GoodStuff[(w x h)] label icon command**

Specifies a window manager built-in command, as described in the fvwm man page, which should be executed when a button is pressed. The label field is the text which will be written on the button. The icon field specifies an X11 bitmap file or XPM color icon file, containing the icon to display on the button. GoodStuff will search through the path specified in the fvwm IconPath or PixmapPath configuration items to find the icon file. If the optional (w x h) is specified, then this button will be w times the normal button width and h times the normal button height. If label is "-" then the button label will not be displayed.

If command is an fvwm Exec command, then the button will remain pushed in until a window whose name or class matches the qouted portion of the command is encountered. This is intended to provide visual feed- back to the user that the action he has requested will be performed. If the qouted portion contains no characters, then the button will pop out immediately. Note that users can continue pressing the button, and re-executing the command, even when it looks "pressed in."

A single extension to the fvwm built-in command set is provided. A command of the form: *GoodStuff junk clock.xpm Swallow "Clock" rclock & will cause Good-

Stuff to spawn an rclock process, and capture the first window whose name or resource is "Clock", and display it in the button-bar. This is handy for applications like xclock, xbiff, and xload. Modules can be swallowed by specifying SwallowModule instead of just "swallow" NOTE: if you use xclock for this application, you will want to specify xclock -padding 0.

SAMPLE CONFIGURATION

The following are excepts from a .fvwmrc file which describe GoodStuff initialization commands:

```
#####################################################################
# Load any modules which should be started during
# fvwm initialization
ModulePath /usr/lib/X11/fvwm:/usr/bin/X11
Module GoodStuff
# Make goodstuff title-bar-less, sticky, and give it an icon
NoTitle GoodStuff
Sticky GoodStuff
Icon "GoodStuff" toolbox.xpm
#####################################################################
#Definitions used by the modules
*GoodStuffFore Black
*GoodStuffBack #908090
*GoodStuffGeometry -135-5
*GoodStuffRows 1
*GoodStuffFont -adobe-helvetica-medium-r-*-*-12-*-*-*-*-*-*-*
*GoodStuff Resize resize.xpm Resize
*GoodStuff Move arrows2.xpm Move
*GoodStuff Lower Down Lower
*GoodStuff Raise Up Raise
*GoodStuff Kill bomb.xpm Destroy
*GoodStuff Dopey rterm.xpm Exec "big_win" xterm -T big_win -geometry
80x50&
*GoodStuff Snoopy rterm.xpm Exec "small_win" xterm -T small_wi &
*GoodStuff Xcalc rcalc.xpm Exec "Calculator" xcalc &
*GoodStuff XMag magnifying_glass2.xpm Exec "xmag" xmag &
*GoodStuff Mail mail2.xpm Exec "xmh" xmh &
*GoodStuff(4x1) - whatever SwallowModule "Fvwm Pager" FvwmPager 0 3
```

The last line is a little tricky - it spawns an FvwmPager module, and captures it to display in a quadruple width button. Since "-" is used for the button label, the Pager will be resized to be as big as possible within the button's borders. "whatever", of course, is a non-existent icon file, and will be ignored by GoodStuff.

AUTHOR

Robert Nation

2.8.9 FvwmIconBox Manual Page

NAME

FvwmIconBox - the FVWM iconbox module

SYNOPSIS

FvwmIconBox is spawned by fvwm, so no command line invocation will work.

DESCRIPTION

The FvwmIconBox module provides an icon manager. The user can do operations, like iconify and deiconify, for each icon shown in the module via mouse and keyboard.

FvwmIconBox reads the same .fvwmrc file as fvwm reads when it starts up, and looks for lines similar to "*FvwmIconBoxFore green".

COPYRIGHTS

The FvwmIconBox program is original work of Nobutaka Suzuki.

Copyright 1994, Nobutaka Suzuki. No guarantees or warranties or anything are provided or implied in any way whatsoever. Use this program at your own risk. Permission to use this program for any purpose is given, as long as the copyright is kept intact.

INITIALIZATION

During initialization, FvwmIconBox will eventually search a configuration file. The configuration file is the same file that fvwm used during initialization.

If the FvwmIconBox executable is linked to another name, ie ln -s FvwmIconBox MoreIconBox, then another module called MoreIconBox can be started, with a completely different configuration than FvwmIconBox, simply by changing the keyword FvwmIconBox to MoreIconBox. This way multiple clutter-reduction programs can be used.

INVOCATION

FvwmIconBox can be invoked by binding the action 'Module FvwmIconBox' to a menu or key-stroke in the .fvwmrc file. Fvwm will search directory specified in the ModulePath configuration option to attempt to locate FvwmIconBox.

CONFIGURATION OPTIONS

FvwmIconBox reads the same .fvwmrc file as fvwm reads when it starts up, and looks for lines as listed below:

***FvwmIconBoxFore color**

Tells the module to use color instead of white for the window foreground. This option affects only the foreground color of background_bitmap specifed in *FvwmIconBoxPixmap option described below.

***FvwmIconBoxBack color**

Tells the module to use color instead of black for the window background.

***FvwmIconBoxIconFore color**

Tells the module to use color instead of black for non-selected icon text.

***FvwmIconBoxIconBack color**

Tells the module to use color instead of white for the non-selected icon background.

***FvwmIconBoxIconHiFore color**

Tells the module to use color instead of black for selected icon text.

***FvwmIconBoxIconHiBack color**

Tells the module to use color instead of white for the selected icon background.

***FvwmIconBoxPixmap pixmap**

Tells the module to use pixmap for the window background_pixmap.

***FvwmIconBoxFont fontname**

Tells the module to use fontname instead of fixed for text.

***FvwmIconBoxSortIcons**

Tells the module to sort all icons in iconbox in alphabetical order.

***FvwmIconBoxPadding number**

Specifies the number of pixels between icons. The default value is 5.

***FvwmIconBoxPlacement primary secondary**

Specifies icon placement policy. primary and secondary can be Top, Bottom, Left and Right. The following eight combinations are available:

primary secondary

Left Top

Left Bottom

Right Top

Right Bottom

Top Left

Top Right

Bottom Left

Bottom Right

Top: Icons are placed from top to bottom.

Bottom: Icons are placed from bottom to top.

Left: Icons are placed from left to right.

Right: Icons are placed from right to left.

For example, when the placement is "Left Top", icons are placed from left to right, and new rows are added from top to bottom. The default vale is "Left Bottom".

***FvwmIconBoxLines**

Specifies the number of icons placed in a line. If primary is Left or Right, this value specifies the number of columns. If primary is Top or Bottom, this value specifies the number of rows. For example, seven icons are placed in a row if *FvwmIconBoxLines is 7 and *FvwmIconBoxPlacement is "Left Top". The default value is 6.

***FvwmIconBoxHideSC direction**

Specifies the undisplayed scroll bar . Direction can be either Horizontal or Vertical.

***FvwmIconBoxGeometry <width>x<height>{+-}<X>{+-}<Y>**

Specifies the location and / or size of FvwmIconBox. width and height are measured in icons, not pixels. The default value is 6x1+0+0.

***FvwmIconBoxMaxIconSize <width>x<height>**

Specifies the maximum size of icon bitmap. A bitmap larger than this size is clipped to this size. The default value is 48x48.

***FvwmIconBoxMouse Button Action Response[, Response]**

Tells the module to do Responce when Action is done on button Button. Available Responsees are built-in commands in Fvwm (e.g. Iconify, Delete, Focus) and available Actions are Click and DoubleClick.

***FvwmIconBoxKey Key Response[, Response]**

Tells the module to do Response when Key is pressed. Available Responcees are, besides Fvwm built-in commands, the following six FvwmIconBox built-in commands: Next, Prev, Left, Right, Up, and Down.

Next: Change the hilited-icon to the next.

Prev: Change the hilited-icon to the previous.

Left: Move the slider of the horizontal scrollbar to left. Icons move to right accordingly.

Right: Move the slider of the horizontal scrollbar to right. Icons move to left accordingly.

Up: Move the slider of the vertical scrollbar to up. Icons move to down accordingly.

Down: Move the slider of the vertical scrollbar to down. Icons move to up accordingly.

***FvwmIconBox windowname bitmap-file**

Specifies the bitmap to be displayed in iconbox for windowname. Windowname can be window name, class name, or resource name. Windowname can contain "*" and "?" like Fvwm configuration file. The bitmap-file is either the full path name to a bitmap file, or a file in the IconPath or PixmapPath. If bitmap-file is specefied to be "-", the icon for a window corresponding to win- downame is not shown in the iconbox.

SAMPLE CONFIGURATION

The following are excepts from a .fvwmrc file which describe FvwmIconBox initialization commands:

```
###############################################################
# Make no icon to be shown in root window
# by spcifying in your .fvwmrc
# (1) "SuppressIcons" if PRUNE is defined in configure.h, or
# (2) "Style "*" NoIcon" if PRUNE is defined.
#SuppressIcons
Style "*" NoIcon
 ###############################################################
*FvwmIconBoxIconBack #cfcfcf
*FvwmIconBoxIconHiFore black
*FvwmIconBoxIconHiBack LightSkyBlue
*FvwmIconBoxBack #5f9ea0
#*FvwmIconBoxFore blue
*FvwmIconBoxGeometry 5x1+0+0
*FvwmIconBoxMaxIconSize 64x38
*FvwmIconBoxFont -adobe-helvetica-medium-r-*-*-12-*-*-*-*-*-*-*
*FvwmIconBoxSortIcons
*FvwmIconBoxPadding 4
*FvwmIconBoxLines 10
*FvwmIconBoxPlacement Left Top
*FvwmIconBoxPixmap fvwm.xpm
#*FvwmIconBoxHideSC Horizontal
#
# mouse bindings
#
*FvwmIconBoxMouse 1 Click RaiseLower
*FvwmIconBoxMouse 1 DoubleClick Iconify
```

```
*FvwmIconBoxMouse 2 Click Iconify -1, Focus
*FvwmIconBoxMouse 3 Click Module "FvwmIdent" FvwmIdent
#
# Key bindings
#
*FvwmIconBoxKey r RaiseLower
*FvwmIconBoxKey space Iconify
*FvwmIconBoxKey d Close
#
# FvwmIconBox built-in functions
#
*FvwmIconBoxKey n Next
*FvwmIconBoxKey p Prev
*FvwmIconBoxKey h Left
*FvwmIconBoxKey j Down
*FvwmIconBoxKey k Up
*FvwmIconBoxKey l Right
#
# Icon file spcifications
#
*FvwmIconBox "*" unknown1.xpm
*FvwmIconBox "Fvwm*" -
*FvwmIconBox "FvwmFileMgr" folder2.xpm
*FvwmIconBox "rxvt" term.xpm
*FvwmIconBox "xterm" xterm.xpm
*FvwmIconBox "editres" editres.xpm
*FvwmIconBox "xman" xman.xpm
*FvwmIconBox "xbiff" mail1.xpm
*FvwmIconBox "xmh" mail1.xpm
*FvwmIconBox "xcalc" xcalc.xpm
*FvwmIconBox "GoodStuff" toolbox.xpm
```

AUTHOR

Nobutaka Suzuki (nobuta-s@is.aist-nara.ac.jp).

2.8.10 FvwmIdent Manual Page

NAME

FvwmIdent - the FVWM identify-window module

SYNOPSIS

FvwmIdent is spawned by fvwm, so no command line invocation will work.

DESCRIPTION

The FvwmIdent module prompts the user to select a target window, if the module was not launched from within a window context in Fvwm. After that, it pops up a window with information about the window which was selected.

FvwmIdent reads the same .fvwmrc file as fvwm reads when it starts up, and looks for lines similar to "*FvwmIdentFore green".

COPYRIGHTS

The FvwmIdent program, and the concept for interfacing this module to the Window Manager, are all original work by Robert Nation and Nobutaka Suzuki.

Copyright 1994, Robert Nation and Nobutaka Suzuki. No guarantees or warranties or anything are provided or implied in any way whatsoever. Use this program at your own risk. Permission to use this program for any purpose is given, as long as the copyright is kept intact.

INITIALIZATION

During initialization, FvwmIdent will eventually search a configuration file which describes the colors and font to use. The configuration file is the same file that fvwm used during initialization.

If the FvwmIdent executable is linked to another name, ie ln -s FvwmIdent MoreIdentify, then another module called MoreIdentify can be started, with a completely different configuration than FvwmIdent, simply by changing the keyword FvwmIdent to MoreIdentify. This way multiple clutter- reduction programs can be used.

INVOCATION

FvwmIdent can be invoked by binding the action 'Module FvwmIdent' to a menu or key-stroke in the .fvwmrc file. Fvwm will search directory specified in the ModulePath configuration option to attempt to locate FvwmIdent. Although nothing keeps you from launching FvwmIdent at start-up time, you probably don't want to.

CONFIGURATION OPTIONS

FvwmIdent reads the same .fvwmrc file as fvwm reads when it starts up, and looks for lines as listed below:

***FvwmIdentFore color**

Tells the module to use color instead of black for text.

***FvwmIdentBack color**

Tells the module to use color instead of black for the window background.

***FvwmIdentFont fontname**

Tells the module to use fontname instead of fixed for text.

AUTHOR

Robert Nation and and Nobutaka Suzuki (nobuta-s@is.aist-nara.ac.jp).

2.8.11 FvwmPager Manual Page

NAME

FvwmPager - the FVWM Pager module

SYNOPSIS

FvwmPager is spawned by fvwm, so no command line invocation will work. From within the .fvwmrc file, FvwmPager is spawned as follows:

```
Module FvwmPager 0 3
```

or from within an fvwm pop-up menu:

```
Popup "Module-Popup"
Title "Modules"
Module "GoodStuff" GoodStuff
Module "NoClutter" FvwmClean
Module "FvwmIdentify" FvwmIdent
Module "Banner" FvwmBanner
Module "Debug" FvwmDebug
Module "SaveDesktop" FvwmSave
Module "Pager" FvwmPager 0 3
EndPopup
```

where "0" is the first desktop to show, and "3" is the last.

DESCRIPTION

The FvwmPager module shows a miniature view of the Fvwm desktops which are specified in the command line. This is a useful reminder of where your active windows are. Windows in the pager are shown in the same color as their fvwm decorations.

The pager can be used to change your viewport into the current desktop, to change desktops, or to move windows around.

Pressing mouse button 1 in the pager will cause you viewport to change to the selected page of the selected desk. If you click with button 1 in the desk-label area, you will switch desks but not pages within the desk.

Dragging mouse button 2 on a miniature view of a window will cause that window to be move to the location where you release the mouse button, but your viewport will not change. If you drag the window out of the pager and onto your

desktop, a full size image of the window will appear for you to place. There is no way to pick up a full size image of the window and move it into the pager, however. Since some mice do not have button 2, I have made provisions to drag windows in the pager by using pressing modifier-1 (usually Alt) and dragging with button 3.

Clicking mouse button 3 on a location within the current desktop will cause the viewport to move to the selected location, but will not align the viewport to a page boundary. Dragging button 3 will cause the viewport to move as you drag. Button 3 operations are limited to the current desktop only.

When iconified, the pager will work as a fully functional current page only pager. Windows and viewports can be moved within the icon of the pager. Users will want to make sure that they have no lines similar to Icon "Fvwm Pager" whatever in their .fvwmrc files.

COPYRIGHTS

The FvwmClean program, and the concept for interfacing this module to the Window Manager, are all original work by Robert Nation.

Copyright 1994, Robert Nation. No guarantees or warranties or anything are provided or implied in any way whatsoever. Use this program at your own risk. Permission to use this program for any purpose is given, as long as the copyright is kept intact.

INITIALIZATION

During initialization, FvwmPager will eventually search a configuration file which describes the time-outs and actions to take. The configuration file is the same file that fvwm used during initialization.

If the FvwmPager executable is linked to another name, ie ln -s FvwmPager OtherPager, then another module called Other- Pager can be started, with a completely different configuration than FvwmPager, simply by changing the keyword FvwmPager to OtherPager. This way multiple pager programs can be used.

KEYBOARD FOCUS CONTROL

You can direct the keyboard focus to any window on the current desktop by clicking with button 2 on its image in the pager. The window does not need to be visible, but it does need to be on the current page.

INVOCATION

The invocation method was shown in the synopsis section

CONFIGURATION OPTIONS

FvwmPager reads the same .fvwmrc file as fvwm reads when it starts up, and looks for certain configuration options:

***FvwmPagerGeometry geometry**

Completely or partially specifies the pager windows location and geometry, in standard X11 notation. In order to maintain an undistorted aspect ratio, you

might want to leave out either the width or height dimension of the geometry specification

***FvwmPagerRows rows**

Tells fvwm how many rows of desks to use when laying out the pager window.

***FvwmPagerColumns columns**

Tells fvwm how many columnss of desks to use when laying out the pager window.

***FvwmPagerIconGeometry geometry**

Specifies a size (optional) and location (optional) for the pager's icon window. Since there is no easy way for FvwmPager to determine the height of the icon's label, you will have to make an allowance for the icon label height when using negative y-coordinates in the icon location specification (used to specify a location relative to the bottom instead of the top of the screen).

***FvwmPagerStartIconic**

Causes the pager to start iconified.

***FvwmPagerFont font-name**

Specified a font to use to label the desktops. If font_name is "none" then no desktop labels will be displayed.

***FvwmPagerSmallFont font-name**

Specified a font to use to label the window names in the pager. If not specified, the window labels will be omitted. Window labels seem to be fairly useless for desktop scales of 32 or greater.

***FvwmPagerFore color**

Specifies the color to use to write the desktop labels, and to draw the page-grid lines.

***FvwmPagerBack color**

Specifies the background color for the window.

***FvwmPagerHilight color**

The active page and desk label will be highlighted by using this background pattern instead of the normal background.

***FvwmPagerLabel desk label**

Assigns the text label to desk desk in the pager window. Useful for assigning symbolic names to desktops, ie

*FvwmPagerLabel 1 Mail

*FvwmPagerLabel 2 Maker

*FvwmPagerLabel 2 Matlab

DeskTopScale number

If the geometry is not specified, then a desktop reduction factor is used to calculate the pager's size. Things in the pager window are shown at 1/number of the actual size.

AUTHOR

Robert Nation

2.8.12 FvwmScroll Manual Page

NAME

FvwmScroll - the FVWM scroll-bar module

SYNOPSIS

FvwmScroll is spawned by fvwm, so no command line invocation will work.

DESCRIPTION

The FvwmScroll module prompts the user to select a target window, if the module was not launched from within a window context in Fvwm. After that, it adds scroll bars to the selected window, to reduce the total desktop space consumed by the window.

FvwmScroll reads the same .fvwmrc file as fvwm reads when it starts up, and looks for lines similar to "*FvwmScrollFore green".

FvwmScroll should not be used with windows which move or resize themselves, nor should it be used with windows which set the WM_COLORMAP_WINDOWS property. Operation is fine with windows that have a private colormap.

COPYRIGHTS

The FvwmScroll program, and the concept for interfacing this module to the Window Manager, are all original work by Robert Nation.

Copyright 1994, Robert Nation. No guarantees or warranties or anything are provided or implied in any way whatsoever.

Use this program at your own risk. Permission to use this program for any purpose is given, as long as the copyright is kept intact.

INITIALIZATION

During initialization, FvwmScroll will eventually search a configuration file which describes the colors to use. The configuration file is the same file that fvwm used during initialization.

If the FvwmScroll executable is linked to another name, ie ln -s FvwmScroll MoreScroll, then another module called MoreScroll can be started, with a completely different configuration than FvwmScroll, simply by changing the keyword FvwmScroll to MoreScroll.

INVOCATION

FvwmScroll can be invoked by binding the action 'Module FvwmScroll x y' to a menu or key-stroke in the .fvwmrc file. The parameter x and y are integers, which

describe the horizontal and vertical window size reduction. Fvwm will search directory specified in the ModulePath configuration option to attempt to locate FvwmScroll. Although nothing keeps you from launching FvwmScroll at start-up time, you probably don't want to.

CONFIGURATION OPTIONS

FvwmScroll reads the same .fvwmrc file as fvwm reads when it starts up, and looks for lines as listed below:

***FvwmScrollFore color**

Tells the module to use color instead of grey for scroll bars themselves.

***FvwmScrollBack color**

Tells the module to use color instead of black for the window background.

BUGS

When the scroll bars are removed by clicking on the button in the lower right corner, the window does not restore its location correctly.

AUTHOR

Robert Nation

2.8.13 FvwmWinlist Manual Page

NAME

FvwmWinList - the FVWM window list module

SYNOPSIS

FvwmWinList is spawned by fvwm, so no command line invocation will work.

DESCRIPTION

The FvwmWinList module provides a widow list made up of buttons, each corresponding to a window that FVWM is managing. Clicking on the buttons with any of the three mouse buttons will either do a default action or can be user configured. Like the other modules, FvwmWinList only works when fvwm is used as the window manager.

COPYRIGHTS

The FvwmWinList module is the original work of Mike Finger.

Copyright 1994, Mike Finger. The author makes no guarantees or warranties of any kind about the use of this module. Use this modules at your own risk. You may freely use this module or any portion of it for any purpose as long as the copyright is kept intact.

INITIALIZATION

During initialization, FvwmWinList will scan the same configuration file that FVWM used during startup to find the options that pertain to it. These options are discussed in a later section.

INVOCATION

FvwmWinList can be invoked by fvwm during initialization by inserting the line 'Module FvwmWinList' in the .fvwmrc file.

FvwmWinList can also be bound to a keystroke, mouse button, or menu option to be invoked later, in this case using 'Transient' as an argument will cause Fvwm-WinList to resemble the built in window list.

FvwmWinList must reside in a directory that is listed in the ModulePath option of FVWM for it to be executed by FVWM.

