NORTEL NETWORKS

Preside Multiservice Data Manager

Customization

Administrator Guide

241-6001-301

Preside Multiservice Data Manager

Customization

Administrator Guide

Publication: 241-6001-301
Document status: Standard
Document version: 14.3RSUP
Document date: December 2003

Printed in Canada

Publication history

December 2003

14.3RSUP Standard
Commercial availability.

Contents

About this document **11**
Who should read this document and why 11
What you need to know 11
How this document is organized 12
Text conventions 12
Related documents 14

Chapter 1
Customizing resources used by MDM tools **15**
About resources used by MDM tools 15
Syntax of a resource definition statement 16
Predominance 17
Specificity 17
Location 19
How specificity and location affect predominance 20
Customizing resources for all users of an MDM tool 20
Example 1, customizing resources for the Network Viewer in the ND tools resource file 21
Example 2, Customizing font resources in the Msm resource file 22
Setting font resources in the Msm resource file 24
Customizing resources for a number of MDM users 24
Setting resources for the Network Viewer in a common resource file 25
Setting resources for a single user in the .Xdefaults file 26
Setting resources for the Network Viewer in the .Xdefaults file 26
Customizing resources in a command line 27

Useful utilities 27
Customizing color maps 28
Setting the color map from the Options menu 29
Selecting the color map with the useMDMColormap resource 30

Chapter 2
Customizing the toolsets and Start Tool menus 31

About menu items that can be customized 31
Preside MDM window 32
Start Tool menus 33
Push-buttons in icon bars 34
Menu definition records 35
File syntax 35
Substitution variables in a Start Tool menu definition file 47
Useful OSF/Motif resources 48
Automatic search path 49
Finding menu definition files for Preside MDM window menus 50
Finding menu definition files for Start Tool menus 52
Where to find icon bar definition files for the Network Viewer icon bars 55
Customizing the toolsets menus 55
Adding a new toolset entry 56
Changing an existing toolset entry or submenu 56
Changing the toolsets primary menu 57
Customizing the Start Tool menus 60
Adding a new entry to a Start Tool menu 60
Changing an existing Start Tool menu 61
Customizing an icon bar 61
Fields in the icon bar definition files 63
Creating a customized icon bar definition file 64
Customization examples 65
Example 1, Setting the startup options in a menu definition file 66
Example 2, Creating a submenu item that invokes a macro 66

Chapter 3
Creating and using macros 69
About macros for MDM 69
Within a user's MDM session 70
Outside a user's MDM session 72
Macro structure 74
Where macros are stored 75
Using the cmccmd program 75
Establishing new a connection with cmccmd 76
Dropping an existing connection with cmccmd 77
Issuing operator commands with cmccmd 78
Listing the available routes with cmccmd 80
Identifying individual command responses 80
Identifying that a connection is available 81
Setting the Command Console mode 81
Writing macros for the Command Console 83
Invoking a macro from the Command Console 83
Running environment for command macros run from the Command Console 84
Examples of macros to be run from the Command Console 86
Invoking macros (that are not stand-alone macros) from outside the Command Console 92
Writing stand-alone command macros 92
The command wrapper program (cmcwrap) 93
Making use of the utilities and macros provided with the MDM software 95
MDM utilities grouped by function 95
Other MDM utilities 104
Macros provided with the MDM software 106

Chapter 4
Creating snmpCmd macros 113
What you need to know 113
Runtime environment 114
Environment variables related to Scotty and TCL 114

snmpCmd macros 115
MIBs 116
Utilities 116
About snmpCmd macros 118
General command macros 119
Device-specific snmpCmd macros 121
What happens when you run an snmpCmd macro 122

Chapter 5
Extracting alarms in text format 127
About the rncsalarm utility 127
Command syntax 128

Chapter 6
Customizing the Component Information Viewer diagnostics 131
Diagnostic menu management 131
CIV diagnostic utilities 133
execWithDest 133
execDPNCommand 135
execPPCommand 136
ppCompSelector 138
execWithIPAddr 138

Index 141

About this document

The following topics are discussed in this section:

- "Who should read this document and why" (page 11)
- "What you need to know" (page 11)
- "How this document is organized" (page 12)
- "Text conventions" (page 12)
- "Related documents" (page 14)

Who should read this document and why

This document contains reference and procedural information for customizing Preside Multiservice Data Manager (MDM) tools and resources and for writing macros. All of these tasks are advanced tasks that are performed relatively rarely by experienced system administrators.

This document is aimed at system administrators who specialize in managing networks.

What you need to know

Users of this document require the following knowledge and skills:

- working knowledge of UNIX, the Solaris operating system, the DPN network and the network control system (NCS), and the Passport network
- knowledge of Preside Multiservice Data Manager (MDM) and how to work with the MDM user interface
- working experience or training in the administration of Sun workstations

- working experience or training in the installation, configuration, and troubleshooting of SunLink X.25, and Frame Relay software products

How this document is organized

241-6001-301 *Preside MDM Customization Administrator Guide* contains the following sections:

- "Customizing resources used by MDM tools" (page 15)contains information that allows you to customize X-window resources and Preside Multiservice Data Manager (MDM) resources to change the appearance of some of the MDM tools to suit your requirements.
- "Customizing the toolsets and Start Tool menus" (page 31) contains information that allows you customize tools menu items to change the user interfaces of some of the MDM tools to suit your requirements.
- "Creating and using macros" (page 69) contains information about creating and using macros in the MDM software environment.
- "Creating snmpCmd macros" (page 113) contains information about creating snmpCmd command macros for devices that are monitored using SNMP protocol.
- "Extracting alarms in text format" (page 127) contains instructions for extracting alarms in text format by means of the *rncsalarm* utility.
- "Customizing the Component Information Viewer diagnostics" (page 131) contains information that lets you customize the Component Information Viewer diagnostics.