CONFIGURATION OPTIONS

The following options can be placed in the .fvwmrc file

***FvwmWinListGeometry {+-}<X>x{+-}<Y>**

Specifies the location and gravity of the FvwmWinList window. At the current time, size is not supported and FvwmWinList will resize itself as buttons are added. If the NoAnchor option is not specified then the windows gravity corner will be anchored, and the window will grow in the opposite direction. (i.e. If the geometry is specified -5-5, that is SoutEastGravity. This will cause the window to draw up and to the left as windows are added)

***FvwmWinListFont font**

Specifies the font to be used for labeling the buttons.

***FvwmWinListFore color**

Specifies the color to use for the button names.

***FvwmWinListBack color**

Specifies the color for the buttons.

***FvwmWinListUseSkipList**

Tells FvwmWinList to not show the windows that are listed on a WindowListSkip line if the configuration file.

***FvwmWinListNoAnchor**

By default, FvwmWinList will anchor the gravity corner so the window will grow in the opposite direction. This undoes that option.

***FvwmWinListUseIconNames**

Tells FvwmWinList to use the icon name of the window instead of the full window name. This is useful to keep the width of the window small.

***FvwmWinListAction action response[,reponse...]**

Tells FvwmWinList to do response when action is done. The currently supported actions are: Click1, Click2, Click3. The currently supported responses are any fvwm built-in commands, including modules and functions.

SAMPLE CONFIGURATION

The following are excepts from a .fvwmrc file which describe FvwmWinList initialization commands:

########

```
# Pop up the window list in tranient mode on button 3 press & hold
Mouse 3 R A Module "FvwmWinList" FvwmWinList Transient
############################ Window-Lister ################################
*FvwmWinListBack DarkOliveGreen
*FvwmWinListFore PaleGoldenRod
*FvwmWinListFont -*-new century schoolbook-bold-r-*-*-*-120-*-*-*-*-*-*
*FvwmWinListAction Click1 Iconify -1,Raise
*FvwmWinListAction Click2 Iconify
*FvwmWinListAction Click3 Module "FvwmIdent" FvwmIdent
*FvwmWinListUseSkipList
*FvwmWinListUseIconNames
*FvwmWinListGeometry -50-85
# I like it anchored
#*FvwmWinListNoAnchor
```

AUTHOR

Mike Finger (mfinger@mermaid.micro.umn.edu) (Mike_Finger@atk.com) (dood-man on IRC, check the #linux channel)

File Transfer Operations

This section discusses file transfer operations and three file transfer protocols, TFTP, FTP, and NFS that the ViewStation uses to load files from the host.

3.0 Section 3.0 - Network and Host System Operations

3.1 Network File Transfer Operations

Many of the ViewStation's operations depend on getting files from the host computer. These include its loading its server code for booting, loading font files, downloadable clients, and so on. The ViewStation knows the paths to these files but depends on the host's file transfer protocols to move them.

3.1.1 ViewStation File Transfer Protocols

There are several file transfer protocols that can move files to the ViewStation. These are NFS, TFTP (ViewStation's default), and FTP.

These processes are all configured and run on the host computer. The ViewStation depends on the availability and correct operation of these protocols.

3.1.1.1 Selecting a File Transfer Protocol

The ViewStation uses the file transfer protocol selected for a given host by the entry in the Remote Hosts menu of Setup Mode. This menu defines the protocol for each host, so you can communicate with one host via TFTP and another via NFS, for instance.

Other selections on the menu are also important for file transfer operations, such as access parameters (usernames and passwords), if applicable, and local and remote mount directories.

Note that NFS is available for booting the ViewStation with version 3.2 of HDSware. Earlier versions don't support NFS in the bootprom.

3.1.2 NFS

NFS (Network File System) is a file transfer protocol from Sun Microsystems. NFS allows the HDS ViewStation to access fonts and/or local clients as if they were on a local disk drive. Note that support for NFS is not present in the ViewStation's bootprom, so NFS is not available for first-time booting of the ViewStation.

Setting up NFS on your host system involves decisions about security and file system management. Consult your system administrator.

NFS allows the ViewStation to perform transparent file access over the network. The ViewStation dynamically mounts the filesystem and then transfers the file; it does not use a static mount. The HDSware tape may be loaded on any filesystem on any host available to the ViewStation.

There are two ways to handle NFS processes - one for Sun OS 4.1.x (or BSD types) and one for Solaris 2.x (or Sys V R4.x types). The NFS functions are the same, but the syntax and file names are different.

3.1.2.1 For Sun OS 4.1.x and BSD type UNIX:

To use NFS, you must export the filesystems and/or directories the ViewStation requires. For example, suppose you loaded the HDSware tape in the */home/HDS* filesystem.

- First, verify that the */home/HDS* directory is listed in the */etc/exports* file.
- If filesystem access needs to be restricted to the ViewStation, add the ViewStation's name to the access list for the desired filesystem. For example (in the */etc/exports* file):

  /home/HDS -access=hdssun5:hdsfx2

  Thus, only hdssun5 and hdsfx2 have mount privileges to the */home/HDS* filesystem. A read-only flag may also be added for these hosts' access permission.

  /home/HDS -access=hdssun5:hdsfx2, ro

- Check the transfer block size. The ViewStation uses a 8192 byte block size by default. If 8192 is too large for your network environment, you must set the correct value in the ViewStation's configuration file, which is */hds-fx/config/config.fx*. The ViewStation resource is:

  HDSsetup.NFS.block.size: 8192 *(permitted values are 128 to 8192)*

- Finally, run the command to export the filesystem:
  exportfs -a

Check your system NFS man pages for more details and arguments.

3.1.2.2 For Solaris 2.x and Sys V R4.x type UNIX:

To use NFS, you must export the filesystems and/or directories the ViewStation requires. For example, suppose you loaded the HDSware tape in the */home/HDS* filesystem.

- First, add this line to the */etc/dfs/dfstab* file.

  share -F nfs -oro -d "For HDS X Terminals" /home/HDS

Note that this line sets "read-only" permission, and has a "-d" comment line argument.

- Check the transfer block size. The ViewStation uses a 8192 byte block size by default. If 8192 is too large for your network environment, you must set the correct value in the ViewStation's configuration file, which is */hds-fx/config/config.fx*. The ViewStation resource is:

  HDSsetup.NFS.block.size: 8192 *(permitted values are 128 to 8192)*

- Finally, run the command to export the filesystem:

  shareall

Check your system NFS man pages for more details and arguments.

3.1.2.3 Setting Up the ViewStation for NFS

1) Open the Setup Mode window

2) Click on the Setup Options icon

3) Select the Remote Hosts line

4) Select Remote Host 1 (or the host you want)

5) Enter the Host Name, Network Address (IP), and Host Type

6) Set the Access Protocol to NFS

7) Set both the Remote Mount Directory and the Local Mount Directory to the HDSware directory on exported file system on the host computer.

3.1.2.4 Using NFS for ViewStation File Access

The HDS ViewStation uses an internal file system that closely resembles the NFS file system, using mount points to establish local and remote path specifications. Local and Remote Mount Directories are entered in the Remote Hosts menu of Setup Mode, for each different host if desired. Individual paths, such as for fonts or clients, are accessed by means of the information the mount points provide.

A Local Mount Directory defines an internal ViewStation location for the server to look for things. The file system maintains an internal list of these Local Mount Directories and searches that list when it opens a file. (The list contains all the specified internal file locations, like HDS$$ROM, HostNames, Local Mount Directories, ViewStation device names like /dev/flash or /dev/tty1, etc.) If the Local Mount Directory is specified, then it is used as the mount point. If no Local Mount Directory is specified, the Host Name is prepended with a leading slash and it becomes the mount point.

A Remote Mount Directory defines a location on the remote host as a starting place for the ViewStation to look for things. The Remote Mount Directory is typically a directory location where the HDSware tape was loaded or some special fils are located. The Mount Point is the main directory, with subdirectories specified by the path itself. The Remote Mount Directory is prepended to the path (after the mount point is stripped off) to give the complete file location.

For example, if the Local Mount Directory specifies "/fonts" (the leading slash is necessary), then a Mount Point is created in the ViewStation. The fontpath is specified as "/fonts/100dpi,/fonts/misc". The Remote Mount Directory for that host is "/usr2/lib/X11/HDS/hds-fx/fonts". Thus, when a font is requested, the ViewStation finds the /fonts mount point and the fontpath, strips off the mount point, prepends the Remote Mount Directory, and locates the desired font in the reconstructed path "/usr2/lib/X11/HDS/hds-fx/fonts/100dpi", etc.

This file structure provides additional functionality for using the ViewStation on your network.

For instance, if you want to have a backup font host, you could specify "/fonts" as Local Mount Directories on both Host1 and Host2. The ViewStation's mount point would first go to Host1, and if that file open failed, it would go on to the mount point on Host2, and so on. The Remote Mount Directory could be different for Host1 and Host2, but as long as those were correctly specified on the Remote Hosts menus, the files would be found.

Another example of customization for the ViewStation would be defining different fontpaths for different hosts. If you had one host that needed OPEN LOOK fonts and another host that needed word processing fonts, but you didn't want to load all fonts on all hosts, you could use this configuration to handle the special fontpaths.

1. Set the ViewStation's fontpath as follows (on the User Preferences Menu):

   /fonts1/100dpi,/fonts2/Xol,/fonts3/WPfonts

2. On the Remote Hosts Menu, set:

For Host 1

   Local Mount Directory: /fonts1

   Remote Mount Directory: /usr/libX11/hds-fx/fonts

which results in a full fontpath of:

   /host1/usr/lib/X11/hds-fx/fonts/100dpi

For Host 2

   Local Mount Directory: /fonts2

   Remote Mount Directory: home/openwin/MITXol/fonts

which results in a full fontpath of:

   /host2/home/openwin/MITXol/fonts/Xol

For Host 3

   Local Mount Directory: /fonts3

   Remote Mount Directory: /usr3/lib/Asterix/fonts

which results in a full fontpath of:

/host3/usr3/lib/Asterix/fonts/WPfonts

3.1.3 TFTP

TFTP is Trivial File Transfer Protocol. It is the default file access protocol for the HDS ViewStation. If TFTP doesn't work, the simplest thing is to select another file transfer protocol, either FTP or NFS. This is selectable for each host in the Remote Hosts menu of Setup Mode.

Some host computers (such as Sun and HP) do not start TFTP automatically. This is the ViewStation's default protocol.

3.1.3.1 Using TFTP

TFTP operates invisibly, since it requires no passwords or user identification. You can use TFTP manually by typing "tftp" at your user prompt. If it is running, you will get a "tftp>" prompt and you can do "?" to get information about TFTP commands.

Note that TFTP has a file size limitation of 16 MB. This prevents its use in loading the filesystem file for the ViewStation's hard disk and may also limit its utility with moving other large files.

3.1.3.2 Starting TFTP on your Host

A startup line for TFTP is present in the */etc/inetd.conf* file on most host systems, but is usually commented out. With the comment character removed, the TFTP protocol is often run in "secure" mode. If you want to run TFTP as your file transfer protocol for the ViewStation, you must verify that it is running, and then decide if you want to use it in secure mode or not.

To start your TFTP service on a host (you must be root user to do these steps):

1. Check to see if TFTP was started. Look in the */etc/inetd.conf* file to see if the TFTP line has a comment character. The line looks like this:

#tftp dgram udp wait root /usr/etc/in.tftpd in.tftpd -s /tftpboot

2. Remove the comment character (#) from the beginning of the line.

3. Restart the inetd process. Find the process identification number (PID) from the following command:

```
ps -aux | grep inetd
```

Then issue this command:

```
kill -1 <PID>
```

This restarts the inetd process and starts the TFTP daemon.

3.1.3.3 TFTP in Secure Mode

If the TFTP process is in secure mode (check for the "-s /tftpboot" at the end of the line), the TFTP process must change to the */tftpboot* directory to move requested files. If your files are not in that directory, or you don't want to put them there, or

you want to transfer files from other directories, you will have to make some changes.

You have several options:

■ **Use secure TFTP**

This means that all your files must be located in the */tftpboot* directory. This partition is usually fairly small, so check the available storage space with the "df" command if you want to use this location. The HDSware tape requires about 60 MB for a complete installation, or about 15 MB for a minimal installation.

Note that the file transfer operation pre-pends */tftpboot* to whatever directory you specify in your paths, so take this into account when you enter your HDSware path in the HDSware menu of Setup Mode or in your configuration file.

■ **Use unsecure TFTP**

Remove "-s /tftpboot" from the TFTP line. This means that TFTP is not secure and can move files from or to any directory on the system. This is convenient, but you may want to keep the security features.

■ **Link your files to */tftpboot***

You could create a tftpboot directory in your */usr* partition, */usr/tftpboot*, and link it to */tftpboot*, as follows:

```
cp -r /tftpboot /usr
rm -rf /tftpboot
ln -s /usr/tftpboot /tftpboot
```

3.1.3.4 TFTP in Secure Mode on Sun Systems

If the TFTP process is in secure mode (check for the "-s /tftpboot" at the end of the line), it must change to the */tftpboot* directory to move requested files. If your files are not in that directory, or you don't want to put them there, or you want to transfer files from other directories, you will have to make some changes. Note that this change to the */tftpboot* directory pre-pends that directory to your path, so enter your HDSware path accordingly.

You have several options:

1. Use secure TFTP - this means that all the files you want to transfer must be located in the secure directory, which is */tftpboot* by default; it can be any directory you specify on the tftp startup line. This partition is usually fairly small, so check the available storage space with the "df" command. The HDSware tape requires about 21 MB for a complete installation, or about 9 MB for a minimal installation.

Example #1: If the secure path is /tftpboot, then

```
cd /tftpboot
ln -s . tftpboot
```

Now a file access to /tftpboot/foo is successful.

Example #2: If the secure path is /usr/tftpboot, then

```
cd /usr/tftpboot
ln -s /usr .
```

Now a file access to /usr/tftpboot/foo is successful.

2. Do not use secure TFTP - Remove "-s /tftpboot" from the TFTP line and restart the inetd process, as described above. This means that TFTP is not secure and will move files from or to any directory on the system. This is convenient, but you may want to keep the security features.

3. Use a file Link - Link your files to */tftpboot*. For instance, you could create a tftpboot directory in your */usr* partition, */usr/tftpboot*, and link it to */tftpboot*, as follows:

```
cp -r /tftpboot /usr
rm -rf /tftpboot
ln -s /usr/tftpboot /tftpboot
```

3.1.3.5 TFTP on Hewlett Packard Systems

Typical HP installations do not start the TFTP daemon. When TFTP is started, various security mechanisms limit file access. These security mechanisms are discussed below.

On HP systems there is a startup line for TFTP present in the */etc/inetd.conf* file, but it is usually commented out. With the comment character removed, the TFTP protocol must be restarted. Note that TFTP can read only from directories available through the security mechanisms.

If you want to run TFTP as your file transfer protocol for the ViewStation, you must start it or verify that it is running, and then add the appropriate HDSware directory to the TFTP access list.

To start your TFTP service on the HP and add the HDSware directory (you must be root user to do these steps):

1. Check to see if TFTP is running; check for a comment character on the TFTP line in the */etc/inetd.conf* file. The line looks like this:

```
#tftp dgram udp wait root /etc/in.tftpd tftpd\
/usr/lib/uxinstlf.700\
/usr/lib/uxinstkern.700
```

2. Remove the comment character (#) from the beginning of the line.

3. Add the appropriate HDSware directory (in this case: */usr/lib/HDSware)* to the TFTP access list.

```
tftp dgram udp wait root /etc/in.tftpd tftpd\
/usr/lib/uxinstlf.700\
/usr/lib/uxinstkern.700\
/usr/lib/HDSware
```

Note the backslash character as a designator of line continuation.

4. Restart the inetd process. Find the process identification number (PID) from the following command:

ps -aux | grep inetd

Then issue this command:

kill -1 <PID>

This restarts the inetd process which starts the TFTP daemon.

TFTP in "Secure Mode" on HP Systems

TFTP on HP/UX systems operates with security limitations. Access to the file system is limited by the TFTP configuration. There are three possibilities for setting the TFTP security.

1. File access is limited by the filenames or directories specified on the TFTP startup line in */etc/inetd.conf*, as described in the previous section.

```
tftp  dgram  udp  wait  root  /etc/in.tftpd  tftpd\
    /usr/lib/uxinstlf.700\
    /usr/lib/uxinstkern.700\
    /usr/lib/HDSware
```

In this case, TFTP file transfers are permitted for the two .700 files and any files in the */usr/lib/HDSware* directory. You could add other file names or directories as needed. Symbolic links are permitted in this path.

2. File access is limited to the home directory of the pseudo-user "tftp". In general, the "tftp" user is identified in the */etc/passwd* file as follows:

```
tftp:*:510:guest:tftp server:/usr/tftpdir:/bin/false
```

where tftp is the username, * is the password, guest is the membership, */usr/tftpdir* is the home directory, and the default shell is bogus since there is never a login to the tftp user's shell.

A TFTP file transfer initiates a chroot() call so it changes its root directory to the home directory of the tftp pseudo-user, which is */usr/tftpdir* in this case. All files transferred must be in this directory, and the file paths you enter must take this into account. Symbolic links are not permitted in this directory. This security mechanism for TFTP simulates the behavior of "TFTP secure mode" found in other operating systems.

3. You can specify both of these TFTP security variables. When you request a TFTP file transfer, TFTP first looks for the file using its home directory (which is */usr/tftpdir* in this example) as the root directory of its search. If the requested file is not found, it searches from the "root" directory specified in the TFTP startup line in */etc/inetd.conf*, which in the example above is */usr/lib/HDSware*.

Check the man pages on your HP system for the TFTP arguments and options for your system. Check the relevant files to see what the current TFTP configuration is on your system.

3.1.3.6 TFTP File Transfer Protocol on IBM RS6000 Systems

The IBM RS6000 does not start TFTP in its default configuration.

It is easy to start the TFTP protocol with the SMIT tool. If you use the fast path mechanism with the SMIT tool, this is very simple. Type:

```
smit inetd.conf
```

This points the tool to the inet daemon configuration file and gives you a menu for the daemon's processes, (list, add, change, and delete). If you "list" the processes, you will see that TFTP is absent. Using the "add" command, you can click on TFTP and SMIT adds it to the running daemon processes.

The TFTP daemon does not run in secure mode by default, so it is ready to load the ViewStation's files from any directory.

3.1.4 FTP

FTP is normally included with the host operating system software, but it is often not started (with a Sun installation, for example); refer to the */etc/inetd.conf* file for FTP as above. You should check to see if the FTP daemon is running; check for a comment character on the FTP line in the */inetd.conf* file. The following line should be present in the */etc/inetd.conf* file.

```
ftp stream tcp nowait root /usr/etc/in.ftpd in.ftpd
```

Follow the example above if you need to restart the inetd process.

FTP requires a valid username and a real password to run, so the appropriate accounts should be set up on the Sun (normal login accounts and passwords are sufficient, but they must be entered in the ViewStation's Setup Mode Remote Hosts fields to be used for file loading).

FTP also verifies the user's shell with the *getusershell()* query to the */etc/shells* file. Sun does not create this file in normal installation. Unix System V Release 4 and Solaris 2.x replace the Bourne shell (*sh*) with the Korn shell (*ksh*) as the standard shell. If the Korn shell is not added to the */etc/shells* file, or this file doesn't exist, the query detects the Korn shell as invalid and the FTP request is denied. If you are using the Korn shell, you must make these changes.

3.1.4.1 Anonymous FTP File Transfer

Anonymous FTP file transfers do not require a password, but have other security limitations. All files to be transferred must be in a special anonymous FTP directory. You must set up a special anonymous FTP account if you intend to boot the ViewStation using the FTP protocol. Since FTP requires a username and password, and these can't be entered from Monitor Mode, anonymous FTP is needed to boot.

Refer to your man pages for details about setting up an anonymous FTP account. In summary, you must enter a "anonymous ftp" user in the password file (*/etc/passwd*) with appropriate group and ownership limitations, and then use that anonymous ftp user's home directory as the root directory for file transfers.

Anonymous FTP file transfers initiate a chroot() call, so the new root directory is the home directory for the anonymous ftp user.

In this case, as with "secure TFTP" above (see Section 3.1.3.3), you must understand how these security mechanisms affect the entry of the HDSware path on the ViewStation. Your path on the ViewStation must take into account the new anonymous ftp root directory.

BOOTP Operations

This section describes BOOTP operations and its use with booting and configuring ViewStations.

Section 3.0 - Network and Host System Operations

3.2 BOOTP Operations

BOOTP is a network service program (refer to RFC 951, RFC 1048, and RFC 1084 for specific details) running on your host computer that listens for startup or boot requests. When the BOOTP daemon receives a boot request, it looks in the bootp table (a file called */etc/bootptab*) for information about the machine making the request.

3.2.1 Starting with BOOTP

When you turn the ViewStation on, it sends a Bootp request (four times) as its default behavior. This boot request contains only the ViewStation's hardware address (ethernet address). BOOTP requests are supported across the Token Ring interface (effective with HDSware 3.2). The first host responding, whether from the Ethernet or the Token Ring network, is used as the BOOTP host.

You can disable the BOOTP requests on the Startup Options menu of Setup Mode by selecting "Setup" in the Determine IP address field.

When the BOOTP daemon receives a boot request, it takes the hardware address and matches it to an entry in its bootptab file. From this entry in the file, it takes the information appropriate for that ViewStation and sends a response that gives the ViewStation its assigned IP address, the location of its boot file, and so on.

3.2.1.1 Ethernet Hardware Address

The Ethernet hardware address is a 12 digit number (hexadecimal) unique to each network device. It consists of a six digit number for the vendor (for HDS this is 008096) and six digits more to make this a unique number.

This hardware address is found on a label on the ViewStation's back panel or as part of the ViewStation serial number.