Text conventions

This document uses the following text conventions:

- `nonproportional spaced plain type`

  Nonproportional spaced plain type represents system generated text or text that appears on your screen.

- **nonproportional spaced bold type**

  Nonproportional spaced bold type represents words that you should type or that you should select on the screen.

- *italics*

  Statements that appear in italics in a procedure explain the results of a particular step and appear immediately following the step.

  Words that appear in italics in text are for naming.

- [optional_parameter]

  Words in square brackets represent optional parameters. The command can be entered with or without the words in the square brackets.

- <general_term>

  Words in angle brackets represent variables which are to be replaced with specific values.

- UPPERCASE,lowercase

  In Preside Multiservice Data Manager (MDM), uppercase and lowercase letters that appear in UNIX commands and parameters must be matched exactly. The system matches upper and lowercase characters differently.

- |

  This symbol separates items from which you may select one; for example, ON|OFF indicates that you may specify ON or OFF. If you do not make a choice, a default of ON is assumed.

- ...

  Three dots in a command indicate that the parameter may be repeated more than once in succession.

The term absolute pathname refers to the full specification of a path starting from the root directory. Absolute pathnames always begin with the slash (/) symbol. A relative pathname takes the current directory as its starting point, and starts with any alphanumeric character (other than /).

Related documents

See the following documents for related information:

- 241-6001-011 *Preside MDM Fault Management User Guide*
- 241-6001-013 *Preside MDM Remote Network Communication System User Guide*
- 241-6001-015 *Preside MDM Network Model Administrator Guide*
- 241-6001-022 *Preside MDM Network Reporting System User Guide*
- 241-6001-200 *Preside MDM Application Programming Interface Primer*
- 241-6001-201 *Preside MDM Network Model API Reference Guide*
- 241-6001-203 *Preside MDM Alarm and Status API Reference Guide*
- 241-6001-204 *Preside MDM DPN Provisioning API Reference Guide*
- 241-6001-207 *Preside MDM Passport Provisioning API Reference Guide*
- 241-6001-209 *Preside MDM Provisioning Command Filter API Reference Guide*
- 241-6001-211 *Preside MDM Embedded Programming Interface Reference Guide*
- 241-6001-303 *Preside MDM Administrator Guide*
- 241-6001-804 *Preside MDM Workstation Utilities User Guide*

Chapter 1
Customizing resources used by MDM tools

This section contains information that allows you to customize X-window resources and Preside Multiservice Data Manager (MDM) resources to change the appearance of some of the MDM tools to suit your requirements. This section contains the following information:

- "About resources used by MDM tools" (page 15)
- "Syntax of a resource definition statement" (page 16)
- "Predominance" (page 17)
- "Specificity" (page 17)
- "Customizing resources for all users of an MDM tool" (page 20)
- "Customizing resources for a number of MDM users" (page 24)
- "Setting resources for a single user in the .Xdefaults file" (page 26)
- "Customizing resources in a command line" (page 27)
- "Useful utilities" (page 27)
- "Customizing color maps" (page 28)

About resources used by MDM tools

Preside Multiservice Data Manager (MDM) tools are built using X toolkits of the OSF/Motif type. Toolkits provide a means to simplify the design and coding of the software needed to build MDM tools and to ensure consistency in the appearance of the tools and the way in which their buttons and menus operate. Toolkits provide a standard set of objects known as widgets for such things as push-buttons, scroll bars, text fields, and so on. Each widget has a

set of resources associated with it that defines the appearance of the widget and how it acts. For example, resources that define the background color and border width for a push-button in the icon bar of a Preside Multiservice Data Manager (MDM) tool.

There may be widgets within widgets arranged into a hierarchy. For example, the Network Viewer is equipped with many widgets such as an icon bar, and the icon bar has a number of push-buttons that are subwidgets of the icon bar widget.

In addition to the standard X resources for OSF/Motif widgets, MDM software uses many non-widget based X resources for controlling various aspects of the appearance and functions of MDM tools.

Many sources are available to obtain information about setting standard X resources. These include the *X man* pages and third-party publications such as the *X-Window System User's Guide,* ISBN 1-56592-015-5, from O'Reilly and Associates Inc.

The following sections explain the syntax used for writing resource definition statements, provide predominance rules for resource definition statements, explain the methods used to set resources for MDM tools, and contain a few examples that show how to set them.

Syntax of a resource definition statement

The syntax of a statement used to define a resource for an MDM tool is as follows:

```
<client>*<widget>[<b>subwidget>]<b><attribute>:
<value>
```

where:

`<client>` is the client program, or a specific instance of the client program. If you omit the client name, the definition applies to all client programs.

`<widget>` corresponds to the name of a widget for which the value of a resource is being set

<b> is an asterisk (*) or a period (.) character that represents the relationship (the binding) between items such as widgets, subwidgets, and attributes. If there can be no other item between two items in the hierarchy to which the items belong, the items are said to have a tight binding and a period is used to separate them. If other items from the hierarchy can be inserted between the two items, the items are said to have a loose binding and an asterisk is used to separate them.

<subwidget> corresponds to the name of a subcomponent that belongs to the component specified by widget

<attribute> is the name of the resource attribute being set

<value> is the value of the attribute

The following is an example of a resource definition statement that sets a resource for the Network Viewer tool. This statement sets the background color for the title (label) of the Network Viewer window to the color blue.

```
ND*statusLabel.background: blue
```

In this example, ND is the client, statusLabel is the name of the widget, background is the attribute name, and blue is the value of the attribute.

Predominance

Two main factors affect the way in which one resource definition statement overrules (predominates) another. These are:

- the specificity of the resource definition statement
- the location of the resource definition statement

Specificity

Resource definitions can be written with varying degrees of specificity. As an example, the following five definitions can be used to set the color of the label in the main window of the Network Viewer tool. These sample definitions are listed with the most specific statement first and the least specific statement last:

```
ND*statusLabel.background: blue
ND*XmLabel.background: yellow
ND*background: brown
```

```
ND*Background: grey
*XmLabel.background: green
*background: red
*Background: pink
```

The general rule is that the more specific a resource definition statement is, the more likely it is to overrule (predominate) a less specific statement, as follows:

- The more widgets and subwidgets there are in a resource definition statement, the more specific the definition. In the samples, client names are in fact widgets, so definitions that contain client names (ND) predominate definitions that do not contain client names.
- Widget names that begin with an uppercase letter represent classes of widgets and widget names that begin with a lowercase letter are specific instances of a class of widget. Resource definitions for specific instances of a widget predominate resource definitions for classes. In the samples, the definition *background: red predominates the definition *Background: pink and the definition ND*statusLabel.background: blue which is a specific instance of an XmLabel predominates the definition ND*XmLabel.background: yellow
- Bindings between items such as clients, widgets, and attributes are indicated by an asterisk (*) or a period(.). If there can be no other item between two items in the hierarchy to which them items belong, the items are said to have a tight binding and a period is used to separate them. If other items in the hierarchy can be inserted between them, the items are said to have a loose binding and an asterisk (*) is used to separate them. Tight bindings overrule loose bindings.

Location

Although it is possible to define resources for a Preside Multiservice Data Manager (MDM) tool in many locations on UNIX platforms, for user accounts that run the default MDM user environment, resources can be defined in five main locations. These locations and the order in which the system consults them to determine settings for resources are as follows:

1 resources specified by means of *-xrm* arguments added to the command lines used to start an MDM tool.

These arguments can be added to the tool startup commands in the toolsets menu definition files, start tool menu definition files, and icon bar definition files.

2 the .Xdefaults file in a user's home directory

Resource definitions in this file only apply to a single MDM user. Resources definitions in this file are automatically applied to the user' session whenever the user logs in. However, if they are to be applied without having the user log out then log back in again, they must be applied manually by means of the xrdb utility.

3 a common resource definition file, whose definitions are added to a user's session by executing the xrdb -merge command

When this command is added to the .dt/sessions/sessionetc (CDE) file or .xsession (Bourne Korn, or C-shell) file of every user account that is to use the common resource definition file, it applies the definitions to the user's session whenever the user logs in.
Resource definitions applied this way can apply to a single user of an MDM tool, some users of the tool, or all users of the tool. This method is best suited for applying resource definitions to some users of an MDM tool, but not all users.

4 MDM tools resource definition files located in directory /opt/MagellanNMS/cfg/app-defaults

This directory also contains three subdirectories called C, ja, and zh. Subdirectory C is used for storing customized English language resource definition files. Subdirectories ja and zh contain the Japanese and Chinese language equivalents. Resource definitions in these files apply to all users of an MDM tool.

5 MDM tools resource definition files located in directory /opt/MagellanNMS/lib/app-defaults

This directory contains three subdirectories called C, ja, and zh. Subdirectory C contains the default English language resource definition files provided with the MDM software for each of the MDM tools. Subdirectories ja and zh contain the Japanese and Chinese language equivalents. The MDM tools resource definition file are named according to the tool to which they apply. For example, ND is the file name for the Network Viewer. Resource definitions in these files apply to all users of an MDM tool.

How specificity and location affect predominance

When the software obtains the resource definitions from various resource locations, it merges them and applies them according to their specificity as described in “Predominance” (page 17). The most specific definition predominates. However, if a given resource is defined with the same degree of specificity in more than one of the locations listed in “Specificity” (page 17), the software uses the definition it consults last to determine the setting for the resource.

Customizing resources for all users of an MDM tool

Use the information in this section to customize resources for all users of a Preside Multiservice Data Manager (MDM) tool by customizing resource definition files.

Guidelines for setting resources in an MDM tools resource definition file are as follows:

- Do not modify a resource definition file in directories /opt/MagellanNMS/lib/app-defaults/C, /opt/MagellanNMS/lib/app-defaults/ja, or /opt/MagellanNMS/lib/app-defaults/zh These directories are reserved for files that are provided with the MDM software. Instead, copy the file into one of directories /opt/MagellanNMS/cfg/app-defaults/C, /opt/MagellanNMS/cfg/app-defaults/ja, or /optMagellanNMS/cfg/app-defaults/zh and use this new file as the starting point for creating a customized resource definition file.
- At the first line of your new file, use the #include statement to include the original resource file you copied from.

- In any new customized resource definition file you create, only retain statements that define the resources that you wish to set. Remove all other statements from your new file.
- Some tools only allow you to modify a subset of the resources. For information about such restrictions, refer to the Nortel Networks technical publication (NTP) that describes the use of the tool. For example, for the resources you are allowed to modify for the Network Viewer Tool, see 241-6001-011 *Preside MDM Fault Management User Guide*.

Example 1, customizing resources for the Network Viewer in the ND tools resource file

The following procedure is an example of how resources can be set in a resource definition file for all users of a Preside Multiservice Data Manager (MDM) tool, using the Network Viewer tool as an example.

1. Log on as root.
2. Copy the resource file for the Network Viewer tool into a new file in directory /opt/MagellanNMS/cfg/app-defaults/C:

   ```
   cp /opt/MagellanNMS/lib/app-defaults/C/ND \
   /opt/MagellanNMS/cfg/app-defaults/C/ND
   ```

3. Access the directory that contains the copied file.

   ```
   cd /opt/MagellanNMS/cfg/app-defaults/C
   ```

4. Change the file permissions of the newly created file so you can modify it, and to allow read access by the group and by others.