The ViewStation's serial number consists of three characters to give the date of manufacture (first digit is the year - 4=1994; second two digits are the month - 05=May). The remaining six digits are the lowest six digits of the hardware address. Thus c8161b from the serial number, combined with the vendor code, would give the complete hardware address - 008096c8161b.

3.2.1.2 Bootptab Command Fields

The fields in the */etc/bootptab* file have an abbreviation for the entry name (such as **ha** for hardware address, and **ip** for IP address, etc.) and an associated value. These values are sent with the bootp response and received by the ViewStation to start its boot process. Check the syntax for these files on your particular system.

The BOOTP response is limited to 1024 bytes. The ViewStation uses the following Bootp fields:

:ha=	hardware address example: :ha=008096006a5c:\
:ip=	IP address example: :ip=128.91.6.15:\
:hd=	home directory example: :hd=/home/hdssun6/boot:\
:bf=	bootfile < *filename (limited to 128 chars)* > example: :bf=V30-D.hds:\
:gw=	gateway address example: :gw=128.91.2.5:\
:sm=	subnet mask example: :sm=255.255.0.0:\
:ds=	domain nameserver (entry is one or two addresses separated by a space) example: :ds=128.91.3.12 128.91.3.1:\ more than two entries are ignored; if only one is used, secondary is zero
:hn	host (ViewStation) name host(no entry; host does a name lookup based on :ip)
:T128=	(vendor tag) configuration password example: :T128="ApollO":\
:T129=	(vendor tag) configuration filename example: :T129="config.fx":\

:T130= (vendor tag) configuration directory
example: :T130="/home/hds-fx/config":\

BOOTP may be used to load the configuration file (containing Setup Mode values) or load only the ViewStation boot file (containing the server code) or both. Do not confuse these functions.

These entries and values, along with any other BOOTP fields your system may use, are retrieved from the bootp table and sent to the ViewStation in the bootp response. The configuration path and filename are also sent if the vendor tag fields are set. This configuration file contains individual resources and values for Setup Mode fields, which are given in detail in Section 1.7.3.

When you turn on the ViewStation, it sends the BOOTP request automatically. BOOTP then sets the IP address for the ViewStation and loads the software file. You have a root window with the Setup icon in the lower right corner.

3.2.1.3 Problems with BOOTP Starting

If there is a problem with starting the ViewStation via Bootp, try Monitor Mode, or look for these things:

- Verify that BOOTP is running on your system. If you do a "ps" command, you may not see a BOOTP process because it may be sleeping. The best procedure is just to restart it.
- Verify that your entries in the bootptab file are correct; verify the file syntax and the paths and filenames. Check the length of the response; some file names may be truncated for being too long.
- Verify that the file transfer protocol, TFTP by default, is running so the file can be transferred correctly.
- If you are running TFTP in secure mode, remember that this changes the root directory to your secure directory (such as */tftpboot*) and TFTP pre-pends this directory to your path. This must be reflected in your hd and T130 entries, which should then start with the directory below */tftpboot*.
- Note that there is a 64 byte limit (actually 60 bytes) on the "vendor" fields, which includes T128, T129, T130, gw, sm, and ds.

3.2.2 Setting Up BOOTP on your Host

The BOOTP daemon is usually provided with your host's operating system, though it may not be started as part of your system installation script. If you don't have a copy of the BOOTP daemon, you can get the source code or executable files for some systems from the */HDSware/hds-fx/contrib* directory of the HDSware tape or you can probably find a copy on the Internet.

The Bootp daemon is usually started from the */etc/inetd.conf* file, which manages the network services. If this file doesn't have a "bootp" entry, or the entry is commented out, you should add a line to start BOOTP. The syntax for this entry may vary, but it should look something like this:

```
bootps dgram udp wait root /etc/bootpd bootpd -t0 /etc/bootptab
```

This example starts the BOOTP server (bootps), gives the path to the BOOTP daemon itself, then the path to the bootptab, in this case "/etc/bootptab". This command line contains a timeout value of 0 (-t0 - never timeout) so you can always see whether the process is running. Enter the locations appropriate to your system configuration.

You should also check your */etc/services* file for entries for BOOTP's operation. This file sets the appropriate sockets for communication and should have an entry something like this for BOOTP:

```
bootps      67/udp      bootp
bootpc      68/udp
```

These entries indicate that the BOOTP server (bootps) uses socket 67 and the BOOTP client (bootpc) uses socket 68. Check your operating system materials for syntax, though these UDP socket numbers should be uniform across all host types.

You should verify that the BOOTP daemon is running on your host. This can be a little tricky if the BOOTP daemon was initiated in the inetd process, since this doesn't actually start the BOOTP daemon until a BOOTP request is received (the default timeout for the daemon is 15 minutes). If you try to start a ViewStation and it sends a BOOTP request, then the daemon (if it is there) will start and linger long enough for you to check for its process with the **ps** command.

You can start BOOTP manually, or restart the inetd daemon as follows. First locate the inetd process using the **ps** command, then send that process ID a Hangup signal to restart the inetd daemon.

kill -1 <inetd process ID>

If you modify your bootptab, the BOOTP daemon will notice the new file date and reread it when the daemon receives a subsequent BOOTP request.

3.2.2.1 Installing BOOTP on your Host

If Bootp is not installed on your system, you can use the source code for bootp provided in the */HDSware/hds-fx/contrib* directory on the HDSware tape. The */contrib* directory has several executable bootp versions as well as the source code for bootp. Try the source code (BSD or SysV). If it doesn't work, you will have to compile the source code. Use the appropriate steps in the process outlined below.

Step 1. If you want to load the bootp code from the HDSware tape, move to the appropriate directory to unload the tape:

cd /tftpboot

tar xvfB <tape-device> ./hds-fx/contrib/bootp/src

(If your system is System V based, use this directory:)

./hds-fx/contrib/bootp/src/bootp.sys5

(If your system is BSD based, use this directory:)

./hds-fx/contrib/bootp/src/bootp.SUN4

• Read the README file to prepare to build the bootp daemon.

• Check the Makefile to set the configuration file location. The line in the Makefile should read:

```
CONFFILE = -DCONFIG_FILE=\"/etc/bootptab\"
```

• Now build the bootp daemon:

```
make
```

Step 2. When the bootp daemon executable is finished, you must verify that your */etc/services* file contains the following two lines:

```
bootps 67/udp # bootp server
bootpc 68/udp # bootp client
```

Step 3. Edit the */etc/inetd.conf* (or */usr/etc/inetd.conf* for SysV) file to start the bootpd process on reboot. The entry should read:

```
bootps dgram udp wait root /etc/bootpd bootpd
```

At this point, the bootp process should be running on the host.

3.2.2.2 BOOTP Debugging

The HDSware directory for BOOTP (in */HDSware/hds-fx/contrib/bootp/src/bootp.xxx*) contains some sample files that can help you work with BOOTP. A file called "dobootp" shows you how to start BOOTP in stand-alone mode, and sets the debug level to 5. This gives you a powerful diagnostic tool to see how BOOTP is working.

3.2.3 Sample *bootptab* File

This file would normally be located in */etc/bootptab*. If the file doesn't exist, copy the sample file provided with the source code:

```
cp /etc/bootptab.orig /etc/bootptab
```

In the bootptab file, note that lines marked # are comment lines. Note also that the passwords shown here are in clear text, since the bootp table can be read-protected on the system. (A configuration file specified by BOOTP cannot be read-protected, since it must be readable by TFTP, and so passwords occuring there must be encrypted. Configuration files loaded directly by the ViewStation can be read-protected, and so can use clear-text passwords.) The configuration file is loaded from the host which sent the bootp response.

This sample file contains a section for global settings, which would apply to everything on the system and/or also a subnet entry, which would apply to smaller divisions of the entire system. The entries for individual machines are entered last. Refer to the RFC documents and your system documents for the specific details of *booptab* file syntax. This is sometimes a difficult problem; for instance, some systems require the boot file name to have a leading slash, others refuse to read it with the leading slash, and some allow :hd and :bf on the same line, others do not.

```
# First, we define a system entry (such as subnet2) which
# specifies the things every host on that subnet uses.

subnet2.default:\

:hd=/home/HDSware/hds-fx/image:\

:bf=V30-D.hds:\

:T128="Galahad":\

:T129="test.3.cfg":\

:T130="/home/hdssun1/boot":

# Now come the individual ViewStations

marketing#1:\

:tc=subnet2.default:\

:ht=ethernet:\

:ha=008096000167:\

:ip=128.91.5.68:

marketing#2:\

:tc=subnet2.default:\

:hd=:bf=/home/HDSware/hds-fx/image/V30.hds:\

:ht=ethernet:\

:ha=00809600015d:\

:ip=128.91.5.13:
```

This sample file has sample entries for Bootp. Note that the vendor tag fields are present and that the configuration file called for the marketing#2 unit is test.3.cfg.

When you use BOOTP to specify the download of the boot file (server code), several different combinations of bootp fields can be used to attempt the download process. These combinations are listed below in the order they are tried, with the bootp field names shown in bold:

1. **T130** and **T129**
2. **T130** and hds.setup
3. **hd** and **T129** (if there is no T130 field)
4. path of **hd** and **hds.setup** (if there are no T129 or T130 fields)

3.2.3.1 Password Security with BOOTP

If you are using a password (T128) as part of your bootp process, note that the */etc/bootptab* file can have restricted access, but the configuration file (T129) must

have no restrictions so it can be available to the TFTP file transfer protocol. This requires that a password contained in the configuration file must be encrypted.

Refer to Section 1.11.1.6 for more details.

XDMCP Operations

This section describes the operation of XDMCP, the X Display Manager Command Protocol, which runs oun your host computer and manages the ViewStation's display.

Section 3.0 - Network and Host System Operations

3.3 XDMCP Operations

Xdm is a host-based program that manages the X terminal's display and session. It has several functions:

- Presents a login screen; verifies user login and password
- Starts the user session after verification
- Sets up the X environment
- Terminates the environment properly and puts up the login screen again

There are some variants of Xdm that have substantial differences in operation from Xdm but still provide the overall session management. Some of these are the VUE environment by Hewlett Packard, DXsession by Digital Equipment Corp., and Toolchest by Silicon Graphics. These are similar in function to Xdm, but some of the files and syntax may be very different. Consult your man pages for operation of these systems. The ViewStation's "manage my display" request is recognized by all of these systems.

3.3.1 Starting Xdm on your Host

You must be the root user to start Xdm. The Xdm executable and configuration files are in */usr/lib/X11/xdm*. Look at the configuration files to be sure they reflect your system environment. Actual startup will probably be from an *rc* or *rc.local* file on your host.

3.3.2 Xdm Configuration Files

These are the configuration files for X11R5 XDMCP. Older Xdm systems won't have all of these files.

The xdm-config file contains resources specifying xdm related data files;

DisplayManager.servers:	/usr/lib/X11/xdm/Xservers
DisplayManager.errorLogFile:	/usr/lib/X11/xdm/xdm_errors
DisplayManager*resources:	/usr/lib/X11/xdm/Xresources
DisplayManager*setup:	/usr/lib/X11/xdm/Xsetup
DisplayManager*startup:	/usr/lib/X11/xdm/Xstartup
DisplayManager*reset:	/usr/lib/X11/xdm/Xreset
DisplayManager*session:	/usr/lib/X11/xdm/Xsession
DisplayManager*authorize:	false

3.3.2.1 Xdm Authorize

The *authorize* resource controls the user-based access control (different from host-based access control governed by *xhost*). The example in the paragraph above has access control turned off.

The following resources set access control on for the local display and then using the unqualified authorize afterward to set host-based access control for other displays.

```
...
DisplayManager._0.authorize: true
DisplayManager*authorize: false
...
```

Access control is discussed from the ViewStation's point of view in Section 1.11.3.

3.3.2.2 Xresources File

This file contains resources used by Xdm, typically for its login banner. If you want to change these resources, for instance by adding your personalized greeting, you do it by modifying entries in this file.

Look in this file to see the resource names and settings.

3.3.2.3 Xservers File

The Xservers file contains a list of displays which Xdm must manage. These are typically displays which do not have XDMCP and thus cannot select to have Xdm manage them. Normally the ViewStation selects whether Xdm manages its display or not, based on the user's selection in the XDMCP menu in Setup Mode.

This file is usually empty except for a single entry for the host computer's local display:

```
:0 Local local /usr/bin/X11/X :0
```

The Xservers file was the way Xdm managed network devices under X11R3. With X11R4 and R5, individual devices send an Xdm request.

3.3.2.4 Xsetup File

The Xsetup file is a new file with X11R5. In most cases, it is empty. It allows the user to specify any startup commands that should be run BEFORE the Xdm banner is up and the user logs in. This would typically be something like xmodmap that would put special keyboard modifications in place, such as foreign-language mappings, that the user would need to type in his name and password.

The ViewStation has a corresponding file, the server-reset file, that allows this kind of configuration. The ViewStation also has a local xmodmap client, refer to Section 2.6.13, which can be used for this purpose.

3.3.2.5 Xstartup File

The Xstartup file contains startup commands that should be executed AFTER the user has logged in and is verified. This file is typically empty.

3.3.2.6 Xreset File

The Xreset file contains commands that should be executed after the user's login has been verified.

3.3.2.7 xdm-errors File

The xdm-errors file contains error messages relative to Xdm operation. If you have problems with Xdm, look in this file.

If you have more extensive Xdm problems, restart Xdm with the -debug argument set (with a fairly high value, say 30). This will give extensive error reporting for the Xdm process.

3.3.2.8 xdm-pid

The xdm-pid file stores the process ID for the parent Xdm process. This is sometimes an important value, since each Xdm connection spawns a daughter process and it is difficult to determine which is the parent process.

There is a resource in the Xresource file which permits changing this file's name.

3.3.2.9 Xsession file

The Xsession file (as distinct from the .xsession file) is the system's default startup script. If there is no user's .xsession file, this .Xsession file will be used.

In the standard MIT release, this file starts an xterm and the twm window manager. Of course, your system administrator can modify this script.

3.3.3 Xdm Versions

There are a number of different versions of Xdm. These are substantially different depending on their chronology, whether X11R3, R4, or R5. There are also some differences in vendor releases, though these differences are generally minor.

If you have problems with your Xdm process, it may be helpful to look at its version and operation. Look at the Xdm man pages on your system.

In summary:

X11R3 selected hosts to manage by looking at the Xservers file. This management was set by the system administrator and had no user selectivity.

X11R4 changed the functionality to permit the user's selection of Xdm management and the method of making the connection. This was called XDMCP.

X11R5 improved Xdm control of the management process and added an Xdm "Chooser" client to aid in selecting an Xdm host.

3.3.4 Using Xdm with your ViewStation

Using Xdm with the ViewStation is simple. There are only two requirements; first, that Xdm be running on your host, and second, that you request Xdm access from the XDMCP menu of Setup Mode.

There are several methods of Xdm access.

3.3.4.1 Xdm Direct Access

In this case you select the Xdm Access as Direct and specify a host address for the Xdm host. The ViewStation sends its Xdm request directly to that host and an Xdm session is established.

3.3.4.2 Xdm Indirect Access

Indirect access for Xdm means that the ViewStation can connect to one of several host computers, with one computer serving as a redirection agent for the system. Typically a chooser box appears with a fixed list of Xdm hosts. The user's selection is then forwarded to that host.

3.3.4.3 Xdm Broadcast Access

When Broadcast is selected as the access method, the ViewStation broadcasts an Xdm request. All Xdm hosts that respond to the broadcast are put into the Chooser box and the user can make the selection of the Xdm host.

3.3.4.4 Xdm Chooser

Xdm, with version X11R5, has a Chooser client. This is a selection box that lists the Xdm hosts that are available, generally from an Indirect or Broadcast Xdm access selection.

3.3.4.5 ViewStation's Chooser Client

The ViewStation has a local Chooser client. The ViewStation's Chooser works the same way as the host-based chooser, but it has two important differences. The ViewStation's chooser works with both X11R4 and X11R5 Xdm processes and it allows the user to enter an Xdm host into its selection box. These are often very convenient features.

You can start the ViewStation's Chooser client from the XDMCP menu in Setup Mode. Its operation is described fully in Section 2.6.4.

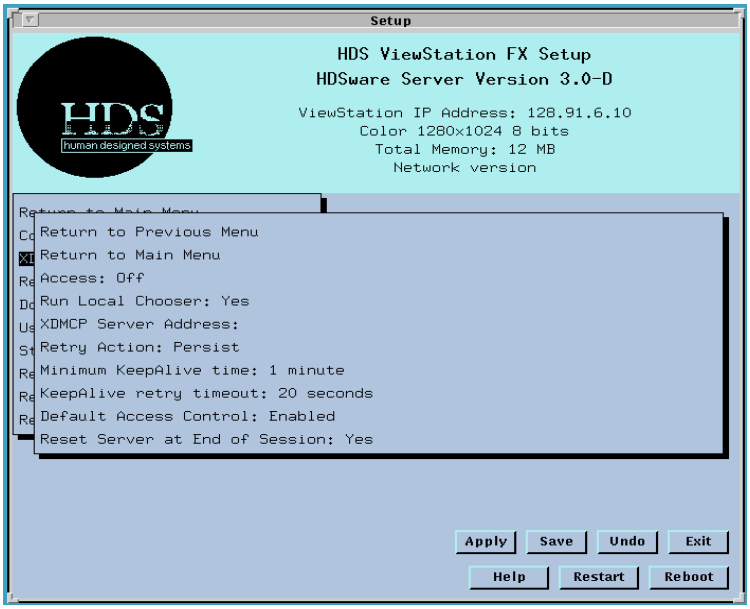
3.3.5 Using XDM on the ViewStation

The ViewStation will use Xdm if you select Xdm access on the XDMCP Options menu of Setup Mode. With an access method selected, the ViewStation sends an Xdm request each time it starts (or the server resets). Response to this request by the Xdm process begins Xdm's control the ViewStation's display. Note that any processes running on the ViewStation (local clients, including Setup Mode) will be killed by Xdm and won't be available until the Xdm login is complete.

3.3.5.1 XDMCP Options Selection in Setup Mode

XDMCP (X Display Manager Control Protocol) is a host-based handshaking protocol to establish a connection with a host and to manage your display. It works with a host client to control your logon process, start the clients you specify, and set up configuration parameters whenever you turn on your ViewStation.

The XDMCP Options Menu is shown below:



The **Access Type** field specifies whether you use xdmcp and which host you want to have manage your display. There are five values for this field:

Off (default value) specifies that you do not want to use XDMCP.

Broadcast specifies that you want to accept an offer to manage your display from the first host that answers your broadcast request (your IP Broadcast Address is set in the Communications Options Menu). You can use the local HDS Chooser client to give you a menu of available hosts.

Direct specifies that you want to communicate only with the xdm client running on the host shown in the XDMCP Server Address field.

Indirect specifies that you want the host given by the XDMCP Server Address field to act as an Indirect host. This Indirect host usually forwards the request to another host computer. You can also use the local HDS Chooser client to give you a menu of available hosts.

Hosts access method specifies a file of connection options in a host file. The "Hosts" access method allows you to specify multiple hosts with individual access methods for your XDMCP request. You can set up an Xdm Hosts file by selecting the Hosts access method and entering the path to your Xdm hosts file in the "Hosts File" line. A sample file is given below:

The Xdm hosts file contains a list of hosts, one per line. Each host entry can be one of three XDMCP query types - Direct, Indirect, or Broadcast. Host names starting with an asterisk ('*') are queried via Indirect requests. Host names not starting with an asterisk are queried via Direct requests. A host name of 'BROADCAST' tells the ViewStation to perform a broadcast request. Each host entry also has an associated retry timeout. The timeout parameter specifies how many seconds to send requests to a host before trying the next host in the list. The default timeout is 0 seconds. Hosts are tried in the order in which they appear in the hosts file.

Characters after a '#' character are comments.

This is a sample XDMCP hosts file:

```
# XDMCP hosts example file

*hdssun1 # Indirect request to hdssun1, timeout=0

hds486 timeout=10 # Direct request to hds486

hdshp timeout=15 # Direct request to hdshp

BROADCAST # Broadcast request
```

This file specifies that at start_time + 0 seconds, an indirect request is sent to hdssun1 and a direct request is sent to hds486. At start_time + 10 seconds, a direct request is sent to hdshp in addition to requests to hdssun1 and hds486. At start_time + 25 seconds, a broadcast request will be sent in addition to requests to hdssun1, hds486, and hdshp.

The **Run Local Chooser** field lets you select whether you run the HDS ViewStation's local Chooser client. When you use the Indirect or Broadcast access for Xdm, this client puts up a menu of available Xdm hosts. It tests the network to see which hosts respond to your Xdm request and then lets you select the one you want. This is a client running on the ViewStation. It is described in detail in Section 2.6.4. It works with both R4 and R5 Xdm versions.

The **XDMCP Server Address** field shows the IP address of the Direct or Indirect Access host shown in the previous field. The default value is 0.0.0.0.

The **Retry Action** field shows whether you want the ViewStation to keep trying to get the xdm display manager to respond or not. The values are Persist and Stop. If you want to use xdm, you probably want the requests to persist until you get a response.

The **Minimum KeepAlive time** field gives the minimum time the ViewStation waits after it has received its last message (of any kind) from the host computer before sending a KeepAlive packet to the display manager. This lets the ViewStation continue its session even if there is no activity from the host computer. The value is an integer value in minutes, with a default value of one minute. This means that the ViewStation sends a KeepAlive packet to the display manager after 1 minute of inactivity. It sends another after 2 minutes, another after 3 minutes, and so on. Any ViewStation activity from the keyboard or the mouse will wake up the session again and reset the KeepAlive time to 1 minute. The net effect is that your session stays alive even if you leave for the weekend.

The **KeepAlive retry timeout** field sets the time the ViewStation waits for a response from the host after it has sent its KeepAlive packet. This is an integer value in seconds, with a default value of 20 seconds. The ViewStation waits 20 seconds after its request for a response; if it doesn't receive one, it closes the connection because the display manager hasn't answered. After it sends the first request (at one minute of inactivity) it sends another request after 2 seconds, another after 4 seconds, and so on until the retry timeout value is reached.

The **Default Access Control** field sets the ViewStation's Access Control security, also called MIT Magic Cookie. The default setting is Enabled. Access control means that Xdm establishes an .Xauthority file with the initial logon. This file is used for security control of other login connections. Refer to Section 1.11.3 for more details and methods of using or modifying Access Control security.

The **Reset Server at End of Session** field selects whether the ViewStation's server software resets itself at the end of an Xdm session. The default setting is Enabled. Depending on your system configuration, this may or may not be desireable, since a server reset removes a lot of the user's settings.

These settings and timeout values should be worked out with consideration of the settings for the corresponding values for xdm on the host computer.

Note also that all of these xdm variables are saved in NVM because they are used very early in the power-up process. If you change the IP Broadcast address value, which can be changed by Apply Changes for the current session only, xdm will use the IP Broadcast address set in NVM when it starts. If you are using the Broadcast address for xdm, you should save its value in NVM each time you edit the field.

3.3.6 Using the .xsession File

There are two Xsession files, an Xsession file in the Xdm directory that is a system-wide default file and an .xsession file that is in the user's home directory. Usually the .xsession is the only file of interest to the user. The Xsession file is controlled by the system administrator and is used only if the .xsession file is not found.

The user customizes his display environment by using an individual .xsession file. This file contains a list of user commands or selections appropriate for the user's environment, such as window manager selection, clients to be run, etc.

The .xsession file is located in the user's home directory and is run when xdm starts up. (There is a general .Xsession file created by the system administrator that contains sufficient information to run the system and your display with default values without the customization of an individual .xsession file.)

A typical .xsession file might contain some entries like these:

```
xrdb -merge .Xdefaults
xmodmap -e "keycode 102 = Delete"
xclock -fg red -bg white -geom -0+0 &
xterm -e "rsh hdssun5" -fg orange -bg black &
xterm -fn 10x20 -bg blue -fg white &
twm&
xterm -title LOGIN -geometry 80x24+10+10
```

You should notice several things about this file. The first line is the command to load the resource file. The next lines start some clients, an xclock, an xterm on a remote host, an xterm with a big font, and a window manager. (Additional levels of customization can occur here, since each of the clients has an app-defaults file (found in */usr/lib/X11/app-defaults*) which can be used for individual settings, such as scroll bars for the xterms, or special emulations, etc. These customizations are often most important for the window manager's operation, which often uses a run command file, such as .twmrc, found in the user's home directory.)

The most important thing to notice about the .xsession file is that the last entry cannot run in the background. If you assign this last client to be a background process, xdm will detect the process as not active and assume you are finished and will therefore close your session. In this example, the last process is an xterm window with the label LOGIN in the title bar. This means that you can open or close any number of clients and windows in your session, but as soon as you close the LOGIN window, xdm will close your session. You could use your window manager or a client like xclock as your LOGIN window, but the same rules apply; when that window closes, the entire session is over.

You may also use the xsession file to execute shell script files, which might be used to set up and start a window manager; for instance, your shell script file might contain the various environment variables and path settings needed by the OPEN LOOK or Motif window managers. The xsession file itself must be executable, which you can do with the chmod command (**chmod 777 .xsession**).

For instance, you could create a simple shell script, called do.olwm, to start the OPEN LOOK window manager. This script might contain some environment settings and the command to start the window manager:

```
setenv OPENWINHOME /usr/openwin
```

```
setenv LD_LIBRARY_PATH $OPENWINHOME/lib
setenv X11ONLY=YES
setenv MANPATH $OPENWINHOME/share/man:/usr/man
set path=( $OPENWINHOME/bin $OPENWINHOME/bin/xview $path )
echo "Starting OPEN LOOK Window Manager"
echo "OLWM DISPLAY = $DISPLAY"
$OPENWINHOME/bin/olwm &
```

This example uses simple settings and "default" paths and directories; your installation may use different locations and directories. This shell script must be executable and should be in your home directory. Your entry in the .xsession file would be **do.olwm** instead of the **twm&** command in the previous example.

3.3.6.1 Vendor Display Class String

If you are using Indirect or Broadcast xdm requests, you may need to enter a vendor display class string for the ViewStation in the DisplayManager.servers resource. The HDS ViewStation display class string is "HDS-ViewStationFX". The manufacturer's class string is "-Ethernet-%s" where %s is your ViewStation's Ethernet hardware address in hexadecimal form, *nn:nn:nn:nn:nn:nn*.

3.3.6.2 ViewStation .xsession File

The xsession file shown above is a very simple implementation of the Xdm startup file. In practice, your xsession file would probably contain some code to analyze the display type (for color or monochrome differences), or for display capability (HDS ViewStation displays with local clients and other devices without this capability). This could be done by node naming distinctions or by analyzing the device's vendor string.

```
....
# extract a host name from $DISPLAY
HOST=`echo $DISPLAY | sed "s/".*$//"`
# check for HDS ViewStations
# all HDS ViewStations are color units
# all HDS ViewStations use local clients
if expr $DISPLAY : "hds*"
then
xsetroot -solid "PaleGreen"
xrdb -merge $HOME/.Xdefaults-hds
xmodmap -e keymap-hds
rsh -n $HOST hdslocal mwm '&'
```

```
rsh -n $HOST hdslocal hdsoclock '&'
rsh -n $HOST hdslocal hdsmemory -iconic '&'
fi
#
# create LOGIN window
xterm -title LOGIN -geometry 80x24+10+10
....
```

These examples can be customized further as your system needs require. The principal advantage of Xdm is its management of the user's session, so the system administrator can automatically provide a complete session environment for each user.

Xrdb - X Resource Database Operations

This section describes X resources, some of their customizing capabilities, and using Xrdb to manage them.

Section 3.0 - Network and Host System Operations

3.4 X Resource Operations

Resources are variables that define a window's or a client's properties. They are important to the way you use the ViewStation because they control the way a window looks and works. They also control client keyboard mappings.

In most cases, the resources as set by the client application's designers or your system administrator are sufficient for most applications and environments. However, the many "look and feel" issues of a graphical user interface can be customized by appropriate selection of X resources.

3.4.1 X Resource Format

Resources are specific for each client and typically start with the client name, such as xterm or hdsclock. The next entries in the resource line are typically some component of the client, such as scrollbar, backgroundcolor, or title. These entries can be several levels deep, such as scrollbarLocation or scrollbarWidth. Each of these variables then has an associated value, either true / false or some entry like a color or a font name.

Some examples of resources are:

xterm*scrollbar: true

xterm*scrollbarLocation: left

xterm*background: AntiqueWhite

3.4.1.1 X Resource Syntax

The syntax of these resources is very specific and you should be very careful with making entries. The man pages for each client have lists of resources available for that client and will show you the exact resource name and its permitted values.

Note that there may be other resources not specifically mentioned there, but are included as part of the toolkit which was used to build the client. For instance, if a client (such as HDSterm) uses the Motif toolkit for its GUI, it will support Motif resources in addition to its own listed resources.

3.4.1.2 Resource Classes and Resource Instances

The terms Class and Instance indicate the generality of a resource. A Class may have several Instances. By convention, class names use an uppercase letter as their first character and instance names use a lowercase letter. For example:

xterm*background: AntiqueWhite

xterm*Background: PaleGreen

are instance and class resources, respectively.

A resource instance is always associated with a class. Usually you can tell what the association is because the names are similar. However, there are cases where the names may not appear to have any connection, for instance in xterm, the resource cursorColor belongs to the class Foreground (because the cursor is part of the foreground).

3.4.1.3 Class and Instance Hierarchy

There is a hierarchy of execution for classes and instances. Resource instance settings override resource class settings. This may seem backward, since users often think that classes are more powerful than instances. It is more intuitive to think of classes as "general" settings and instances as "specific" settings.

This mechanism allows client and user settings to work together. If you, as a user, specify a resource with a class name, and an application program specifies an resource instance, the application's value will be used. In this way, specific settings are preserved and preferences honored if possible.

For example, suppose you had set the Instance resource:

xterm*foreground: AntiqueWhite

Because this is a resource instance, it will apply to the foreground text only and not to other instances of the Foreground class (such as cursorColor).

If you set the Class resource:

xterm*Foreground: AntiqueWhite

it would apply to all instances of that class (foreground text and cursorColor).

If you set both Class and Instance resources:

xterm*foreground: AntiqueWhite

xterm*Foreground: PaleGreen

the result will be a foreground text color of AntiqueWhite and a cursor color of PaleGreen.

3.4.1.4 Resource Bindings

In the hierarchy of resources, there can be many levels of variables. The number of levels and their connections are often unknown so resource specifications can be entered with a kind of "wildcard" (called a loose binding) to cover unknown levels. Variables in the resource are linked with "*" and "." for loose and tight bindings respectively.

In most resource specifications that a user would use and change, a "loose" binding using the "*" is appropriate. This is a general specification of the resource hierarchy and indicates that any number of intermediate steps may be present.

A "tight" binding uses the "." to link levels of the hierarchy. This tight binding is used when there must be an exact match of the specified resource and the widget hierarchy. Tight bindings override loose bindings.

From a user's perspective, loose bindings are preferred, both from their ease of use and their low level of priority. Any application program that needs a particular resource setting will use a tight binding to insure that it is in place and the user's settings should not interfere. For example, you might use a loose binding to select a large font for an application window but the application, by using tight bindings, would continue to use its own font sizes for the window's buttons and menus.

3.4.1.5 Resource Precedence

There are several rules for using resources:

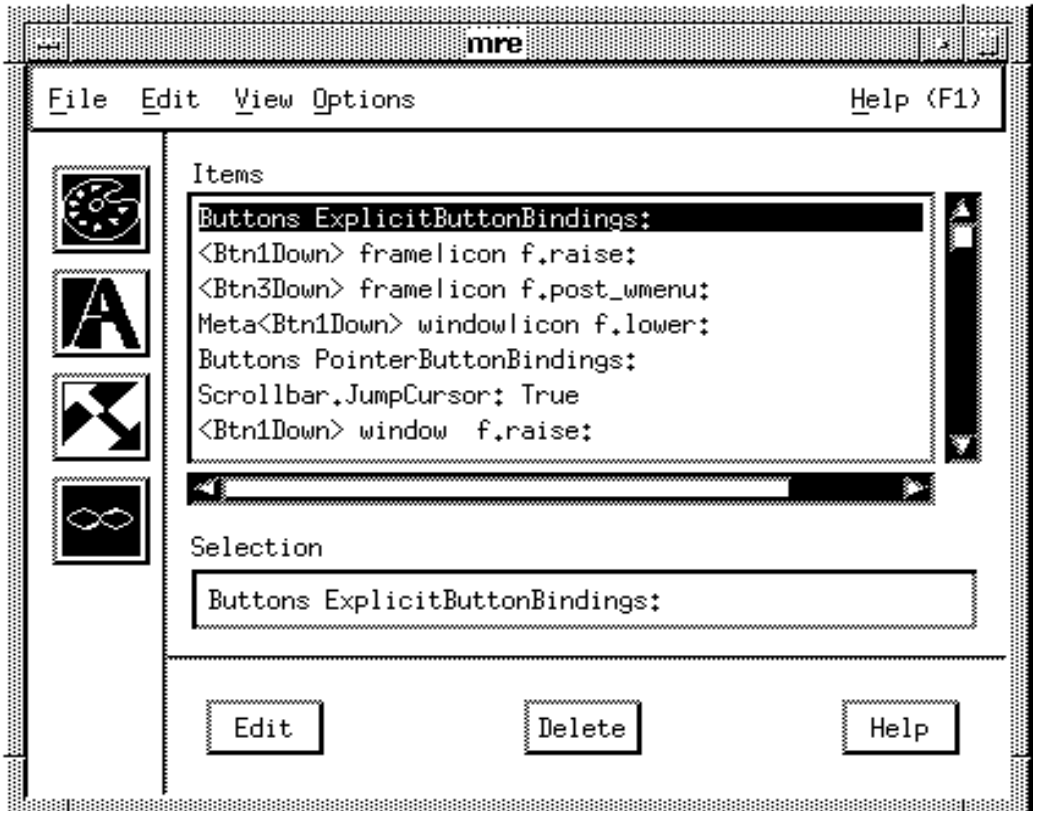
- Most recent entries override older entries. This means you can append a resource to the end of a resource file and it will override earlier entries.
- Instance specifications take precedence over class specifications.
- Explicit specifications take precedence over partial specifications.
- Tight bindings take precedence over loose bindings.

3.4.1.6 X Client - editres

The editres client is included with the MIT X distribution. It displays the client's resource hierarchy and lets you see what the names and connections are. Note that editres may not work smoothly with X11R4 Motif clients. It works better with X11R5 and more recent Motif releases.

3.4.1.7 X Client - mre

The Motif resource editor is called mre. This 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.



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 above.

3.4.2 X Resource Database

The resources you've set are put into a file and read by a resource manager on demand. There are several locations for these resource files, depending on their function.

3.4.2.1 The X Resource File

The X resource file itself is simply a list of resources indicating the user's preferences for various applications, usually located in the user's home directory.

Another resource file can also be an app-defaults file, which has resources specific to one client only and is generally located in a system-wide directory such as */usr/lib/X11/app-defaults*.

A resource file can also be an .Xdefaults file and it will typically have resources for many different clients.

A resource file can also be an .rc file, such as .mwmrc, usually in a user's home directory.

Each of these file types is discussed below. It is best to keep resources together in a consistent location. The actual location is not too important, but you must be sure the file is read when you start up.

A typical .Xdefaults file might contain:

```
!
!X Resource specifications
!
xterm*scrollbar: true
xterm*background: SkyBlue
xterm*saveLines: 300
XTerm*Curses: true
XTerm*JumpScroll: true
XTerm*MultiScroll: true

xmanCommands*Geometry: 132x232+0+0
xmanSections*Geometry: 156x255+0+0
xmanSearch*Geometry: 316x48+0+0

Mwm*keyboardFocusPolicy: pointer
Mwm.autoClientMenu: false
Mwm.autoClientMenuName: HDS Clients
Mwm*buttonBindings: PointerButtonBindings
Mwm*windowMenu: WindowMenu
Mwm*keyBindings: DefaultKeyBindings
Mwm*fadeNormalIcon: True

HDSAudio*background: IndianRed1

HDSTerm*screen.SaveLines: 300
HDSTerm*statusLine: false
HDSTerm*bitmapDirectory: /HDS$$ROM
```

```
!HDSTerm*screen.font: -adobe-courier-medium-r-normal--14-*

HDSview*background: aquamarine3
HDSview*toc.visibleItemCount: 40
HDSview*locator.width: 100

HDSDClock*foreground: red
HDSDClock*background: black
HDSDClock*geometry: 220x65-5+5
HDSDClock*date: %W, %M %d
HDSDClock*font: 10x20
HDSDClock*tails: Yes

HDSCalc*foreground: LightSkyBlue1
HDSCalc*background: SkyBlue4
```

This sample file indicates some of the resources you might use for some clients and the way they are all used together. Note the different syntax for the resources depending on each client. Note also that trailing spaces will create an error, so be careful how you write the file.

3.4.2.2 Application Resource Files

Each application program has resources set in its own resource file. This is the "applications-default" file, or app-defaults. For X clients, this is located in */usr/lib/X11/app-defaults*. For other applications, the app-defaults file is sometimes in the same directory as the executable file. For OpenWindows, it is in *$OPENWINHOME/lib/app-defaults*.

There are also environment specifications XUSERFILESEARCHPATH and XAPPLRESDIR that specify a directory for these files. These environments may or may not be set on your host. If XAPPLRESDIR has not been defined, the application will look in the user's home directory for a resource file with the application's class name.

3.4.2.3 Run Command Files for Resources

The UNIX method of entering these settings is with a run command file, or .rc file. This file is also used as a reference location for resources. These files can contain resources, and other variables, for each application. It is advisable to locate all the appropriate resources in a single file to simplify your system control and to insure that the files are read.

The .rc file, such as .mwmrc for the Motif Window Manager, is an important configuration location. The MWM resource specifies this file location:

Mwm*configFile: /<home directory>/.mwmrc

3.4.2.4 Server-related Resource Files

Resources for server-related settings are typically in the user's home directory in a file called .Xdefaults. This file can contain resources for all clients which run on the ViewStation server, regardless of where the clients are executed.

3.4.2.5 Host-related Resource Files

Host-related resources give you a way to make configuration adjustments based on the host executing the application. The resource file in this case is in the user's home directory with the name .Xdefaults-<hostname>.

3.4.2.6 Resource Specification from the Command Line

You can set a resource from the command line using the -xrm argument. This is often useful for test purposes. For example:

xterm -xrm '*background: LightSteelBlue' &

3.4.2.7 ViewStation Local Client Resources

The HDSperx clients have resources that can be set in the normal way, typically in the .Xdefaults file. In addition, there are special client attributes which are not conventional resources, but special properties that are unique to ViewStation operations; these resources are set in the ViewStation's clients.config file.

These attributes are things like paths and environments, as well as password protection, memory requirements, etc. These are discussed in detail in Section 2.1.5.

3.4.3 Loading Resource Files with Xrdb

Because there are so many application programs and so many places for resource files, the automatic reading and setting of resources for applications can be an uncertain process. You should verify that each application finds the correct resource files when it starts.

You can also manage your resources and ensure that they are read by explicitly reading them with xrdb, the X resource database manager. Xrdb is a host-based X utility that you can run in a startup file, like .xsession. The command line is:

xrdb -merge $HOME/.Xdefaults

This specifically merges the resources in the .Xdefaults file with resources from other locations, like app-defaults or .mwmrc.

Network Printing Operations

This section describes the HDS ViewStation's Print Spooler and Print Model, host-based scripts to enable network printing to the ViewStation. Sample files for both BSD and System V UNIX systems are included.

Section 3.0 - Network and Host System Operations

3.5 Network Printing Operations

The HDS ViewStation is designed as a network device with full operational capability for network printing. This is managed by host-based programs, called the Print Spooler or LP Print Service, which permits the host's print capability to manage the ViewStation and its ports, including the parallel port and both serial ports.

Network printing is handled by a "print spooler" on UNIX BSD systems and a "lp print service" on UNIX System V systems. Both are supported and are described, with sample files, in the material below.

These scripts and sample files are found in the HDSware distribution in the */hds-fx/config/lpd-support* and */hds-fx/config/lp-support* directories.

3.5.1 The LPD Print Spooler for BSD Systems

HDS has generated a sample print spooler that lets you take advantage of the local printing capability of the ViewStation. Printing is generally a system-wide function, so these files will have to be modified or adapted to your particular situation. This print spooler is intended as a sample only. Your system may be different, or use different commands or syntax, so you will have to make the necessary adjustments.

3.5.1.1 LPD Daemon

On BSD UNIX systems, print spooling is handled by a daemon process known as LPD. The shell command to print a file is usually "lp" or "lpr", which sends mes-

sages to the LPD daemon indicating the names of the files to print, the number of copies, etc. The LPD daemon process is then responsible for opening the path to the printer requested, passing the file through various filters, maintaining the print queue, etc.

The LPD process is started by the system administrator. At startup, it reads a file called */etc/printcap* which lists the printers available to the LPD daemon and whether those printers are local or remote. If the printer is a remote printer, then transfering the print file also involves the LPD on the local host communicating with the remote host's LPD. In this case, the remote host is responsible for temporarily storing the print file in its local file system until it can be sent to the printer.

Since the HDS ViewStation may not have any local disk memory for this temporary storage, one host's LPD should store and maintain the print job queues for the printer attached to the ViewStation. To make this arrangement, the system administrator should configure the ViewStation's printer as if it were locally connected to that controlling host. This is done by adding an entry into the controlling host's */etc/printcap* file. A sample printcap file entry is included in this LPD directory.

The key to making this arrangement work is the use of an input filter, such as the hds-spool.sh file included in this directory. This input filter is specified in the */etc/printcap* file as "if=<filter pathname>". The LPD daemon writes the contents of the print file to the standard input of this filter. The filter then performs several tasks; it opens a remote shell on the ViewStation and transfers the print file to an "opendev" task, which routes the data to the device specified on the opendev command line. It also logs any errors generated to the filter's standard error channel and appends them to the log file (which was specified in the */etc/printcap* entry "lf=<log file pathname>".

On a system with multiple ViewStations, you would make a copy of the input filter, hds-spool.sh, each with its own name and variables, for each ViewStation. Each ViewStation would also have its own printcap file.

3.5.1.2 Sample Input Filter File

The sample input filter is shown below. The script is "hds-spool.sh" and is written as a Bourne shell script, which may be customized as needed. As it appears here, it is hardcoded to send its output to a specific ViewStation hostname (or IP address) and to a specific hardware port. (The ViewStation can connect to printers on its parallel port or to printers on its two serial ports.) If you want to set up a system with a single host controlling printers connected to more than one ViewStation, you will have to create separate */etc/printcap* entries for each printer, and the input filter script specified for each printer must be modified so it connects to the correct ViewStation and hardware port on the ViewStation.

Note that the ViewStation will not abort the print job if the printer is off-line or out-of-paper. Instead, the ViewStation will stop accepting data on its "shell" TCP port, which in turn will cause the lpd process executing the input filter shell on the controlling host to block until the ViewStation resumes accepting data. Print

jobs may be aborted either by "killing" the opendev process on the ViewStation, or by using "lpc" or other print spooler administration tool on the controlling host.