   ```
   chmod 644 ND
   ```

5. Using a UNIX editor, open the new file and change the settings of the resources you wish to modify.

   See 241-6001-011 *Preside MDM Fault Management User Guide* to determine which ND resources you are allowed to modify.

6. At the first line of your new file, use the #include statement to include the original resource file you copied from.
7. Remove all statements from this file, except the ones you have modified.

Note: Another way to proceed is to open the file provided with the MDM software, open a new file in directory /opt/MagellanNMS/cfg/app-defaults/C, copy the statements you wish to modify from the old file into the new file, then modify the resource definitions in the new file.

8 Save the file and exit from it.

9 Have users restart the Network Viewer and verify that the changes are successful.

Example 2, Customizing font resources in the Msm resource file

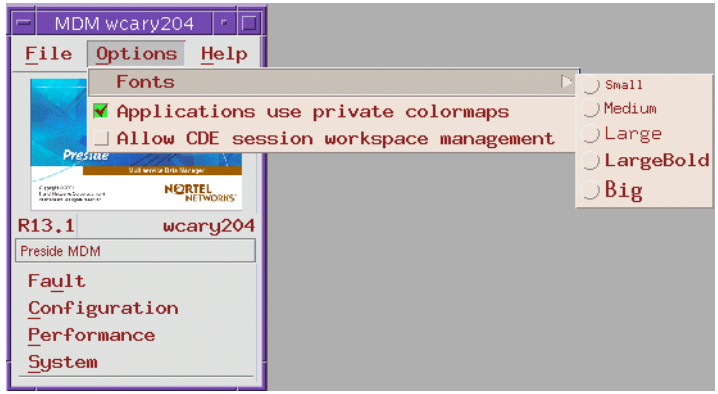
The following example shows how to set resources in a resource definition file for all users of a Preside Multiservice Data Manager (MDM) tool, using the Motif Session Manager (Msm) tool as an example. Most resources in the Msm's resource definition file only apply to the appearance of the cascading menus that are available from the Preside MDM window. However, some of the resources affect all MDM tools that are started from the cascading menus. The following example defines a resource that affects all MDM tools.

The default Msm tool resource file contains font resources that allow an operator to select one of five fonts for displaying information in the main windows of all MDM tools that are opened from the Preside MDM window.

Note: Font changes and substitutions have no effect on Java applications that are part of MDM, such as Backup and Restore, Data Viewer and Nodal Provisioning.

The default font selection menu available to an operator from the Preside MDM window is shown in the figure "Default font selection menu" (page 23).

Figure 1
Default font selection menu



The resource modification in this example defines a new sixth font selection called nicefont.

Three resource definition statements are used to define a font select in the Fonts menu. These are as follows:

```
Msm*numFonts: <number of fonts>
```

defines the maximum number of font selections available from the Fonts menu. In file /opt/MagellanNMS/lib/app-defaults/C/Msm, this resource is set to 5. If you wish to provide more than five choices in the Fonts menu, you must increase this number accordingly.

```
Msm*fontMnemonic<n>: <fontname>
```

defines the name that appears as a selection under the Fonts menu. In this resource, <n> identifies the font selection involved. The value for <n> must lie within the range of numbers defined for resource Msm*numFonts

```
Msm*fontname<n>:<font specification>
```

defines the size and type of the font to be displayed. In this resource <*n*> identifies the font selection involved. The value for <n> must lie within the range of numbers defined for resource Msm*numFonts.

Setting font resources in the Msm resource file

1. Log on as root.
2. Copy the Msm resource file into a new file in directory /opt/MagellanNMS/cfg/app-defaults/C:

   ```
   cp /opt/MagellanNMS/lib/app-defaults/C/Msm \
   /opt/MagellanNMS/cfg/app-defaults/C/Msm
   ```

3. Access the directory that contains the newly copied file:

   ```
   cd /opt/MagellanNMS/cfg/app-defaults/C
   ```

4. Change the file permissions of the newly created file, so you can modify it, and to allow read access by the group and by others.

   ```
   chmod 644 Msm
   ```

5. Enter the following command to display a list of all of the fonts available for use on your workstation, then select one of the fonts for the new menu item.

   ```
   /usr/openwin/bin/xlsfonts | more
   ```

6. Using a UNIX editor, open the new file for editing.
7. Modify resource Msm*numFonts to allow for the creation of a new sixth font selection as follows:

   ```
   Msm*numFonts: 6
   ```

8. Add the two resource definitions to define the new font selection, as follows:

   ```
   Msm*fontMnemonic6: nicefont
   Msm*fontName6: 20x20
   ```

   where:

   20X20

   is one of the fonts that appears when you enter the *xlsfonts* command.
9. Save the file and exit from it.
10. Have users log out and log back in again to make the changes active.

Customizing resources for a number of MDM users

Use the information in this section to set resources for specific users of a Preside Multiservice Data Manager (MDM) tool.

The following example shows how resources for the Network Viewer can be set for a number of MDM users by defining the resources in a common resource file then applying them to each MDM user's account by running the *xrdb* utility whenever the user logs in. To ensure that an MDM user account obtains resource definitions from the common file whenever the user logs in, the following command must be added to the .dt/sessions/sessionetc (CDE) or .xsession (Bourne, Korn, or C-shell) file of every user account that is to use the common resource file:

```
xrdb -merge <pathname of common resource file>
```

Although this method is most useful when you wish to customize resources for some users of an MDM tool, you may use it to customize resources for all users of an MDM tool. Do not customize resources for some MDM tools with this method and customize resources for other tools by creating new customized resource definition files in directories /opt/MagellanNMS/cfg/app-defaults/C, /opt/MagellanNMS/cfg/app-defaults/ja or opt/MagellanNMS/cfg/app-defaults/zh

Setting resources for the Network Viewer in a common resource file

1. Log on as root.
2. Access the directory in which you wish to store the common resource file.

   ```
   cd <directory path name>
   ```

3. Using UNIX editor, open a new file for editing. This file is to contain resource definitions common to the Preside Multiservice Data Manager (MDM) users. For example, the absolute path name for this file is /opt/settings/myfile.
4. Insert statements into the file to define resources for all tools, or for a single MDM tool. For example, to set the color used in the Network Viewer for indicating the in-service state (stateINSV) to white:

   ```
   ND*stateINSV: white
   ```

5. Save the file and exit from it.
6. Change file permissions to allow read access by the group and others, and read-write by the owner.

   ```
   chmod 644 myfile
   ```

7 Do one of the following for each user account that will use the common resource file:

- If the account is running CDE, add the following command to set-up file .dt/sessions/sessionetc:
  xrdb -merge /opt/settings/myfile
- If the account is running Bourne. Korn, or C-Shell add the following command to set-up file .xsession:
  xrdb -merge /opt/settings/myfile

8 Save the file and exit from it.

9 Have each of the users log out and log back again to obtain the new settings, or alternatively have each user enter the following command to activate the settings:

```
xrdb -merge /opt/settings/myfile
```

Setting resources for a single user in the .Xdefaults file

Use the information in this section to set resources for a single Preside Multiservice Data Manager (MDM) user by defining them in the .Xdefaults file of user account's home directory. Resources in this file are automatically applied to the user' session whenever the user logs in, but if they are to be applied without having the user log out then back in again, they must be applied manually by means of the xrdb utility.

The resource set in the following example defines the color used to display the in-service state in the Network Viewer tool.

Setting resources for the Network Viewer in the .Xdefaults file

1 Using a UNIX editor, open the .Xdefaults file in the user's home directory.

2 For every resource specification you want to customize, add a line to this file but make sure that this line is prefixed with the tool's resource file name. For example, to change the IN-SERVICE state color to white for the Network Viewer, add the following line to the .Xdefaults file:

```
ND*stateINSV: white
```

Note: As written, this command changes state and color resources. Omitting *ND* at the beginning of the command allows the changes to state and color resources to apply to both the Network Viewer and Component Information Viewer related components panels.

Seei 241-6001-011 *Preside MDM Fault Management User Guide* for more information about the resources you wish to modify.

3 Save the file and exit from it.

4 Apply the new resource settings to the user's account by entering the following command in a UNIX Access window:

```
xrdb -merge ~/.Xdefaults
```

Customizing resources in a command line

Use the information in this section to set resources in the startup command of a Preside Multiservice Data Manager (MDM) tool.

You can set resources for an MDM tool by adding resource definitions in *-xrm* arguments to the tool's startup command. However, the number of resources that can be set in this way is limited because of the potential length of the command line.

Startup commands for tools that appear in the Preside MDM window cascading menus are contained in toolset definition files. For descriptions of these files, see "Toolset definition files" (page 51).

The following example shows a command that starts the Network Viewer with the labels visible and icons hidden:

```
/opt/MagellanNMS/bin/nd -xrm 'ND*showAllabels: True' \
-xrm 'ND*showAllIcons: False'"
```

Useful utilities

The following utility programs are provided with the Solaris operating system and may be used for such things as setting resources, or determining the values that can be used to set the attributes of a resource. The names of the utilities and their purposes are as follows:

- xrdb is used for two main purposes:
  - updating some users of a Preside Multiservice Data Manager (MDM) tool (but not all users) with resources from a common resource definition file
  - updating a single user of an MDM tool with resource set in the user account's .Xdefaults file

For information about this utility, see the man pages for the xrdb command and for an example of its use to set resources for a number of MDM users, see “Customizing resources for a number of MDM users” (page 24).

- xlsfonts displays the names of fonts available on the workstation. The names displayed can be used as attribute values in resource definition statements for font resources. For information about this utility, see the man pages for the xlsfonts command.
- xco displays the colors available on the workstation and their names. The names displayed can be used as attribute values in resource definition statements for color resources. For information about this utility, see the man pages for the xco command.

Customizing color maps

When Preside Multiservice Data Manager (MDM) is running with other applications that use colors, for example NetScape, there is a risk of running out of color cells in the system’s color map. The MDM tools are designed to reduce this risk by re-using the same color cells. Nevertheless, the more applications you run, the more likely you are to run out of color cells.

There are two main ways to deal with color cell shortages:

- allow the system to run out of color cells (essentially to do nothing)
- use private color maps that are dedicated to the application

When you run out of color cells, some colors may be rendered as different colors. This is not a problem, unless the color has a specific meaning; for example, if the color red indicates an alarm, but is rendered as grey because of a color cell shortage. When this occurs, private color maps must be used to obtain accurate color rendition and meaning.

With private color maps, an application has its own private color map which becomes active whenever you move the cursor into one of the application’s windows (changes the focus). Colors are rendered correctly with this method. When moving the cursor from a window for one application into a window for another application, the colors outside the new application’s window flash to a different set of colors.

The MDM software may be set up to use color maps in the following ways:

- use a common color map for all applications
- use a separate private color map for each MDM tool

There are two ways to set up color maps for MDM tools:

- by means of the Options menu in the Preside MDM window

  See "Setting the color map from the Options menu" (page 29).

- by setting the useMDMColormap resource in the resource file for an MDM tool, in a user account's .Xdefaults file, or in the startup command for an MDM tool

  See "Selecting the color map with the useMDMColormap resource" (page 30).