3.5.1.3 Spooler Shell Script

This is the hds-spool.sh script. It is written as a Bourne shell script to let the host's LPD process to write to an HDS ViewStation. It is a sample input filter which you need to customize. Each ViewStation with a printer should have one of these files, with a unique name, with its variables set appropriately.

```
# !/bin/sh
# hds-spool.sh: Sample input filter to allow lpd to print to
# an HDS ViewStation
# Copyright 1992, 1993, 1994, 1995 HDS Network Systems, Inc.
# All rights reserved.
# SCCSID = "@(#)hds-spool.sh 1.3 6/22/93"
PATH=/bin:/usr/bin:/usr/etc:/usr/ucb; export PATH
TMP=/tmp
PROG=$0
OPENDEV=opendev
# Set RSH_CMD as needed for your system (RSH, RCMD, or REMSH, etc.)
RSH_CMD=/usr/ucb/rsh
# Set DEV=/dev/pp0, DEV=/dev/tty1, or DEV=/dev/tty2 for
# ViewStation FX models;
# Set DEV=printer for older ViewStation models
DEV=/dev/pp0
# Set     xterm=<hostname of ViewStation>
xterm=hdsfx
ABORT=2
name=unknown
host=unknown
while [ $# -gt 0 ]
do
    case "$1" in
     -w*)
        ;;
     -l*)
        ;;
     -i*)
        ;;
     -n)
    if [ $# -gt 1 ]
    then
```

```
        shift
         name="$1"
    fi
        ;;
     -h)
    if [ $# -gt 1 ]
    then
        shift
         host="$1"
    fi
        ;;
 -*)
        echo $PROG: Filter flag $1 not supported. >&2
        exit $ABORT
        ;;
    esac
    shift
done
# Extract base name of this shell program
OLDIFS="$IFS"
IFS=/
set $PROG
IFS="$OLDIFS"
PROG=`eval echo \\\$$#`
status=0
trap "rm -f $TMP/tfile$$; rm -f $TMP/tlog$$; exit \$status" 0
cat - > $TMP/tfile$$
status=0
finished=0
while [ "$finished" -eq 0 ]
do
    $RSH_CMD $xterm $OPENDEV $DEV < $TMP/tfile$$ > $TMP
    /tlog$$ 2>&1
 status=$?
    if fgrep 'opendev' $TMP/tlog$$ > /dev/null
        then
            if fgrep 'EBUSY' $TMP/tlog$$ > /dev/null
            then
            sleep 10
            else
            echo "`date` $PROG: `fgrep 'opendev' $TMP/
            tlog$$`" >&2
            status=$ABORT
```

```
            fi
         else
            finished=1
      fi
   if [ "$status" -gt 0 ]
      then
         status=$ABORT
         finished=1
         echo "`date` $PROG: Aborted print job for ${name}@${
         host}">&2
      fi
done
exit $status
```

3.5.1.4 Printcap File

This is a sample printcap file. You need to customize this for your environment.

```
#
# This is a sample of printcap entries used to print to HDS
# ViewStations.
#
# The spool directory must be created in /usr/spool/, and you
# must also "touch" the "lp" file for each printer attached
# to a ViewStation. The "lp" file is used simply to lock each
# printer during printing. The input filter "if" must be
# unique for each separate printer, since it specifies which
# ViewStation to print to. You could get fancy and embed the
# ViewStation name in the input filter name, and use links to
# a single input filter, but this seems overkill unless you
# plan to support very many printers connected to
# ViewStations.
#
# SCCSID = "@(#)printcap 1.2 6/22/93"
#

1|hdsVS-1|Test of printer spooler to HDS ViewStation:\
 :sd=/usr/spool/hdsVS:lf=/usr/adm/lpd-errs:\
 :if=/usr/local/bin/hds-spool.sh:\
 :sh:lp=/dev/hdsVS-1:
#
```

There are several things to note:

• The "hdsVS-1" on the first line is the network name of the printer. It is the destination name you give for your print job. It should probably be the printer name

(such as PS, or LJ) rather than a name associated with the ViewStation, unless there are multiple printers of that type connected to ViewStations.

• The ":sd" entry is the spool directory and contains the printer's filter file */usr/spool/hdsVS*; again, its name should reflect the printer name or type rather than the ViewStation name.

• The ":lp" entry on the last line is the network name of the printer. This line acts as a "lock" mechanism to see if the device is currently in use. You have to create this file with the "touch" command when you add this entry to your printcap file (do "touch /dev/hdsVS-1").

• The ":if" entry is the input filter path and name for the ViewStation's input filter, which specifies the ViewStation's name and the port for the printer connection. Make sure this path and name are correct for each ViewStation and printer.

• The file reads as if it were a single line, so use the backslash characters to end lines. The ":" character separates the variables and their values from each other.

3.5.1.5 ViewStation Print Spooler Example

The following example shows how these files should be customized for a ViewStation printing on a network. Both the input filter file and the printcap file need to be modified.

Modifying the Input Filter File

1. You must have a separate input filter file for each ViewStation with a printer, or even a separate file for each port used for printing on a single ViewStation. Name these files in a way convenient for your system, such as:

   hds64-spool.sh
   hds65-spool.sh

2. Enter the ViewStation's name (not IP address) as the "xterm" variable.

3. Enter the ViewStation's printer port as the "DEV" variable.

4. If necessary, change the RSH_CMD command line to reflect your system's remote shell command (rcmd for SCO systems, for instance).

Modifying the Printcap File

You must make an entry in your printcap file (usually */etc/printcap*) for each network printer. You must make these changes to the sample printcap entry:

1. Enter the network printer name for the printer attached to the ViewStation.

2. Enter the input filter filename and path you created for the ViewStation on the "if=" line in the printcap entry.

Sample Printcap File

If your ViewStation's name was "nancyfx" and you wanted the printer's network name to be "HP-LJ", and you had named the input file with the nancyfx settings in it "hds65-spool.sh", you would make these modifications to the printcap entry.

```
1|HP-LJ|Test of printer spooler to HP-LJ on ViewStation65:\
:sd=/usr/spool/HP-LJ:lf=/usr/adm/lpd-errs:\
```

```
:if=/usr/local/bin/hds65-spool.sh:\
:sh:lp=/dev/HP-LJ:
```

The hds65-spool.sh script would have variables "DEV=/dev/pp0" and "xterm=nancyfx" set.

Add this entry to your */etc/printcap* file with other network printer entries. Remember to create the :lp file (by "touch /dev/HP-LJ") when you add the entry.

3.5.2 The LP Print Service for UNIX System V Printing

Printing with the ViewStation on UNIX System V computers uses a slightly different process and different files, but the procedure is very similar. Descriptions and sample files for LP Print Service are distributed with HDSware software (Version 3.0.3 or higher) in the */hds-fx/config/lp-support* directory.

3.5.2.1 Using LP Print Service with the ViewStation

HDS has generated a sample printer model that lets you take advantage of the local printing capability of the ViewStation. Printing is generally a system-wide function, so this file will have to be modified or adapted to your particular situation. This printer model is intended as a sample only.

Your system may be different, use different commands or syntax, so you will have to make the necessary adjustments.

On UNIX System V systems, printing support is handled by the LP print service. The shell command to print a file is usually "lp". The LP print service is responsible for opening the path to the printer requested, passing the file through various filters, maintaining the print queue, etc.

The LP print service supports both local and remote printers. If a printer is a remote printer, then the print file is sent to the remote printer host's print service. In this case, the remote host is responsible for temporarily storing the print file in its local file system until it can be sent to the printer.

Since the HDS ViewStation may not have any local disk memory for this temporary storage, one host's LP print service should store and maintain the print job queues for the printer attached to the ViewStation. To make this arrangement, the system administrator should configure the ViewStation's printer as if it were locally connected to that controlling host. This is done by adding a new printer model to the controlling host's *lp/model* directory.

3.5.2.2 Sample Printer Model File

A sample printer model file entry, hds-model.sh, is included in this */hds-fx/config/lp-support* directory.

The key to making this arrangement work is the use of a printer model, such as the hds-model.sh file included in this directory. This HDS printer model is specified in the "lpadmin" command when adding a new printer or modifying an existing printer. When printing a file, the LP print service calls the HDS printer model script with multiple arguments, including the file names to print. The HDS printer model calls the normal printer model (configured by the user) and cap-

tures the output. This output is then transferred to the ViewStation via remote shell execution of an "opendev" task. The HDS printer model logs any errors generated to the standard error channel.

The sample printer model included in this directory, "hds-model.sh", is written as a Bourne shell script, which may be customized as needed. As it appears here, it is hardcoded to send its output to a specific ViewStation hostname (or IP address) and to a specific hardware port. (The ViewStation can connect to printers on its parallel port or to printers on its two serial ports.) If you want to set up a system with a single host controlling printers connected to more than one ViewStation, you will have to create separate printer model entries for each printer.

Note that the ViewStation will not abort the print job if the printer is off-line or out-of-paper. Instead, the ViewStation will stop accepting data on its "shell" TCP port, which in turn will cause the LP process executing the printer model script on the controlling host to block until the ViewStation resumes accepting data. Print jobs may be aborted either by "killing" the opendev process on the ViewStation, or by using print spooler administration tools on the controlling host.

3.5.2.3 Adding a ViewStation Printer to the LP Print Service

Follow these steps to add a ViewStation printer to your LP Print Service.

• Attach the printer to the desired ViewStation's serial or parallel port. If using a serial printer, make sure the communications settings and cable are correct.

• Make sure there is a host name corresponding to the ViewStation's IP address in the host's IP database. (For example, */etc/hosts*)

• Choose a name for the printer. This probably should be the same as the host name of the ViewStation, but need not be.

For example, your ViewStation name might be: hdsfx1

• Copy and rename the hds-model.sh script to the *lp/model* directory. There should be a separate copy and name for each ViewStation printer. Set the owner and execute permission on each model script.

<path> is the path to the *lp/model* directory, usually */usr/spool/lp/model*

```
cp hds-model.sh <path>/<printername>
cd <path>
chown lp <printername>
chmod +x <printername>
```

For example:

```
cp hds-model.sh /usr/spool/lp/model/hdsfx1
cd /usr/spool/lp/model
chown lp hdsfx1
chmod +x hdsfx1
```

• Set the X terminal name (XTERM), the parallel or serial port (DEV), and the printer model (MODEL) variables for the printer in each of the model scripts. Also set the path to the remote shell command (RSH_CMD) and whether or not

NEWLINE characters should be replaced with CARRIAGE RETURN / NEWLINE characters (NFILTER). All of these configurations are at the beginning of the model script.

For example:

```
RSH_CMD=/usr/bin/remsh
DEV=/dev/pp0
XTERM=mshfx
MODEL=/usr/spool/lp/model/HPDeskJet500
NFILTER="cat -"
```

• Add the printer with the 'lpadmin' command. Specify the printer device as '/dev/null' using the '-v' option. Specify the model as <printername> using the '-m' option. Specify the printer as <printername> using the '-p' option. Also specify any other desired options, such as '-o nobanner'.

Check the 'lpadmin' manual page for supported options and meanings, as well as spacing and syntax.

For example:

```
/usr/lib/lpadmin -p hdsfx1 -v /dev/null -m hdsfx1
```

• Start the new printer with the "accept" and "enable" commands.

For example:

```
accept hdsfx1
enable hdsfx1
```

• Print some test files to the printer.

For example:

```
lp -d hdsfx1 /etc/motd
```

You can monitor the printing with the ViewStation's local Console 'ports' command.

3.5.2.4 HDS-Model File

This is a copy of the hds-model.sh file.

```
#!/bin/sh
#####
# hds-model.sh: Sample model file for use with System V printing
# facilities to allow ''lp'' to print to a device connected to an
# HDS ViewStation's parallel or serial ports.
# This script is a wrapper that calls the normal printer 'model'
# script and then sends the output to the ViewStation.
#####
# Copyright (C) 1994, 1995 HDS Network Systems, Inc. All rights
# reserved.
# ident: SCCSID = "@(#)hds-model.sh 1.1 6/9/94"
#####
PATH=/bin:/usr/bin:/usr/etc; export PATH
```

```
TMP=/tmp
OPENDEV=opendev
#RSH_CMD=/usr/bin/rsh
RSH_CMD=/usr/bin/remsh
#####
# Set DEV=/dev/pp0 (parallel port)
# DEV=/dev/tty1 (serial port 1)
# DEV=/dev/tty2 (serial port 2)
# for ViewStation FX;
#
# Set DEV=printer
# for older ViewStation
#####
DEV=/dev/pp0
#####
# Set the X terminal name.
# XTERM=<ViewStation hostname>, or XTERM=<ViewStation IP address>
#####
XTERM=mshfx
#####
# Set MODEL equal to the path of the normal model script for the
# printer, e.g. MODEL=/usr/spool/lp/model/HPDeskJet500. MAKE SURE
# THE MODEL FILE ie. HPDeskJet500 is executible by everyone
#####
MODEL=/usr/spool/lp/model/standard
NFILTER="cat -"
#####
# Uncomment the next line to map NEWLINE characters
# to RETURN-NEWLINE.
#####
#NFILTER="sed -e s/$/^M/g"
#####
# Get arguments passed to this script and save away to reconstruct
# them when calling the 'MODEL' script.
#####
PROG=$0
ARG1=$1
ARG2=$2
ARG3=$3
ARG4=$4
ARG5=$5
shift 5
FILES=$*
```

```
ABORT=2

name=unknown
host=unknown
#####
# Extract base name of this shell program
#####
OLDIFS="$IFS"
IFS=/
set $PROG
IFS="$OLDIFS"
PROG=`eval echo \\\$$#`

status=0
trap "rm -f $TMP/tfile$$; rm -f $TMP/tlog$$; exit \$status" 0

$MODEL "$ARG1" "$ARG2" "$ARG3" "$ARG4" "$ARG5" $FILES | ${NFILTER}
> $TMP/tfile$$

status=0
finished=0
while [ "$finished" -eq 0 ]
do
    $RSH_CMD $XTERM $OPENDEV $DEV < $TMP/tfile$$ > $TMP/tlog$$ 2>&1
    status=$?
    if fgrep 'opendev' $TMP/tlog$$ > /dev/null
    then
        if fgrep 'EBUSY' $TMP/tlog$$ > /dev/null
            then
               sleep 10
            else
            echo "`date` $PROG: 'fgrep 'opendev' $TMP/tlog$$`" >&2
            status=$ABORT
    fi
 else
    finished=1
fi

if [ "$status" -gt 0 ]
    then
        status=$ABORT
        finished=1
```

```
       echo "`date` $PROG: Aborted print job for
${name}@${host}">&2
    fi
done
exit $status
```

Using the ViewStation on Selected Host Systems

This chapter will help you connect an HDS ViewStation to a Sun host system.

4.0 Section 4.0 - Using the ViewStation on Selected Host Systems

4.1 Sun Workstation Configuration

Each Sun workstation will have a slightly different configuration, depending on the software present on the system and its installation. These suggestions rely on the standard configuration and file location on Sun systems.

There are two basic sections of interest:

- Sun Network Services
  - TFTP protocol
  - FTP protocol
  - NFS protocol
  - Other network services
- Sun Open Windows Applications
  - Open Windows Environment
  - OLWM - on the Sun
  - OLVWM - on the ViewStation

4.1.1 Sun Network Services

The ViewStation uses either the TFTP, FTP, or NFS file transfer protocols to load fonts and local clients, so at least one of these programs (daemons) must be present and running on the Sun. The ViewStation uses TFTP as its default file

transfer protocol. You can select another protocol by changing the entry in the Remote Hosts menu of Setup Mode.

4.1.2 TFTP File Transfer Protocol

Typical Sun installations do not start TFTP. This is the ViewStation's default protocol. A startup line for TFTP is present in the */etc/inetd.conf* file, but is usually commented out. With the comment character removed, the TFTP protocol is run in "secure" mode. If you want to run TFTP as your file transfer protocol for the ViewStation, you must verify that it is running, and then decide if you want to use it in secure mode or not.

To start your TFTP service on the Sun (you must be root user to do these steps):

1. Check to see if TFTP was started. Look in the */etc/inetd.conf* file to see if the TFTP line has a comment character. The line looks like this:

#tftp dgram udp wait root /usr/etc/in.tftpd in.tftpd -s /tftpboot

2. Remove the comment character (#) from the beginning of the line.

3. Restart the inetd process. Find the process identification number (PID) from the following command:

```
ps -aux | grep inetd
```

Then issue this command:

```
kill -1 <PID>
```

This restarts the inetd process and starts the TFTP daemon.

4.1.2.1 TFTP in Secure Mode

If the TFTP process is in secure mode (check for the "-s /tftpboot" at the end of the line), the TFTP process must change to the */tftpboot* directory to move requested files. If your files are not in that directory, or you don't want to put them there, or you want to transfer files from other directories, you will have to make some changes.

You have several options:

- **Use secure TFTP** - this means that all your files must be located in the */tftpboot* directory. This partition is usually fairly small, so check the available storage space with the "df" command. The HDSware tape requires about 21 MB for a complete installation, or about 9 MB for a minimal installation.

  Note that the Sun pre-pends */tftpboot* to whatever directory you specify in your paths, so take this into account when you enter your HDSware path.

- **Use unsecure TFTP** - remove "-s /tftpboot" from the TFTP line and restart the inet daemon. This means that TFTP is not secure and can move files from or to any directory on the system. This is convenient, but you may want to keep the security features.

- **Link your files to */tftpboot*** - for instance, you could create a tftpboot directory in your */usr* partition, */usr/tftpboot*, and link it to */tftpboot*, as follows:

```
cp -r /tftpboot /usr
rm -rf /tftpboot
ln -s /usr/tftpboot /tftpboot
```

4.1.2.2 Paths with Secure TFTP

If you use Secure TFTP, note that any TFTP request does a change directory command to the secure directory, usually */tftpboot*, and this directory is automatically prepended to your path. This means that your path should not contain the */tftpboot* directory, since it will be added automatically.

If your server code file is in */tftpboot/HDSware/hds-fx/image/Vnn.hds*, for example, your HDSware path should be entered as */HDSware* only, not */tftpboot/HDSware*.

4.1.3 FTP File Transfer Protocol

FTP is normally included with the Sun software, but it is often not started with the Sun installation; refer to the */etc/inetd.conf* file for FTP as above. You should check to see if the FTP daemon is running; check for a comment character on the FTP line in the */inetd.conf* file. The following line should be present in the */etc/inetd.conf* file.

```
ftp stream tcp nowait root /usr/etc/in.ftpd in.ftpd
```

Follow the example above if you need to restart the inetd process.

FTP requires a valid username and a real password to run, so the appropriate accounts should be set up on the Sun (normal login accounts and passwords are sufficient, but they must be entered in the ViewStation's Setup Mode Remote Hosts fields to be used for file loading).

FTP also verifies the user's shell with the *getusershell()* query to the */etc/shells* file. Sun does not create this file in normal installation. Unix System V Release 4 replaces the Bourne shell (*sh*) with the Korn shell (*ksh*) as the standard shell. If the Korn shell is not added to the */etc/shells* file, or this file doesn't exist, the query detects the Korn shell as invalid and the FTP request is denied. If you are using the Korn shell, you must make these changes.

4.1.3.1 Anonymous FTP File Transfer

Anonymous FTP file transfers do not require a password, but have other security limitations. All files to be transferred must be in a special anonymous FTP directory.

Refer to your man pages for details about setting up an anonymous FTP account. In summary, you must enter a "anonymous ftp" user in the password file (*/etc/passwd*) with appropriate group and ownership limitations, and then use that anonymous ftp user's home directory as the root directory for file transfers. Anonymous FTP file transfers initiate a chroot() call, so the new root directory is the home directory for the anonymous ftp user.

In this case, as with "secure TFTP" above (see Section 4.1.2.2), you must understand how these security mechanisms affect the entry of the HDSware path on the ViewStation. Your path on the ViewStation must take into account the new anonymous ftp root directory.

4.1.4 NFS File Transfer Protocol

NFS allows the ViewStation to perform transparent file access over the network. The ViewStation dynamically mounts the filesystem and then transfers the file; it does not use a static mount. The HDSware tape may be loaded on any filesystem on any host available to the ViewStation.

There are two ways to handle NFS processes - one for Sun OS 4.1.x (or BSD types) and one for Solaris 2.x (or Sys V R4.x types). The NFS functions are the same, but the syntax and file names are different.

For Sun OS 4.1.x and BSD type UNIX:

To use NFS, you must export the filesystems and / or directories the ViewStation requires. For example, suppose you loaded the HDSware tape in the */home/HDS* filesystem and wanted to use NFS as your file transfer protocol.

• First, verify that the */home/HDS* directory is listed in the */etc/exports* file.

• If filesystem access needs to be restricted to the ViewStation, add the ViewStation's name to the access list for the desired filesystem. For example (in the */etc/exports* file):

```
/home/HDS                    -access=hdssun5:hdsfx2
```

Thus, only hdssun5 and hdsfx2 have mount privileges to the */home/HDS* filesystem. A read-only flag may also be added for these hosts' access permission.

```
/home/HDS                    -access=hdssun5:hdsfx2, ro
```

• Check the transfer block size. The ViewStation uses a 8192 byte block size by default. If 8192 is too large for your network environment, you must set the correct value in the ViewStation's configuration file, which is */hds-fx/config/config.fx*. The ViewStation resource is:

HDSsetup.NFS.block.size: 8192 *(permitted values are 128 to 8192)*

• Finally, run the command to export the filesystem:

```
exportfs -a
```

Check your system NFS man pages for more details and arguments.

For Solaris 2.x and Sys V R4.x type UNIX:

To use NFS, you must export the filesystems and / or directories the ViewStation requires. For example, suppose you loaded the HDSware tape in the */home/HDS* filesystem.

• First, add this line to the */etc/dfs/dfstab* file.

```
share -F nfs -oro -d "For HDS X Terminals" /home/HDS
```

Note that this line sets "read-only" permission, and has a "-d" comment line argument.