Setting the color map from the Options menu

Use the following procedure to select a common color map used by all applications, or private color maps for each Preside Multiservice Data Manager (MDM) tool from the Options menu in the Preside MDM window. The selection you make is remembered for your next MDM session.

1

2 Start an MDM session.

See the section on starting and ending MDM sessions in 241-6001-303 *Preside MDM Administrator Guide*.

3 From the Options menu in the Preside MDM window, select Fonts.

4 If a check mark appears beside Applications use private colormaps, each MDM tool uses its own private color map.

If no check mark appears, all MDM tools use a color map that is common to all applications, including MDM.

5 If the setting is not as you desire it, choose Applications use private colormaps with the right mouse button to toggle the check mark on or off.

The selection is complete, and is remembered the next time you start an MDM session.

Selecting the color map with the useMDMColormap resource

Use the following information to set the useMDMColormap resource to allow a Preside Multiservice Data Manager (MDM) tool to use its own private color map. Selecting the color map using the useMDMColormap resource overrides any selection that is made from the Options menu of the Preside MDM window.

To set an MDM tool to use its own private color map, for a single user, add a line similar to the following to the user account's .Xdefaults file:

```
Msm*useNMSColormap: True
```

For an example of how to do this, see "Setting resources for a single user in the .Xdefaults file" (page 26).

To set up the color map in the command line used to launch a tool, use the *-xrm* start up command option to specify the color map resource, similar to the following example:

```
/opt/MagellanNMS/bin/nd -xrm "nd*useNMSColormap: True"
```

Chapter 2
Customizing the toolsets and Start Tool menus

This section shows you how to customize the toolsets and Start Tool menus so that you can launch Preside Multiservice Data Manager (MDM) tools and utilities to suit your requirements. This section contains the following information:

- "About menu items that can be customized" (page 31)
- "Menu definition records" (page 35)
- "Customizing the toolsets menus" (page 55)
- "Customizing the Start Tool menus" (page 60)
- "Customizing an icon bar" (page 61)

About menu items that can be customized

The Preside Multiservice Data Manager (MDM) tools are equipped with several menu items that you can customize. See the following sections for more information on these menu items:

- "Preside MDM window" (page 32)
- "Start Tool menus" (page 33)
- "Push-buttons in icon bars" (page 34)

Note: Although push-buttons in icon bars are not normally thought of as having an association with menus, they are categorized as menu items because they are defined and customized in a similar manner to toolset menus and Start Tool menus.

- "Customizing the Component Information Viewer diagnostics" (page 131)

Preside MDM window

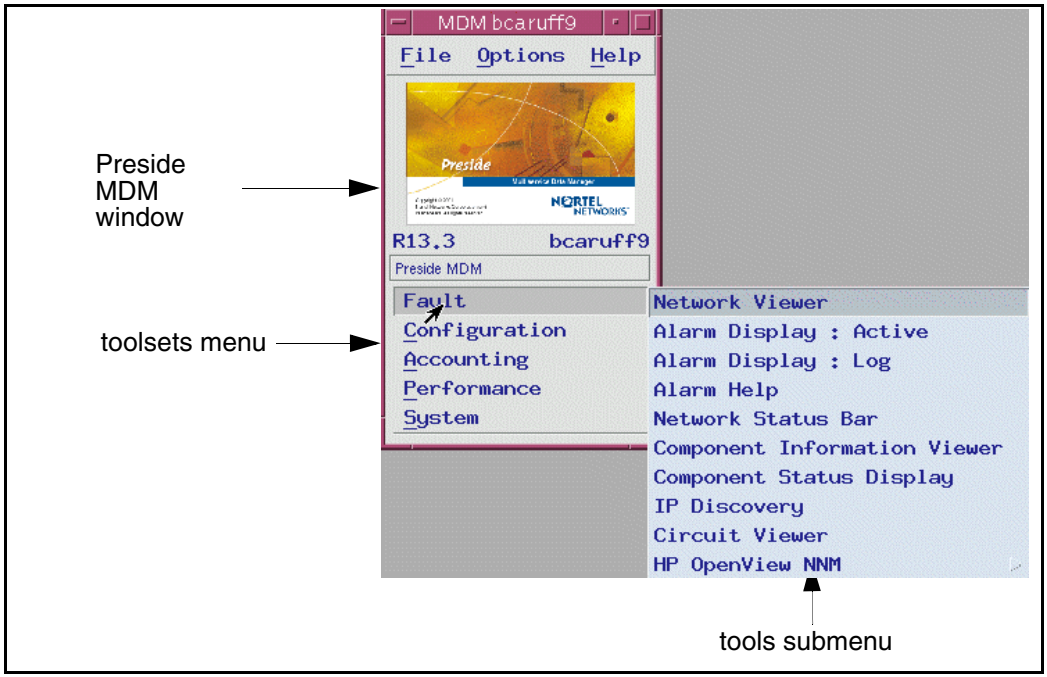
When you log in to a UNIX account that is set up to run the default user environment, as described in 241-6001-303 *Preside MDM Administrator Guide*, a Preside Multiservice Data Manager (MDM) session starts and the Preside MDM window opens. The Preside MDM window provides you with access to the MDM tools by

- a primary menu consisting of the menu items of fault, configuration, accounting, performance and system
- a set of cascading submenus from the primary menu which you can start the tools in the Preside MDM window

For information on customizing the toolsets menus, see "Customizing the toolsets menus" (page 55).

The figure "Preside MDM window and menus" (page 33) shows the Fault menu and its submenus.