• Check the transfer block size. The ViewStation uses a 8192 byte block size by default. If 8192 is too large for your network environment, you must set the correct value in the ViewStation's configuration file, which is */hds-fx/config/config.fx*. The ViewStation resource is:

HDSsetup.NFS.block.size: 8192 *(permitted values are 128 to 8192)*

• Finally, run the command to export the filesystem:

shareall

Check your system NFS man pages for more details and arguments.

4.1.5 Other Network Services

There are several other important network services you may need, depending on the starting mechanism you have chosen.

4.1.5.1 BOOTP (Boot Protocol Daemon)

The BOOTP daemon is used by the ViewStation for determining the ViewStation IP address, downloading server code, setting gateway addresses, setting passwords, etc. These are optional features, but often important in large installations. Again, the BOOTP daemon is usually present but often not started at installation. Check the */etc/inetd.conf* file to start BOOTP. If you want to use these ViewStation features, check that BOOTP is installed and running. Make the appropriate entries in the */etc/bootptab* file.

4.1.5.2 RARP (Reverse Address Resolution Protocol)

The RARP protocol may be used to determine the ViewStation IP address. This utility is generally not present on Sun systems and is replaced by NIS (Network Information Service), formerly called the Sun "Yellow Pages". Refer to the Sun documentation on the NIS utility if you want to use it, but note that the RARP files */etc/hosts* and */etc/ethers* are replaced by the NIS maps. There are Sun utilities, found in */var/yp*, to convert existing RARP */etc/hosts* and */etc/ethers* configuration files to their corresponding NIS files.

4.1.6 Open Windows Applications

Open Windows applications use special environments and configuration files for their operation, including special path names and environment variables. Use the sample file below to set the required variables for the OPEN LOOK Window Manager, **olwm**, substituting the appropriate directories and paths on your Sun host. These entries should be in the ViewStation user's *.cshrc* or *.profile* file.

4.1.6.1 Open Windows Environments

These Open Windows environment variables should be set:

```
setenv OPENWINHOME /usr/openwin
setenv LD_LIBRARY_PATH $OPENWINHOME/lib
setenv X11ONLY=YES
setenv MANPATH $OPENWINHOME/share/man:/usr/man
```

```
set path=( $OPENWIN/bin  $OPENWINHOME/bin/xview  $path )
```

On Sun workstations, X11R5 is usually built with shared libraries in order to save space. These libraries have the same names as some of the Openwin shared libraries. You may notice that running a program compiled with the X11R5 libraries from the OPEN LOOK window manager sometimes displays various Xview error messages on the Sun console indicating that there are incorrect library versions. These messages can be ignored.

The OPEN LOOK window manager can be run from the Sun host or can be run locally on the HDS ViewStation (this is an official v3.0 OPEN LOOK Virtual Window Manager so its operation is identical to the Sun based OLWM; HDSware version 3.2 uses OPEN LOOK Window Manager version 4.0).

When the environment variables are set correctly, as shown above, you can start the OPEN LOOK window manager on the Sun to display on the ViewStation. Make sure the ViewStation DISPLAY variable is correctly entered.

The OPEN LOOK Virtual Window Manager runs locally on the ViewStation to distribute the processing load so you can save Sun memory and CPU power for other applications. The OLVWM behaves identically with the OLWM except that it permits an extended screen on a Virtual Desktop. This is an extension of the real screen to a tiled array of screens. The Virtual Desktop permits the user to locate windows in an off-screen tile and move to them by clicking in the appropriate Virtual Desktop tile.

4.1.7 OLWM Window Manager

There are two ways to run OLVWM on the ViewStation.

- The first way is to click on the OLWM button in Setup Mode's Main Menu. This reads the resources and menus you have specified in the ViewStation's OLWM files. The ViewStation's local HDSperx clients are added to the root menu automatically.
- The second way is to start the ViewStation's OLWM by means of a remote shell process from the Sun host. This technique has the advantage of letting the ViewStation's OLWM read the resource and menu files as they are configured on the Sun host. This technique is discussed in Section 2.5.2.

These steps are necessary:

1. Put the ViewStation's name and IP address in the */etc/hosts* file.

2. Put the ViewStation's name in the */etc/hosts.equiv* file. Use this file to govern ViewStation access to other hosts. Alternately, you may wish to use the user's *$HOME/.rhosts* file. The ViewStation would then have the same security access and limitations as the host machine executing the initial rsh command to the ViewStation.

3. Make sure appropriate environments, paths, etc. (in the environment section above) are available in the user's *.cshrc* or *.profile* files. Paths to the appropriate resource files and the *.openwin-menu* file should also be available. The rsh command looks for these variables in that location.

4. Start the ViewStation's window manager by rsh command from the host.

rsh -n <*ViewStation name***> olvwm '&'**

If you want to use the Motif Window Manager, you can also run that on the ViewStation. All of the considerations above apply to Mwm, especially environments and resource files, including *.mwmrc*.

Start Mwm with:

rsh -n <*ViewStation name***> mwm '&'**

Note that this technique preserves system security because the ViewStation can only rsh to the host that initiated the rsh command (and hosts specifically permitted by the */hosts.equiv* file). The ViewStation cannot rsh without the proper transfer of user names, permissions, and environments by means of the initial rsh. In this way, the ViewStation has no less security than the logon user and the initial rsh host; this process simply extends or limits access to other permitted host machines.

Using the ViewStation on Selected Host Systems

This chapter will help you connect an HDS ViewStation to a Hewlett Packard host system.

Section 4.0 - Using the ViewStation on Selected Host Systems

4.2 Hewlett Packard Systems

This chapter addresses special features and requirements of Hewlett Packard systems as they relate to installing the HDS ViewStation.

4.2.1 Preparing the HP700

The installation of the HDS ViewStation will go smoothly if a few things are set up before the X terminal is started. The HPvue environment has different ways to configure each display device; those suggested below are one way to do it.

There are some things to do before starting the ViewStation.

4.2.1.1 Load the HDSware Tape on the HP host

Load the HDSware tape at a location of your choice on the HP700. Decide on a file transfer mechanism (TFTP, FTP, or NFS are available). This choice may decide where you locate the HDSware software.

If you use TFTP (the ViewStation's default protocol), note that TFTP on the HP host is running with some in secure mode restrictions (discussed below). You should load the HDSware files into a directory which is available to TFTP, usually */usr/tftpdir* or */usr/lib/HDSware* (see the examples below).

4.2.1.2 Verify the startup method

- If you start the ViewStation from flashPROM or monitor mode, you can ignore this section.

- If you are using Bootp to start the ViewStation, you will have to verify that Bootp is running on the HP host. Check to see that the bootp daemon has been enabled in the */etc/inetd.conf* file. The Bootp line in the file should be uncommented. You probably also have to start the bootp daemon (and restart it after you have modified the */etc/bootptab* file).

Next, check the */usr/adm/inetd.sec* file and add these lines, if necessary:

```
bootps allow
bootpc allow
```

Now check the */etc/bootptab* file and enter the appropriate booting information for the ViewStation.

4.2.1.3 Verify the file access method

The ViewStation uses either the TFTP, FTP, or NFS file transfer protocols to load fonts and local clients, so at least one of these programs (daemons) must be present and running on the HP workstation. The ViewStation uses TFTP as its default file transfer protocol. You can select another protocol by changing the entry in the Remote Hosts menu of Setup Mode. Note that NFS is not available for booting the ViewStation.

4.2.1.4 TFTP - Trivial File Transfer Protocol

Typical HP installations do not start the TFTP daemon. When TFTP is started, various security mechanisms limit file access. These security mechanisms are discussed below.

On HP systems there is a startup line for TFTP present in the */etc/inetd.conf* file, but it is usually commented out. With the comment character removed, the TFTP protocol must be restarted. Note that TFTP can read only from directories available through the security mechanisms.

If you want to run TFTP as your file transfer protocol for the ViewStation, you must start it or verify that it is running, and then add the appropriate HDSware directory to the TFTP access list.

To start your TFTP service on the HP and add the HDSware directory (you must be root user to do these steps):

1. Check to see if TFTP is running; check for a comment character on the TFTP line in the */etc/inetd.conf* file. The line looks like this:

```
#tftp   dgram   udp   wait   root    /etc/in.tftpd    tftpd\
      /usr/lib/uxinstlf.700\
      /usr/lib/uxinstkern.700
```

2. Remove the comment character (#) from the beginning of the line.

3. Add the appropriate HDSware directory (in this case: */usr/lib/HDSware*) to the TFTP access list.

```
tftp   dgram   udp   wait   root    /etc/in.tftpd    tftpd\
      /usr/lib/uxinstlf.700\
```

```
        /usr/lib/uxinstkern.700\
        /usr/lib/HDSware
```

Note the backslash character as a designator of line continuation.

4. Restart the inetd process. Find the process identification number (PID) from the following command:

```
ps -aux | grep inetd
```

Then issue this command:

```
kill -1 <PID>
```

This restarts the inetd process which starts the TFTP daemon.

4.2.1.5 TFTP in "Secure Mode" on HP Systems

TFTP on HP / UX systems operates with security limitations. Access to the file system is limited by the TFTP configuration. There are three possibilities for setting the TFTP security.

1. File access is limited by the filenames or directories specified on the TFTP startup line in */etc/inetd.conf*, as described in the previous section.

```
tftp   dgram   udp   wait   root    /etc/in.tftpd    tftpd\
      /usr/lib/uxinstlf.700\
      /usr/lib/uxinstkern.700\
      /usr/lib/HDSware
```

In this case, TFTP file transfers are permitted for the two .700 files and any files in the */usr/lib/HDSware* directory. You could add other file names or directories as needed. Symbolic links are permitted in this path.

2. File access is limited to the home directory of the pseudo-user "tftp". In general, the "tftp" user is identified in the */etc/passwd* file as follows:

```
tftp:*:510:guest:tftp server:/usr/tftpdir:/bin/false
```

where tftp is the username, * is the password, guest is the membership, */usr/tftpdir* is the home directory, and the default shell is bogus since there is never a login to the tftp user's shell.

A TFTP file transfer initiates a chroot() call so it changes its root directory to the home directory of the tftp pseudo-user, which is */usr/tftpdir* in this case. All files transferred must be in this directory, and the file paths you enter must take this into account. Symbolic links are not permitted in this directory. This security mechanism for TFTP simulates the behavior of "TFTP secure mode" found in other operating systems.

3. You can specify both of these TFTP security variables. When you request a TFTP file transfer, TFTP first looks for the file using its home directory (which is */usr/tftpdir* in this example) as the root directory of its search. If the requested file is not found, it searches from the "root" directory specified in the TFTP startup line in */etc/inetd.conf*, which in the example above is */usr/lib/HDSware*.

Check the man pages on your HP system for the TFTP arguments and options for your system. Check the relevant files to see what the current TFTP configuration is on your system.

4.2.1.6 ViewStation File Path Specifications

The important thing for ViewStation configuration is that these security mechanisms affect how you specify the HDSware path for the ViewStation. Since TFTP changes its root directory based on the security mechanisms in place, this has the effect of prepending those secure directory paths to the pathname you enter on the ViewStation.

For example, if your HDSware is located in */usr/lib/X11/HDS/HDSware*, and the TFTP secure directory is */usr/lib*, you must specify your ViewStation's HDSware path as */X11/HDS/HDSware*, since TFTP will prepend its secure directory to the path for the file search.

4.2.1.7 FTP - File Transfer Protocol

FTP is normally included with the HP/UX software, but it is often not started with the HP installation; refer to the */etc/inetd.conf* file for the FTP line as above. You should check to see if the FTP daemon is running; look for a comment character on the FTP line in the */etc/inetd.conf* file.

```
ftp  stream  tcp  nowait  root    /etc/ftpd    ftpd -l
```

Follow the example above if you need to restart the inetd process.

FTP requires a valid username and a real password to run, so the appropriate accounts should be set up on the HP (normal login accounts and passwords are sufficient, but they must also be entered in the ViewStation's Setup Mode Remote Hosts fields to be used for FTP file loading).

FTP transfers with usernames and passwords use the root directory (/) for the search path, so full file names are needed. File access is limited by the restrictions on the user's access.

4.2.1.8 Anonymous FTP File Transfer

Anonymous FTP file transfers do not require a password, but have other security limitations. All files to be transferred must be in a special anonymous FTP directory.

Refer to your man pages for details about setting up an anonymous FTP account. In summary, you must enter a "anonymous ftp" user in the password file (*/etc/passwd*) with appropriate group and ownership limitations, and then use that anonymous ftp user's home directory as the root directory for file transfers. Anonymous FTP file transfers initiate a chroot() call, so the new root directory is the home directory for the anonymous ftp user.

In this case, as with "secure TFTP" above (see Section 4.2.1.5), you must understand how these security mechanisms affect the entry of the HDSware path on the ViewStation. Your path on the ViewStation must take into account the new anonymous ftp root directory.

4.2.1.9 NFS - Network File System

In order to use NFS on the HP system, you must export a filesystem for the ViewStation or other workstations to access. This filesystem can be exported to either the world or a selected set of hosts.

1. Add a line for each file system you want to export in the */etc/exports* file. These entries have the form:

```
<partition name> <modifiers>
or,   /usr ro
      /home ro
```

where the partition is an mountable partition and the modifiers are as listed in the the exports man page. The "ro" (read-only) option is recommended.

2. Modify the */etc/netsfsrc* file to start the NFS server:

Change

```
NFS_SERVER=0
START_MOUNTD=0
```

to:

```
NFS_SERVER=1
START_MOUNTD=1
```

3. Make sure the */etc/netnfsrc* file is executable by root.

```
chmod o+x /etc/netnfsrc
```

4. Execute the */etc/netnfsrc* script or reboot the HP system to activate NFS. Remember to set the ViewStation's file transfer protocol to NFS for this host.

5. Check the NFS transfer block size. The ViewStation uses a 8192 byte block size by default. If 8192 is too large for your network environment, you must set the correct value in the ViewStation's configuration file. The ViewStation resource is:

```
HDSsetup.NFS.block.size: 8192 (permitted values are 128 to
8192)
```

4.2.2 Configure the User Account

This section contains suggestions for setting up user accounts. There is only one crucial entry, which is setting the fontpath in the vue.settings file. You may want to use different methods and procedures, these are only suggestions to show the correct format.

4.2.2.1 User's .vue Directory

Each user has an individual .vue directory which allows them to customize the HPvue environment. The directory tree and file names are the same for ViewStations and the HP console. In fact, there are very few differences in the files themselves. The directory tree for any display is:

```
/display:0
 /current
```

```
palette.vue vue.resources vue.settings vue.session
```

These files are created automatically for the user when they login for the first time. A general description is given below. The *vue.settings* and *vue.session* files are of particular interest to the ViewStation user. The individual files are described below.

4.2.3 vue.settings file

This file contains entries that are different between the HP console and an HDS ViewStation. For the HP console:

```
Vuesession*Hostname: mustang.aero.org,loop-
back.aero.org
Vuesession*HostState: 1
Vuesession*FontPath: /usr/lib/X11/fonts/hp_roman8/
75dpi/,
 /usr/lib/X11/fonts/iso_8859.1/75dpi/,/usr/lib/X11/
fonts/misc/
Vuesession*ShutDownState: 1
Vuesession*ShutDownMode: 4
```

The same file for an FX ViewStation contains:

```
Vuesession*Hostname: mustang.aero.org,loop-
back.aero.org
Vuesession*HostState: 1
Vuesession*FontPath: /HDS$$ROM/,
 /128.92.3.12/usr/lib/X11/FXUtilities/hds-fx/fonts/hp/
iso100dpi/,
 /128.92.3.12/usr/lib/X11/FXUtilities/hds-fx/fonts/hp/
misc/,
 /128.92.3.12/usr/lib/X11/FXUtilities/hds-fx/fonts/hp/
roman75dpi/
Vuesession*ShutDownState: 1
Vuesession*ShutDownMode: 4
```

Notice the differences with the font path. This path is unique to each display. The iso100dpi and roman75dpi directories in the font path for the ViewStation are HP fonts that have been converted for use on the ViewStation. The tools used to convert the fonts are provided on the HDSware software tape.

4.2.4 Optional Customization for the ViewStation

You can customize the operation of the ViewStation by making some settings in the vue.session file. This file corresponds to the .xsession file in the MIT X11 releases.

4.2.4.1 vue.session file

This file contains the commands required to set up the screen and visual environment to be the same as the last time the user logged in. Additional customization would be to add a line to execute a local client on the HDS ViewStation. For instance, a line to start the local hdsclock is:

```
vuesmcmd -screen 0 -cmd "remsh fx-name hdsoclock -
geometry -0+0"
```

Notice that this requires the hdsoclock to run on "fx-name", which is the unique name of an ViewStation registered in the */etc/hosts* file (and yp if needed).

4.2.5 Other important files in the .vue directory

The .vue directory contains several subdirectories, the most important being a directory containing the display names used by the ViewStation, as well as the display name of the system console. For example, this directory might contain:

```
adam:0 calvin:0 jack:0
bob:0 hobbes:0 wizard:0
```

where `wizard:0` is the display name of the HP system console and the other names are HDS ViewStations. (These names must be the same as those registered in the */etc/hosts* file).

Note that there must be a directory for each display device because each device may have special requirements for its hardware and environment. Further, the user might want a different appearance for different display devices, or the user may want the same appearance for the screen regardless of which display device it is.

Note also that since the HP700 and the ViewStation use different font files, their fontpaths would show different entries. The ViewStation's fontpath line in the .vue file would look like this (depending, of course, on the actual directory location of the HDSware tape contents):

```
adam:0 /current/vue.settings
vuesession*FontPath: /HDS$$ROM/,/<HP host IP>/usr/hds-
fx/fonts/75dpi/
```

Note that this font path setting will override the font path stored in the FX setup or config file.

The corresponding setting for the HP700 would be:

```
wizard:0 /current/vue.settings
vuesession*FontPath: /usr/lib/X11/fonts/iso_8859.1/
75dpi/
```

4.2.5.1 palette.vue

This file contains the colors assigned to various components of the vue environment. The values are in `#rrrrggggbbbb` format. This file should only be modified by the vue environment control panel. This file will be the same format for either the HP console or the ViewStation.

4.2.5.2 vue.resources

This is the same file as .Xdefaults in a generic MIT distribution of the X11 Window System. This file contains resources for client settings to select options to be used when the client is run.

4.2.5.3 Creating a Common .vue Environment

To have all ViewStation's use the same environment, choose the desired environment directory, remove all other ViewStation directory files, then link the ViewStation names to that desired file. This ensures that the vue environment will now look the same for a user starting up on any ViewStation on the network.

New ViewStation's with the same environment can be added to the network by extending the linked .vue file to their name as well.

4.2.6 Settings on the HDS ViewStation

The HDS ViewStation is very easy to configure. From the Main Menu of Setup Mode, select the HDSware button and enter the three items requested:

- ViewStation IP address: *(this value is assigned by your system administrator)*
- HDSware host IP address: *(IP address of the host where the HDSware tape was loaded)*
- HDSware directory: *(the path to the HDSware tape directory)*

When this information is entered, click on the Save in NVM button, and restart the ViewStation. The ViewStation takes this HDSware information and stores it in its configuration file, which it then reads as it restarts. In this way, it locates its fonts, its server code, its client files, etc. and enters them automatically in the appropriate places in the Setup Mode menus.

This automatic ViewStation configuration doesn't set other items that may be important on your system. You should check Setup Mode for other important items, such as a gateway address, a domain name server, remote host entries, etc. When you are finished, save these items in NVM and your Setup is complete.

Using the ViewStation on Selected Host Systems

This chapter will help you connect an HDS ViewStation to a number of different host systems.

Section 4.0 - Using the ViewStation on Selected Host Systems

4.3 DEC VMS Systems

4.3.1 Overview

Digital Equipment Corporation (DEC) has created a proprietary operating system called VMS. It is most frequently used on an Ethernet network with a proprietary transport protocol called DECnet using other proprietary protocols such as LAT. At the same time, DEC manufactured systems using a Unix-based operating system, DEC's Ultrix, with the TCP/IP transport protocol on Ethernet. DEC customers had their choice between these independent systems. DEC's windowing system, DECwindows, is available on both VMS and Ultrix hosts.

In 1990, DEC announced a new transport protocol product, UCX - the VMS/Ultrix Connection, which merged the DECnet and TCP/IP protocols for DEC products, including DECwindows. You must have this TCP/IP layered product for your VMS system.

4.3.2 DEC UCX Protocol

The VMS/Ultrix Connection is available as a DEC product. There are also a number of commercial products that do the same thing, such as WIN/TCP from Wollengong, Multi-Net from TGV, and public domain software PCIP written at Carnegie Mellon University and available on the Internet.

Wollongong's WIN/TCP networking software.

The Wollongong Group

1129 San Antonio Road

Palo Alto, CA 94303

(415) 962-7100

Multi-Net, a TCP/IP multi-format product from:

TGV Corp.

603 Mission Street

Santa Cruz, CA

(800) 848-3440

Public domain networking software called PCIP from Carnegie Mellon University. This software is available only through Carnegie Mellon University at a cost of $150 for the media and the documentation. (An HDS customer reports that the PCIP materials are "a very nice package, and easy to install".)

You can obtain the PCIP software from:

Computer Science Department

Carnegie-Mellon University

Pittsburgh, PA

(412) 268-2647

Using the HDS ViewStation on VMS systems has two major considerations. The ViewStation runs only on TCP/IP networks, so the VMS host must run on TCP/IP (DECnet is not supported). A variety of TCP/IP implementations are available, including one from DEC. The CMU TCP/IP implementation is not supported.

The second consideration is VMS syntax on file names and paths; the ViewStation will use VMS syntax, but its defaults are UNIX-based names and paths, so you must specify host type as VMS so the ViewStation communicates correctly.