Figure 2
Preside MDM window and menus



Start Tool menus

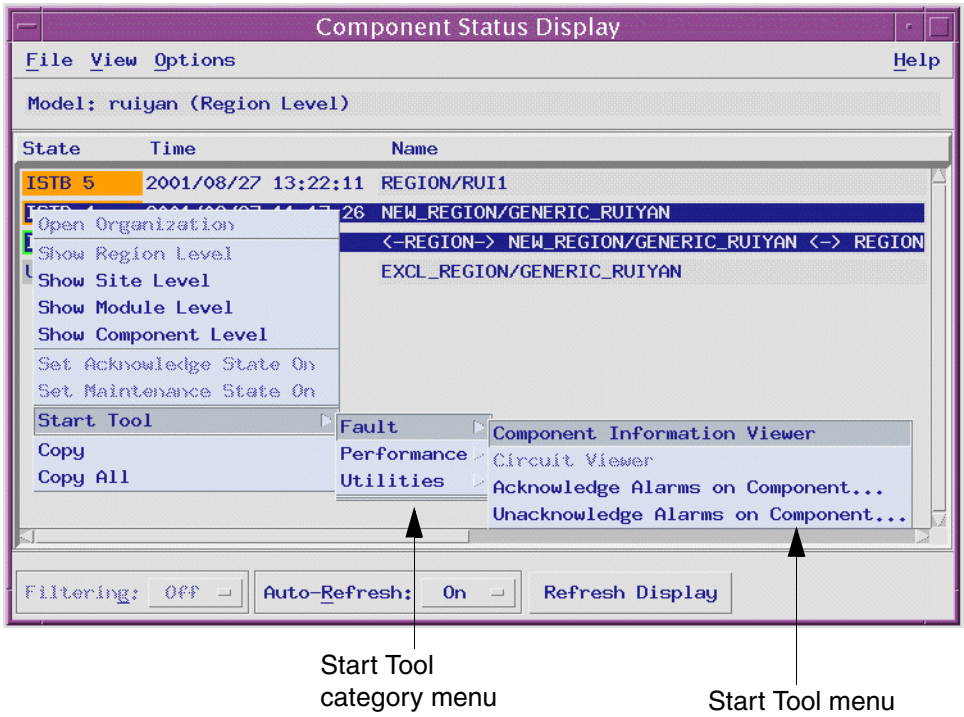
Several Fault tools support one or more pop-up Start Tool menus that let you start another Preside Multiservice Data Manager (MDM) tool from within that Fault tool. Fault tools that support Start Tool menus include the following:

- Network Status Bar
- Network Viewer
- Component Status Display
- Alarm Display
- Component Information Viewer

For information on customizing Start Tool menus, see “Customizing the Start Tool menus” (page 60).

An example of a Start Tool menu is shown in the figure “Sample Start Tool menu for the Component Status Display” (page 34).

Figure 3
Sample Start Tool menu for the Component Status Display



Push-buttons in icon bars

The main window of the Network Viewer has an icon bar. Currently, the Network Viewer is the only tool that has an icon bar. An icon bar contains a set of push-buttons, some of which display an icon to indicate what the push-button does, and others that display a string label.

For information on customizing push-buttons, see “Customizing an icon bar” (page 61).

The figure “Sample icon bar from the Network Viewer main window” (page 35) shows the icon bar.

Figure 4
Sample icon bar from the Network Viewer main window

Menu definition records

Each entry in a Preside MDM window menu or in a Start Tool menu, and each push-button in an icon bar is defined by a set of records contained in a menu definition file. All menu definition records have a common syntax.

For more information, see the following:

- “File syntax” (page 35)
- “Substitution variables in a Start Tool menu definition file” (page 47)
- “Useful OSF/Motif resources” (page 48)
- “Automatic search path” (page 49)
- “Finding menu definition files for Preside MDM window menus” (page 50)
- “Finding menu definition files for Start Tool menus” (page 52)
- “Where to find icon bar definition files for the Network Viewer icon bars” (page 55)

File syntax

In a menu definition file, a record for a menu entry consists of one or more specification lines. General rules for creating such records are as follows:

- each record must be followed by at least one blank line
- each comment line must begin with an exclamation mark (!)
- the information for a record should fit on one line

- file path and script path specifications do not need to start with a directory. If there is no directory, the search paths described in "Automatic search path" (page 49) are invoked.

The records in a menu definition file are partial Motif widget resource descriptions with a few extra resources added. Partial descriptions mean that you only need to specify the resource name and value; the tools menu system manages the widget name itself. See "Customizing resources used by MDM tools" (page 15) for more information about widgets and X resource specifications.

See the following sections for information on the categories into which menu definition records are grouped:

- "Include records" (page 36)
- "Menu entry descriptions (including icon bar push-buttons)" (page 39)
- "Titles" (page 44)
- "Separators" (page 45)
- "Submenus" (page 45)

Include records

These records are used to call up the contents of other files. Include records take one the following forms:

- `#include <file path|GLOB pattern>`
- `#ifinclude <file path|GLOB pattern> <presence test>`

where:

`include` forces the file specified by <file path|GLOB pattern> to be included

`ifinclude` includes the file specified by <file path|GLOB pattern> only if the presence test succeeds

`<file path>` is the pathname of the file to be included. The <file path> can start with a tilde (~) character, which resolves to your current home directory. The absolute file path can also contain the string $LANG. See "Resolving the value of $LANG" (page 49).

`<GLOB pattern>` indicates you can use a series of wildcards to expand the file path. The file path resolves to the first match found for every matching filename found in a specified absolute path or all paths described in "Automatic search path" (page 49) for a specified relative path.