4.3.3 ViewStation Setup Mode Entries for VMS

You must make some adjustments to your Setup Mode entries for working on VMS systems. These adjustments are mainly just the syntax needed for file names and paths. Since these syntax changes are present in the HDSware VMS release, that is the easiest way to enter them. When the ViewStation uses the configuration file to start, it uses those paths as current entries; you should save them in NVM so they are permanently entered in Setup Mode as well.

These are discussed, with examples, below.

Remote Hosts menu

Host Name: <*VMS hostname*>

Network Address: <*IP address*>

Host Type: VMS

Access Protocol: FTP or TFTP or NFS (NFS is available for bootingwith HDSware version 3.2 and later)

Access Parameters:

User Name: <*login_id for FTP*>

Password: <*password for FTP*>

These entries are needed to tell the ViewStation to use VMS naming syntax for this host, and FTP is needed because some versions of VMS and UCX may not support the TFTP protocol.

User Preferences menu

Font Path and RGB Settings

Default Font Path: /fonthost/disk$pack:[*fontpath*]

(for example, /fonthost/disk$pack:[hds-fx.fonts.misc])

(or the preferred path: /FXUtilities/[hds-fx.fonts.misc])

4.3.4 NFS on VMS Systems

If the TCP/IP protocol running on your VMS host supports NFS (almost all do), then NFS can be used to load files (such as fonts, clients, etc.) when the ViewStation requests them. Note that NFS is available for booting the ViewStation with HDSware version 3.2 and later.

NFS transport uses UNIX-style file naming, so you should enter paths and filenames with UNIX syntax on the ViewStation (both in Setup Mode and in the configuration file) if you are using NFS to load them. The NFS protocol on the VMS host translates the syntax to the correct VMS type. The FXUtilities hosttype should be set to UNIX, not VMS.

The HDSware VMS distribution has a separate configuration file for using NFS with a VMS system. This is labeled *./hds-fx/config/config_fx.nfs*. Its syntax is correct for VMS systems using the NFS protocol.

4.3.5 HDSware Installation for VMS Systems

You can install HDSware easily with a few simple commands. You should enter a definition in your SYS$MANAGER:SYLOGICALS.COM file:

DEFINE/SYSTEM/TRANS=CONCEAL HDS$HOME: <*disk device | Logical*>:[*install-directory.*]

where <*disk device*|*Logical*> is the name of the disk where the tape is loaded and *install-directory* is the directory name. Note the period after *install-directory*; this is very important because it links HDSware with that directory. Be careful with the syntax in this command. A sample line might be:

DEFINE/SYSTEM/TRANS=CONCEAL HDS$HOME: DUA0:[HDS.]

This sets up a logical name HDS$HOME for HDSware.

Set the directory to the install directory:

SET DEF HDS$HOME:

Insert the tape in the tape device (note its name in your system). Then load the HDSware software with the command:

BACKUP/NOASSIST <*tape device name*>:HDS304.BCK/SAVE [...]

This will load the HDSware and its directory structure on your system. The tape label has "HDS304" (or the appropriate version number) as its name. The [...] is an important part of the command line; it permits the creation of the HDSware subdirectories in the proper levels and relationships.

This tape requires approximately 60 MB of storage, including the document files.

4.3.6 Booting the ViewStation from a VMS Host

If you want to boot the ViewStation from a VMS host, you must first load the HDSware on the host, as in the previous section. You must have HDSware version 3.0.4 or later (both server code and bootprom) to use this VMS booting procedure.

When you start the ViewStation, press "m" at the beginning of the white screen to enter Monitor Mode. In Monitor Mode, make these settings:

Set the ViewStation's IP address:	**ip** *nnn.nnn.nnn.nnn* also set the netmask and gateway addresses, if needed
Set the HDSware (VMS) host address:	**hh** *nnn.nnn.nnn.nnn*
Set the HDSware path (VMS syntax):	**hd** <*directory name*> as in the example above, this is: **hd** HDS$HOME:
Set the Configuration file:	**cf** /FXUtilities/[hds-fx.config]config.fx note that "FXUtilities" is an abbreviation for the path: **hh** + **hd**
Set the host type to VMS:	**set** HDSsetup.FXUtilities.hosttype: VMS

If needed, you can also set the file transfer protocol to FTP, including the entry of a username and password with the "set" command with the appropriate configuration resource names.

Select the FTP protocol:	**set** HDSsetup.FXUtilities.protocol: FTP
Set the username (for FTP):	**set** HDSsetup.FXUtilities.username: <username>
Set the password (for FTP):	**set** HDSsetup.FXUtilities.password: <password>

When all of these settings are made, you can boot the ViewStation.

Boot the ViewStation:	**boot**

This boots the ViewStation by loading the server code file, whose location is specified in the configuration file, by means of the file transfer protocol you specified. Both of these conditions must be met - that is, the file must be in the location entered (the configuration file knows the correct location unless the HDSware tree has been modified), and the file transfer protocol (TFTP or FTP) must be running and have access to that location.

4.3.7 Automatic ViewStation Configuration

When the HDSware tape is loaded and the HDSware information (host IP address and path) entered in Setup Mode, the ViewStation will configure itself automatically when it is rebooted. The tape location is entered in the ViewStation's internal configuration file, which is then used during the reboot. In this way, the appropriate fontpath information, the server code location, the locations and resources for the HDSperx clients, etc., are automatically entered. (You probably should save these settings in NVM also.)

On reboot, the ViewStation will have the HDSperx clients, the Motif and OPEN LOOK window managers, and X11R4 fonts available, and the HDSperx pull-down menu for your local window managers will be in place. All this configuration takes place automatically with the HDSware feature.

4.3.8 HDSware Tape Contents

The HDSware tape contains server code files, fonts, client application programs, utility programs, font management programs, etc. The HDSware files, including the configuration files, font names, etc. have been modified to conform to VMS syntax. There directories are shown below with their subdirectories. Note that some of these directories, such as [.UTILS] and [.CONTRIB] are provided for sites with Unix platforms.

This list changes with each HDSware release, but this is representative.

[.HDS-FX.IMAGE]

Server code directories - FX server files with different configurations

README

VNN-D.HDS (server code capable of downloading clients)

VNN.HDS (no download capability in server code)

VNN-BOOTPROM.BIN (bootprom file)

[.HDS-FX.IMAGE_U]

These same files in uncompressed format (.UHDS extensions)

[.HDS-FX.FONTS]

X11R5 Font directories - a main font directory, with subdirectories:

[.100DPI]

[.75DPI]

[.MISC]

[.DEC] (DEC fonts)

[.XOL] (MIT X11R5 OPEN LOOK fonts)

[.ASTERX2_1] (Applix Aster*x v2.1 fonts)

[.ASTERX3_0] (Applix Aster*x v3.0 fonts)

[.WP5_1] (special WordPerfect v5.1 fonts)

[.LOTUS123] (special Lotus 123 fonts)

[.SOFTPC] (special SoftPC fonts)

[.HDS-FX.BIN]

HDS client files, such as window managers and HDSperx clients

[.HDS-FX.LIB]

Library directory - containing libraries needed for HDS clients

[.APP-DEFAULTS]

[.BITMAPS]

[.HDSSCRIPT]

[.SHARED]

[.HDS-FX.UTILS]

Utility program directories, with two subdirectories

[.SRC] (Source code with appropriate Makefiles, etc.)

[.SRC.HDSFC960]

[.SRC.BDFTO960]

[.SRC.GETBDF]

[.SRC.HDSMKDIR]

[.SRC.HDSCRYPT] (for password encryption)

[.SRC.FLASHFILESYS] (scripts for managing the flash file system)

[.EXECS] (executable HDS utility programs above)

[.EXECS.SUN4]

[.EXECS.VAX_ULTRIX]

[.EXECS.SCO386]

[.EXECS.HP700]

[.EXECS.DG_AVIION]

[.EXECS.IBM_RS6000]

Other utilities and sample files

[.HDS-FX.RGB]

Sample rgb.txt stored in ViewStation memory

[.HDS-FX.MAN]

Manual pages for the HDS clients in several formats

[.HDS-FX.CONFIG]

Sample configuration files

[.LPD-SUPPORT] (LPD print spooler)

[.LP-SUPPORT] (LP print support)

CLIENTS.CONFIG (configuration file for HDS download clients)

CONFIG.FX (the FX configuration file, including sample files)

SYSLOG.CONF (the syslog.conf file for error reporting)

[.HDS-FX.INCLUDE]

Include files needed for some compilations

[.HDS-FX.CONTRIB]

Other contributed files provided for the user's convenience

[.BOOTP] - source code and BSD and Sys V executables

[.XSCOPE] - source code and Sun4 executables

[.XMKMF] - shell script to generate a makefile from an Imakefile

[.SNFTOBDF] - utility program for generating bdf font files

[.USEPTY] - utility program for pseudo-terminal operations

[.XRSH] - utility program for Xrsh

[.HDS-FX.EXTENSIONS]

X11R5 extensions downloadable into the ViewStation server

These files require approximately 60 MB of storage space (or less, depending on the directories you choose to load), so you should allocate sufficient space for them.

4.3.8.1 VMS File Names and Syntax

You must use VMS naming conventions in all ViewStation entries specific to the VMS host, such as path names, configuration file entries, etc. VMS file names are specified with period separators, as in these examples:

Unix syntax: /fonthost/hds-fx/fonts/misc

VMS syntax: /fonthost/disk$pack:[hds-fx.fonts.misc]

Examples for the configuration file:

Unix syntax: /FXUtilities/hds-fx/config/config-fx

VMS syntax: /FXUtilities/disk$pack:[hds-fx.config]config.fx

4.3.9 Handling Special DEC or VMS Fonts

There are two sides to this problem, first, preparing DEC fonts for the HDS ViewStation, and second, preparing any special fonts (such as Open Windows fonts discussed below) for display on the VAX.

4.3.9.1 Preparing DEC Fonts

DEC fonts typically reside in a "sysfont" directory belonging to the root user. The ViewStation can use .pcf, .bdf, or .snf font files directly (it can also use the X11R5 font server if one is available). This is sufficient flexibility that it is unlikely that you have to do anything to the DEC fonts to use them on the ViewStation.

If you have to convert fonts, there should be .bdf fonts for all the DEC fonts. In most cases these fonts duplicate the X11R5 fonts available with the MIT X distribution, however they may have slightly different names, and there may be some proprietary fonts supplied by DEC. A little investigation of fonts that cannot be loaded often shows the problem and a simple alias for the font name is a good solution.

If you have proprietary DEC fonts in an incompatible format, you must compile them in the HDS ViewStation .snf format, but that compilation must be done on a Unix or Ultrix host. These fonts can be moved to the TCP/IP part of the network as .bdf fonts and converted with the "bdftohds" utility program as described in Section 1.9.5. The resulting .snf fonts can then be moved back to the DECnet side or retained on the TCP side, since the ViewStation can access fonts from any host.

4.3.9.2 Preparing Other Special Fonts

The second situation, that of compiling other fonts for use on the VMS server, is also straightforward. This example shows how to convert Sun Open Windows fonts for use on the DEC VMS server. Go to the source of the special fonts and copy them to the DEChost:

```
sun> cd $OPENWINHOME/share/src/fonts/misc

sun> dnicp *.bdf decvax::[vaxdir]
```

then, on the decvax machine, compile them with the "font" utility:

```
$ font olcursor.bdf

$ font olglyph10.bdf
```

This results in fonts with file names in the form "olglyph10.decw$font;1". Now copy the fonts to the "sysfont" directory (you must have system privileges to do this):

```
$ set def sys$sysroot: [sysfont.decw.user_cursor16]

$ copy [vaxdir] olcursor.decw$font;1 *
```

(note: the Lucida fonts should be put in [sysfont.decw.user_75dpi] and sysfont.decw.user_100dpi]

4.3.10 DECnet Operation with Open Windows

Some systems, such as Open Windows on Sun machines, use their own conventions to communicate with DECnet. This is called "DECnet Internetworking" or DNI (Vers. 7.0 is used for these examples).

There are some steps to do before you make the connection. First, enable the DNI environment for the Open Windows system. Set the environment variable:

```
sun> setenv DNI_X_ENABLE 7
```

and then check whether authorization is needed for the DEC machine. If this is needed, you must modify the xhosts file:

```
sun> xhost decvax::
```

note that the double colon specifies the DECnet transport for this DEC host connection. In order for an X client to connect to the Open Windows server, the DECnet node addresses must also be mapped to their DECnet node names by initializing the NCP database on the Sun (and similarly on the DEC VMS host).

A typical session might then look like this:

```
sun > dnilogin decvax

$ define DECW$DISPLAY SUN_machine::0

$ spawn/nowait run X11_client
```

If you want to run a client, such as mailtool, on the Sun and export the display to the VAX, you can simply use the environment variable as follows:

```
sun> setenv DISPLAY decvax::0

sun> mailtool
```

but note that this will only work if the proper fonts have been installed on the DEC machine, in this case, the Sun Open Windows fonts. For an Open Windows system, these fonts are a minimum set needed to run.

```
olcursor.bdf

olglyph10.bdf (also 12, 14, and 19 points)

luBS08.bdf (also 10, 12, 14 points)

luRS08.bdf (also 10 points)

luFS12.bdf

lutBS12.bdf

lutRS10.bdf (also 12 points)
```

4.3.11 Some typical problems on VMS

Depending on the level of security running on your system, there may be some problems with host access. You should enter names of all the hosts on your network in the xhosts file and in the DECNET_NAMES.DAT file. Just add the names with each one on a new line as follows "<hostname>::".

Using the ViewStation on Selected Host Systems

This chapter will help you connect an HDS ViewStation to a number of different host systems.

Section 4.0 - Using the ViewStation on Selected Host Systems

4.4 IBM RS6000 Systems

4.4.1 Overview

IBM has a line of workstations called the RS6000 series with its own UNIX operating system, AIX. This discussion describes features of AIX 3.2.5, which uses the X11R5 release of X.

Configuration of the IBM RS6000 is greatly simplified with the AIX 3.2.5 release, which includes the "SMIT" utility (System Management Interface Tool).

There is very little special attention needed for connecting ViewStations.

4.4.2 TFTP File Transfer Protocol

The IBM RS6000 does not start TFTP in its default configuration.

It is easy to start the TFTP protocol with the SMIT tool. If you use the fast path mechanism with the SMIT tool, this is very simple. Type:

```
smit inetd.conf
```

This points the tool to the inet daemon configuration file and gives you a menu for the daemon's processes, (list, add, change, and delete). If you "list" the processes, you will see that TFTP is absent. Using the "add" command, you can click on TFTP and SMIT adds it to the running daemon processes.

The TFTP daemon does not run in secure mode by default, so it is ready to load the ViewStation's files from any directory.

4.4.3 IBM RS6000 Fonts

The IBM RS6000 does use some special IBM fonts, roman and bold. These are available in the RS6000's */usr/lib/X11/fonts* directory as .pcf fonts. Since the ViewStation can read the .pcf fonts directly, all you need to do is insure that this directory is in your fontpath.

If you are using the font server on the RS6000, this step is not necessary. Be sure to add the RS6000's font server to the ViewStation's fontpath.

4.4.4 RSH Permissions on the RS6000

If you are using the ViewStation's local window managers, you will need to check the name service and the permissions to be sure the Rsh process works correctly.

Verify that the ViewStation's name is entered correctly in the */etc/hosts* file.

Verify that the permissions in the */etc/hosts.equiv* file are appropriate for the ViewStation (check the initial rsh process and any further rsh commands to other hosts). Normally, this file has an entry:

```
+ +
```

indicating "+" for the host and also "+" for the user, which means that any host and any user has permission. You may want to modify those settings, but be sure that the ViewStation has permissions for the appropriate hosts in order to read the window manager files and/or start processes from the window manager menus.

4.4.5 Starting Xdm on the RS6000

The RS6000 running AIX 3.25 doesn't start Xdm in its default configuration. The Xdmcp client is located in */usr/lib/X11/xdm*. There is an auto-start program in that directory. Simply move to the xdm directory and type:

```
xdmconf
```

This starts the Xdm process. The configuration files are all appropriate for Xdm operation, though you may want some customizations.

Typing xdmconf -d removes the Xdm process. There is a man page called xdmconf.man.

HDS ViewStation Developer's Kit

This section gives an overview of the HDS ViewStation Developer's Kit and its use.

5.0 Section 5.0 - HDS ViewStation Developer's Kit

5.1 HDS ViewStation Developer's Kit Overview

HDS's Developer Kit allows users to compile their application program source code to run on the ViewStation's Intel 960 processor as a local client. The Developer Kit allows you to create the executable file for inclusion in the HDSware software.

5.1.1 HDS ViewStation Architecture

The HDS ViewStation uses a powerful RISC processor, the Intel 960, and an internal Unix micro-kernel architecture that make it possible to run a number of client application programs on the ViewStation. This offers the ViewStation user several important advantages:

- customization of client applications
- distribution of processing load to ViewStations on the network
- conservation of host memory and CPU resources
- conservation of network bandwidth
- improved response and control for the user

The ViewStation Developer's Kit contains tools that let you prepare client applications to run on the ViewStation. It consists of two tapes containing the tools, libraries, compilers, and sample files you need to work.

5.1.2 Developer's Kit Contents

The Developer's Kit consists of two parts, the GNU tools and the HDS libraries. A separate set of documentation is provided.

5.1.2.1 GNU Tools

The HDS ViewStation Developer's Kit consists of a set of GNU tools (v2.0.1) for software development and debugging and a cross-compiler for the Intel 960.

The GNU tools are public domain tools and the Intel compiler is distributed under license from Intel; both are available on a QIC150 tape (or other formats). HDS currently supports the tools in the Sun environment, though they are available on other platforms. If you use another platform, the tools should be correctly installed and operational before you start; consult with HDS if you have any questions.

5.1.2.2 HDS Libraries

The Developer's Kit also contains various libraries and include files needed for the compilation; these include the standard X11 libraries, the Xt Intrinsics and Xm Motif libraries, a Unix compatible C library and a floating point library. These are currently X11R5 libraries and Motif 1.2.3 versions.

There are also sample client files and other examples, including sample makefiles for the ico, maze, and Motif_hello programs.

The ViewStation uses standard X libraries and the Motif toolkit. These are integral parts of the ViewStation server code. Most system calls and procedures are supported so compilation should proceed with no problems. If you have problems with the compilation, especially because of missing libraries, etc. you should contact HDS Technical Support directly.

HDS ViewStation Services

This section describes services that HDS provides for the ViewStation, including warranty and software update information.

6.0 Section 6.0 - HDS ViewStation Services

6.1 HDS ViewStation Warranty

HDS Network Systems, Inc. (HDS) warrants that each ViewStation terminal will be free from defective materials and workmanship for twelve (12) months from the date of shipment to the original customer. Upgrades to current software levels are provided at no cost for twelve (12) months from the date of the original software shipment on QIC150 tape format with the exchange of the tape media itself.

HDS agrees to correct any of the above defects (parts and labor only) when the terminal is returned to the factory, freight prepaid by the customer. Return authorization must be obtained from the HDS Service Department prior to returning the terminal to the factory. The repaired terminal will be returned to the customer freight prepaid.

Under this warranty HDS may at its option repair or replace the defective terminal or terminal components. Normally, HDS will not ship replacement equipment until the defective terminal or terminal component is received (freight prepaid by the customer) at the HDS factory or service depot -- HDS at its option may ship replacement equipment prior to receipt of the defective equipment. In this case the customer will receive an invoice for the full price of the equipment upon shipment of the replacement and a cancellation of that invoice upon receipt of the defective terminal or terminal component at HDS' factory or service depot. The model number and serial number of the terminal must be furnished by the customer at the time of request for warranty service.

This warranty shall be invalid if, in HDS' sole judgment, the terminal or terminal component has been subjected to misuse, abuse, neglect, accident, improper

installation or application, alteration or neglect in use, storage, transportation or handling, or if the serial number has been removed, defaced. or altered.

The customer will ensure that the unit to be serviced is returned to HDS Network Systems, Inc. in the correct packaging, and is properly secured and placed in the appropriate original carton utilizing all original restraining pieces (Styrofoam end caps, foam inserts, etc.). If the customer does not have appropriate shipping medium, HDS Network Systems, Inc. will make the correct materials available to the customer for a nominal charge. Any equipment deemed damaged by HDS Network Systems, Inc, as a result of improperly packed equipment will not be covered by the terms of this contract. The customer will be held liable for any additional repair costs due to the equipment being damaged in transit.

This warranty applies in the United States only.

6.2 HDS ViewStation Service

Technical support for setup or configuration problems is available at 800.HDS.1551. This gives you several connection options: choose "2" for Customer Care for ordering information and upgrades, choose "3" for Service and "hardware" problems, and choose "4" for Technical Support and for "software" problems.

When obtaining service for the ViewStation, the Service Department requires information about the ViewStation, including its serial number, and the type of problem you're having. Service technicians or Technical Support personnel will discuss the problem with you if needed.

If the ViewStation has to be returned to the factory for repair, an RA (Return Authorization) number will be issued. You must include this number on the shipping documents and put it on the shipping carton to facilitate tracking. The customer pays for shipping to the factory.

The ViewStation will be repaired and returned to you freight prepaid.

6.2.1 HDS ViewStation Upgrades

The ViewStation is designed for easy changes and upgrades. In some cases, the base unit must be returned to the factory. This all depends on the type of upgrade needed.

6.2.1.1 Memory Upgrades

Adding memory to a ViewStation is a user upgrade. Disconnect the ViewStation's cables. Turn the base upside-down and remove the four screws. Turn it upright and slide the base's cover off.

- For "FXB/C/D" ViewStations

  These ViewStations have two SIMM slots (long 72 pin connectors) toward the front center of the circuit board.

  You can insert SIMMs of any size (mixed sizes will work).

These SIMMs should be 32 bits wide and 70 ns or faster. They should have tin-lead contacts (not gold). These are not parity-checking SIMMs, which are 36 bits wide. (32 bit wide SIMMs are often called "Mac" SIMMs rather than the 36 bit wide "PC" SIMMs.)

You can add memory by inserting a SIMM module (insert at an angle, press down so it is fully seated, then rotate to a vertical position so the clips on the side lock it in place).

- For ViewStation Ultra models

  SIMMs for the ViewStation Ultra must be added as identical pairs. You cannot mix the SIMM sizes or types.

  The upgrade procedure is the same as for the regular ViewStation described above.

- For older ViewStations (G3 circuit boards)

  Older ViewStations have 4 SIMM sockets and use 8 bit wide SIMMs. These must be added as identical sets of 4 SIMMs. They should be 70 ns or faster and use tin-lead contacts.

  The upgrade procedure is the same, but you must add 4 SIMM modules.

6.2.2 HDSware Software Upgrades

New software versions and feature enhancements for the HDS ViewStation are distributed on CD-ROM or on the HDSware tape. This software upgrade service is available at no charge for one year after the ViewStation's purchase date.

HDSware upgrades are available from HDS's Internet site, http://www.hds.com. A password is required for downloading the software.

Customers who have not subscribed to Premium Software Service can receive a no-charge HDSware CD-ROM or tape upgrade within the software upgrade period. Contact the CustomerCare or Service departments for specific information and shipping instructions.

There is a charge for upgrades on tape formats other than QIC150. For further information you can call the HDS Service Department at 800-HDS-1551 or email at *info@hds.com*.

6.2.2.1 HDSware Premium Software Service

Customers who have ordered the Premium Software Service receive automatic distribution of the HDSware tape. Contact your HDS Customer Care representative for information about Premium Software Service.

I Index

Symbols

.bdf fonts 58
.snf fonts 58

Numerics

4 MB Memory ViewStations 123
4 MB ViewStation 123
4 MB ViewStations 148
4mm and 8mm DAT tape 57

A

Access Control security 535
Access Parameters field 105
Access Protocol field 104
Access to the Console Window 28
Access to ViewStation Ports 41
Adding a PCMCIA Hard Disk Drive 177
Anonymous FTP 521, 561, 570
Anonymous FTP File Transfer 521, 561, 570
Applix Aster*x fonts 51, 53, 579
Apply button 91
ARP protocol 115
ARP, Address Resolution Protocol 70
Audio Clients 387
Audio Device Setting Menu 109, 386
Audio In Device 109, 386
Audio module - FVWM 490
Audio Operation Hardware 385
Audio Out Device 110, 387
Audio Ports - Input / Output Connections 13
auhds 389
Automatic Fontpath Configuration 164
Automatic ViewStation Configuration 55
Auto-Starting an Application in HDSterm 221
Available Memory 93
Available Memory client 96

B

Backing Store 112
Backing Store / Save Unders submenu 111
bdfto960 52, 54, 166, 580
bdfto960 Font Utility 167
bdfto960 Font Utility Program 58
bdftosnf 58
Bell duration field 109
Bell pitch field 109
Bell volume field 109
Blanking screen saver 110
Booting the ViewStation from a VMS Host 123, 124, 578
Booting ViewStations with 4 MB Memory 123
BOOTP 19, 69, 115, 119
bootp 62
BOOTP (Boot Protocol Daemon) 563
BOOTP Benefits 128
BOOTP Command Fields 125
BOOTP request on Token Ring interface 523
bootptab 125, 523

C

CallPort integrated Speaker / Microphone 109, 386
Changing Server Code Versions 148
Changing the event translation 155
Changing the ViewStation Password 148
Changing ViewStation Settings 148
Changing xmodmap 155
Client Configuration Path field 116
clients.config 50, 54, 581
Communications Board option 2
Communications Menu 64

Communications Options Menu 15
Compiling HDSware Utility Programs 58
Compressed Font Files 165
config.fx 50, 54, 581
Configuration File clients.config.dsk 147
Configuration File config.fx.dsk 146
Configuration File field 115
Configuration File for 4 MB ViewStations 148
Configuration File Syntax 134
Configuration Password 89
Configuration password field 114
Configure Serial Port 1 field 64, 99
Configure Serial Port 2 field 64, 99
Configuring the ViewStation for Multimedia 183
Console Window 27, 93, 95
Console Window as a Local Display 29
Console Window Commands 31
Console Window from a Remote Host 29
Console Window Messages 35
Console Window Privileged commands 34
Console Window Security 30
CSLIP 64, 77
CSLIP , "Van Jacobsen compression", RFC 1144 75
CSLIP communications 75
CSLIP protocol 74
Customizing HDSterm Keys 160

D

DEC LK401 (VT420 type) keyboard 3
DEC TK50 cartridge tape 56
Default Font Path field 107
Default Suffix field 106
Default Telnet host field 103
Determine IP address field 115
Determining the IP Address 129
Determining the ViewStation's IP Address 133
Diagnostic Messages from a Remote Terminal 30
Direct Serial Connections 79
Disk drive 119
Display memory warnings at field 113
Display Monitors 4
Domain Name Server 39
Domain Name Servers Menu 15
Domain Name Servers Selection 105
Downloading the RGB Color Definition File 107

E

Entering Setup Mode 85
Environment Variable Keystrokes 151
Ethernet address 19, 70
Ethernet and IEEE 802.3 Differences 71
Ethernet Communications 64, 99
Ethernet Connections 63, 65
Ethernet hardware address 19
Ethernet IP address 66
Ethernet netmask field 66
Ethernet Type field 67
Event Translation table 156
Exit button 92

F

fdformat utility 198
File Names and Paths in Setup Mode 88
flash boot command 56
flash files command 56
flash server command 56
flashfilesys 52, 54, 580
flashfilesys command 56
flashfilesys Utility Program 61
flashPROM 119
FlashPROM Board 2
Floppy Disk Drive 3, 195
Floppy Disk Drive Access 196
Floppy Drive File Ownership 198
Floppy Drive Formatting disks 198
Floppy Drive NFS Access 197
Floppy Drive RCP (Remote Copy) Access 197
Floppy Drive with a Printer 196
Font caching memory 96
Font File Formats - .bdf and .snf 166
Font Handling 163
Font Server Operations 164
Fonts in the X Window System 163
FTP 559, 568
FTP - File Transfer Protocol 570
FVWM - GoodStuff Module 497
FVWM auto-raise module 492
FVWM background changer module 493
FVWM button panel module 497
FVWM debugger module 496
FVWM desktop clutter reduction module 495
FVWM iconbox module 500
FVWM identify-window module 505
FVWM Pager module 506
FVWM scroll-bar module 509
FVWM Virtual Desktop 461
FVWM Virtual Window Manager 459
FVWM window list module 510
FvwmAudio 490
FvwmAuto 492
FvwmBacker 493
FvwmBanner 494
FvwmClean 495
FvwmDebug 496
FvwmIconBox 500
FvwmIdent 505
FvwmPager 506
FvwmScroll 509
FvwmWinList 510
FX Server Code File field 115
FXUtilities Abbreviation 38, 88, 134

G

Generating an Individual Configuration File 143

Get server code from field 115
getbdf 52, 54, 166, 580
getbdf Font Conversion Utility 61
getbdf Font Utility 168

H

Handling .bdf Fonts and Fonts in other Formats 168
Hangup 79
Hard Disk Configuration File 146
HDS Audio 387
HDS Audio File Playback program 390
HDS Audio Manual Page 419
HDS Camera 109, 386
HDS Conference 395, 403
HDS external speaker 110, 387
HDS Font Utility Programs 166
HDS headset 109, 386
HDS internal speaker 110, 387
HDS microphone 109, 386
HDS Movie 395, 401
HDS NetAudio compatible server 389
HDS netTV Simulcast 395
HDS netTV Viewer 395
HDS netVideo Clients 395
HDS netVideo Hardware 392
HDS netVideo Playback Specifications 393
HDS netVideo Specifications 393
HDS screensaver screen lock 111
HDS Tuner 395, 408
HDS tuner 109, 386
HDS Video 396
HDS Video Hardware Configuration 392
HDS ViewCam Camera 396
hdsaudio 184
hdscrypt 52, 54, 580
hdscrypt Encryption Utility Program 61
hdsfc960 52, 53, 166, 580
hdsfc960 Font Conversion Utility 59
hdsfc960 Font Utility 168
HDSinstall 49, 50, 53
hdsmkdir 52, 54, 59, 166, 580
hdsmkdir Font Utility 168
hdsmkdir Font Utility Program 59
HDS-Model File 555
HDSperx 47, 579
HDSperx clients 94
HDSperx Screensaver client 111
hdsplay 390
HDSrunwm Launcher Program 284, 306
hdsshare 381
HDSshare Manual Page 381
HDSterm 79, 93
HDSterm Environment Variable Keystrokes 151
HDSterm Event Translations 159
HDSterm Keymap Groups 159
HDSvidinstall 394
HDSware 47
HDSware Abbreviation 134
HDSware Contributed Programs 61
HDSware Directory Location 87
HDSware distribution formats 47
HDSware host 38
HDSware Host Address 87
HDSware Installation Script 577
HDSware netVideo Software 394
HDSware on an Internal Hard Disk Drive 48
HDSware path 38
HDSware Tape Contents 53
HDSware Utility Programs 58
Help button 92
Host Name field 104
Host resolver file address field 106
Host resolver file field 106
Host Type field 104
http://www.hds.com xxiii
hwconfig command 70

I

IBM 101+ style keyboard 3
IBM 102 key multinational keyboard 3
IBM 3270 79
IBM RS6000 Fonts 586
IEEE 802.3 10BaseT 67
IEEE 802.3 Auto-sense 71
IEEE 802.3 Ethernet 71
Implementing "opendev" in a C Program 42
Include Statements in a Configuration File 144
Installing HDSware netVideo Software 394
Intel INDEO compression 401
Internet Server xxiii
IP (Internet Protocol) address 69
IP Addressing with BOOTP 129
IP Addressing with RARP 129
IP Broadcast Address field 65, 99
IP Internet Protocol address 70
IP Network Addressing 70

K

Key Handling 154
Key Handling in a Client/Server Environment 154
Key Mappings 152
Keyboard and Bell Settings 108
Keyboard Connections 150
Keyboard Connector 9
Keyboard Cursor Mode 150
Keyboard Operations 151
Keyboard Type field 108
Keyboards 3
Keyboards and their Operations 149
Keyclick volume field 108
Keycode Modification Tables 152
KeycodeTranslation file 152
Keymap Changes with Event Translations 157

Keymap Changes with Xmodmap 155
Keystroke Functions on the ViewStation 150
Keysym Changes 156

L

Lexmark 122 key IBM 3270 keyboard 3
Link Integrity 67, 68
Loading Fonts to the ViewStation 170
Loading New Server Code into flashPROM 176
Loading the HDSware Tape Manually 56
Loading the ViewStation's Configuration File 134
Loading the ViewStation's Server File 133
Local Floppy Filesystem Access 197
Local Mount Directories 38
Local Mount Directory 39
Local Mount Directory field 105
Local storage Benefits 128
Logical Connection to the Ethernet 69
Long Break 80
Lotus 123 fonts 51, 53, 580
Low Memory Warning Client 97
LP Print Service 547
LP Print Service for UNIX System V Printing 553
LPD print spooler 50, 54, 580
LPD Print Spooler for BSD Systems 547

M

Making a fonts.alias File 173
Making a fonts.dir File 172
Making Special .snf Font Files 165
Medium Access Unit 67
Memory fragmentation 96
Microsoft AVI format 401
MIT bdftosnf 167
mkfontdir 59
Modifier and Lock Key Changes 156
Modifier Keymappings 153
Monitor Mode 47, 119
Monitor Mode Benefits 129
Monitor Mode Commands 121
Monitor Mode Key Mappings 120
Monitor Mode Keystrokes 151
Monitor Mode Sample Entries 122
Monitor Mode Screen 120
Motif Bindings for HDSterm Keys 162
Motif Window Manager 93, 97
Mount Directory Example 40
Mouse and Pointing Devices 3
Mouse Button Changes 155
Mouse Connector 8
MS-DOS diskettes 57
Multimedia Audio Specifications 183
Multimedia Devices 183
Multinational Keyboards and Mappings 157

N

NetAudio 184
netTV Simulcast 410
netTV Viewer 413
Network Address field 104
Network Connections 63
Network Printing Operations 547
Network timeout value field 65, 99
Network Trailers 71
Network-based ViewStation Operation 131
NFS 559, 568
NFS - Network File System 571
NFS on HP Systems 571
NFS on VMS Systems 577
Non-Volatile Memory 122
Notes on Backing Store and Save Under Features 112

O

Opeartion with the flashPROM Board 43
OPEN LOOK fonts 51, 53, 579
OPEN LOOK Virtual Window Manager 97, 564
OPEN LOOK Window Manager 93, 563
Open Windows Applications 563
opendev 31
opendev Command 41
Operation with PCMCIA Devices 43

P

Parallel Port - DB25 Connector 10
passwdRequired 31
PC Card 175
PC Card Board 3
PC Card Devices 3, 175
PC Card interface 3
PCMCIA (Personal Computer Memory Card Interface Association) 175
PCMCIA device 43
PCMCIA interface 3
PCMCIA Port 13
PEX-3D Extension 212
Ping Command 84
Ping command - ViewStation 84
Power Supply 3
PPP 64
PPP Connection Example 81
PPP daemon on the host 81
PPP Diagnostic Message Level 78
PPP IP Address Negotiation 78
PPP Link Quality Monitoring 78
PPP Link Quality Test Interval 78
PPP Local Authorization Name 79
PPP Local Authorization Password 79
PPP Maximum Receive Unit 79
PPP PAP Authentication Allowed 79
PPP Protocol 77

PPP protocol 74
PPP Protocol Field Compression 78
PPP Protocol for Serial Communications 77
PPP Van Jacobson Header Compression 78
Primary Resolver field 106
Printing from the Console Window 37
Problems with BOOTP Starting 126
Putting the Font Server in the Font Path 164

Q

Quick Starting with Setup Mode 87

R

RARP 69, 115
RARP (Reverse Address Resolution Protocol) 115, 563
RARP Protocol Example 130
Reboot button 92
Remote Access to ViewStation Ports 41
Remote Configuration of the ViewStation 89
Remote Configuration of ViewStation 133
Remote Configuration Security Considerations 148
Remote Hosts Menu 15, 102
Remote Management of ViewStations 148
Remote Mount Directories 38
Remote Mount Directory 39, 105
Remote Setup Mode Operation 86
Reset NVM to Factory Defaults 16
Reset NVM to Factory Defaults Selection 117
Reset Server Command file 117
Reset server when last client exits field 113
Reset X Server field 117
Restart and Reset Menu field 116
Restart and Reset Menu Selection 116
Restart and Shutdown Menu 16
Restart button 92
Restart ViewStation from current RAM image field 117
Restart ViewStation from ROM / Network field 117
Restore Settings from NVM 16, 118
RGB Definition File field 107
rgb.txt colormap file 107
rlogin 29
Route Command 84
Route table for ViewStation 84
Routing table 71
RS6000 585
rsh 29
RSH Permissions on the RS6000 586
Run Local Chooser field 101, 534

S

Sample bootptab File 127
Sample Printer Model File 553
Save button 91
Save Unders 112
Screen saver interval field 110
Screen saver method field 110
Screen Saver Settings 110
Screen saver timeout field 110
Screen Settings field 113
script 37
Secondary Resolver field 106
Selecting the Ethernet Physical Typ 68
Selecting the Monitor Type 5
Send SNMP traps field 116
Serial Connection Example 80
Serial Connections 63
Serial Port - DB-25 Connector 13
Serial Port - RJ11 Connector 9
Serial Port 1 74
Serial Port 1 (RJ11) 74
Serial Port 2 74
Serial Port 2 (DB25) 74
Serial Port Menu 74
Serial Port Physical Connection 74
Serial Terminal Window 79, 93
Server Reset Command File field 116
Server-reset Command File 44
Setting the Fontpath 170
Setup Mode 85
Setup Mode Editing 88
Setup Mode Editing Keys 150
Setup Mode Function Buttons 91
Setup Mode Heading Bar 90
Setup Mode Hidden Icon Hotkey 150
Setup Mode hotkey 85
Setup Mode IP Address Selection 129
Setup Mode Keystrokes 150
Setup Mode Main Menu 90
Setup Mode Operation 86
Setup Mode Password Protection 89
Setup Mode Selection Buttons 90
Setup Mode with Hidden Icon 85
Setup Options Selection 97
Shared Memory Extension 212
Shift-Pause key 80
Short Break 79, 80
SLIP 64, 77
SLIP protocol 74
snftobdf Font Conversion Program 62
SNMP (Simple Network Management Protocol) 116
SNMP Trap host field 116
SoftPC fonts 51, 53, 580
SQE 67, 68
Start CSLIP 79
Start PPP 79
Start SLIP 79
Starting from a ViewStation Hard Disk 119
Starting from Local Storage 119
Starting from Monitor Mode 120
Starting the ViewStation 119
Starting ViewStation Clients 93

Starting with BOOTP 125
Starting Xdm on the RS6000 586
Startup Command File 44
Startup Command file 117
Startup Command File field 116
Startup Options Menu 16
Startup Options Selection 114
Sun and SCO diskettes 57
Sun Keyboard Connector 13
Sun Network Services 559
Sun NIS IP Addressing 130
Sun Open Windows Applications 559
SUN tape handling 56
Sun Type 4 keyboard 3
Sun Type 5 keyboard 3
Syslog Configuration File field 116
syslog.conf 51, 54, 581

T

Tandem 6530 79
Telnet Terminal Window 93
TFTP 559, 568
TFTP - Tririal File Transfer Protocol 568
TFTP File Transfer Protocol on RS6000 521, 585
TFTP in “Secure Mode” on HP Systems 520, 569
TFTP on HP/UX systems 520, 569
TFTP service on HP Systems 519, 568
The flash boot Command 179
The flash files Command 179
The flash server Command 179
ThickWire Connection 68
ThickWire Ethernet 67
ThickWire Ethernet - DB15 Connector 12
ThinWire Connection 67
ThinWire Ethernet 67
ThinWire Ethernet - BNC Connector 12
Token Ring Communications Menu 65, 99
Token Ring Connections 63
Token Ring Interface 71
Token Ring PCMCIA card 63
Twisted-pair Connection 67
Twisted-pair Ethernet 67
Twisted-Pair Ethernet - RJ45 Connector 11
Twisted-pair Ethernet Auto-sense 69

U

Undo button 92
Use backing store field 111
Use save unders field 111
Usepty Script 42
usepty Utility Program 62
User Preferences Menu 15
User Preferences Selection 106
Using Alternate Font Servers 165
Using the fdisk Command 178
Using the flash Command 179
Using the Hard Disk Drive 178

V

Verifying the Fontpath Setting 171
Video Connector - DB15 high-density 11
ViewStation .snf Font Files 165
ViewStation assigned IP address 90
ViewStation Audio Operation 385
ViewStation Audio Specifications 391
ViewStation Audio with Application Programs 390
ViewStation Circuit Board specifications 1
ViewStation Console Shell Window 27
ViewStation Controls and Indicators 8
ViewStation Environmental Requirements 8
ViewStation Error Message Reporting 35
ViewStation External Connections 2
ViewStation firmware version 90
ViewStation Flash File System 43
ViewStation FlashPROM 176
ViewStation Font File Formats 165
ViewStation Hard Disk Drive Storage Devices 177
ViewStation installed memory 90
ViewStation Internal Architecture 27
ViewStation Internal File System Structure 38
ViewStation Keyboards 149
ViewStation Light pen 4
ViewStation Memory specifications 1
ViewStation monitor resolution 90
ViewStation Mount Points 38
ViewStation Multimedia Operation 385
ViewStation Network Services 16
ViewStation Part Number 19
ViewStation PC Card Adapter Card 175
ViewStation Physical Characteristics 7
ViewStation Pointer Mappings 153
ViewStation Print Spooler Example 552
ViewStation Remote Access Reporting 189
ViewStation rhosts Security 189
ViewStation ROM cards 177
ViewStation Serial Connection 74
ViewStation Serial Number 19
ViewStation Server-Resident Fonts 17
ViewStation Specifications 1
ViewStation Startup Considerations 128
ViewStation Startup Files 43
ViewStation Video Memory specifications 2
ViewStation Video Operations 392
ViewStations with 4 MB Memory 123
VMS Setup Mode Entries 576
Vnn.hds Server Code 94
Vnn-D.hds Server Code 94

W

WordPerfect fonts 51, 53, 579

X

X logo screen saver 110

X11R5 MIT-SHM extension 395
Xdm - Broadcast access method 100, 533
Xdm - Direct access method 101, 534
Xdm - Indirect access method 101, 534
Xdm Access Control 535
Xdm Access Type field 100, 533
Xdm Direct 101, 534
Xdm KeepAlive retry timeout field 102, 535
Xdm Magic Cookie 535
Xdm Minimum KeepAlive time 535
Xdm Minimum KeepAlive time field 102, 535
Xdm Retry Action 535
Xdm Retry Action field 102, 535
XDMCP (X Display Manager Control Protocol) 100, 533
XDMCP Options 100, 533
XDMCP Options Menu 15, 100, 533
XDMCP Server Address field 101, 534
Xev (X Event) 154
Xev Key Monitoring 154
xgif 113
xlsfonts 171
xmkmf Utility Program 62
xmodmap 108, 152
xmodmap -pk 152
xmodmap -pm 152
xmodmap -pp 152
Xpmroot 382
xrsh Utility Program 62
xscope Diagnostic Utility 62
xset fp 171