`<presence test>` specifies a test to determine if the menu item represented by the record should be visible in the menu. The supported tests are:

`-f <file path>` checks that the specified file exists

`-r <file path>` checks that the specified file can be read

`-w <file path>` checks that the specified file can be written to

`-x <file path>` checks that the specified file can be executed

`-d <file path>` checks that the specified file is a directory

`-h <file path>` checks that the specified file is a symbolic link

`-e <script path and arguments>` executes the specified script, which should exit with a return code of 0 to indicate success

> ***Note:*** The <file path> and <script path> options can include the tilde (~) character, which resolves to your current home directory. Also, you can include the string $LANG in either of these options. See "Resolving the value of $LANG" (page 49) for more information.

`-E <environment variable> [<value>]` checks to see if the named environment variable exists and if a value is specified, whether the variable's value matches. Two additional environment variables are supported that let many tools use a single menu definition file, but display different menu

entries for different tools. For example, with these variables the entries displayed on a Start Tool menu can vary depending on which Fault tool is invoking the Start Tool menu. These two environment variables are:

- `$APPCLASS` identifies the tool loading the menu.
- `$APPMENU` identifies the menu that is being built.

The table "$APPCLASS and $APPMENU values" (page 39) lists the allowed values of $APPCLASS for toolset menus and Start Tool menus and the allowed values of $APPMENU for Start Tool menus. For Preside MDM window menus, the value of $APPMENU is the name of the toolset definition file, for example, Full.tsets.

You can set the value of $APPMENU in a toolset definition file such as Full.tsets using the tMMenuAppName parameter.

You can also use $@ to test the numerical user-ID value. For example, to allow certain tools to be invoked only by the root user (user-ID 0), use the following test:

```
-E $@ 0.
```

Other environment variables are substituted as usual.

`-L <product license number` or `name prefix>]` checks to see if the product identified either by license number or name prefix is enabled on the workstation. For example, `-L Global Data Manager` verifies if the Global Data Manager product is licensed on the workstation.

By convention, the toolset definition file and tools definition file are the only files that contain *include* and *ifinclude* records. See "Toolset definition files" (page 51) and "Tools definition file" (page 54) for more information.

Table 1
$APPCLASS and $APPMENU values

Tool/Target	$APPCLASS value	$APPMENU value
Component Information Viewer/Alarm List Items	CIV	CIV-Alarm
Component Information Viewer/Related Components List Item	CIV	CIV-Comp
Component Information Viewer/Diagnostic Commands	CIV	CIV-Diag
Component Status Display/Component List Item	CSD	CSD-Comp
Alarm Displays/Alarm List Item	IAD	AD-Alarm
Network Viewer/Icon Bar	ND	ND-Icon
Network Viewer/Editing Mode Icon Bar	ND	ND-IconEdit
Network Viewer/Link Component	ND	ND-Link
Network Viewer/Node Icon	ND	ND-Node
Network Viewer/Subcomponent List Item (Subcomponent dialog) or Button (Shelf dialog)	ND	ND-Sub-comp
Network Status Bar/Component and Subcomponent List Items	StatsBar	StatsBar-Comp

Menu entry descriptions (including icon bar push-buttons)

Menu entries are actually Motif PushButton widgets and therefore support all of the corresponding Motif resources. There are two main forms of records for these entries: with and without bitmap image files. Records that specify bitmap image files are used mainly for defining an icon that appears on a push-button in an icon bar.

The general form without a bitmap image specification is as follows:

```
labelString:          <menu entry name>
tMCommandLine:        <command line to invoke>
tMAction:             <internal tool action to invoke>
tMSelectionPatterns:  <enabling patterns>
tMPresenceTest:       <presence test>
tMPresencePatterns:   <pattern alternatives>
```

```
helpText:              <help information>
<Motif resource line for Push-Buttons>
```

The general form with a bitmap image specification is as follows:

```
labelString:           <menu entry name>
tMCommandLine:         <command line to invoke>
labelType:             pixmap
labelPixMap:           <pixmap path>
tMAction:              <internal tool action to invoke>
tMSelectionPatterns:   <enabling patterns>
tMPresenceTest:        <presence test>
tMPresencePatterns:    <pattern alternatives>
helpText:              <help information>
<Motif resource line for Push-Buttons>
```

As a minimum, you must specify the labelString record, plus one of the tMCommandLine or tMAction lines. Explanations for each of these lines are as follows:

```
labelString:               <menu entry name>
```

is the name of an entry in a menu or the name of a push-button in an icon bar. Some Fault tools allow the use of a substitution variable in this string. See "Substitution variables in a Start Tool menu definition file" (page 47) for information about the use of substitution variables in these lines.

```
labelType:                 pixmap
```

is used in icon bars to specify that a push-button should be displayed as an icon instead of as text. If this line is specified, the labelPixMap line must also be specified.

```
labelPixMap:               <pixmap path>
```

is used in icon bars to specify the full path name of the bitmap image to display as an icon on a push-button. If this line is specified, the labelType line must also be specified. The icon must be in X-11 Bitmap format.

```
tMCommandLine:             <command line to invoke>
```

is the command line to execute when the menu entry or the push-button is selected. This string is in Bourne shell syntax and executed through the Motif Session Manager (MSM). The MSM label in the Preside MDM window displays a response to indicate when a tool is started.

For some Fault tools, this line can contain a substitution variable that passes the name of a component or other information along to the tool being started. See "Substitution variables in a Start Tool menu definition file" (page 47) for information about the use of substitution variables in these lines.

This line can also be used to invoke a macro. For an example, see "Example 2, Creating a submenu item that invokes a macro" (page 66).

If this line is specified, omit the tMAction line.

```
tMAction:                <internal tool action to invoke>
```

identifies an internal action to perform and its parameter. The internal action depends on the MDM tool. See 241-6001-011 *Preside MDM Fault Management User Guide* for more information.

For some Fault tools, this line can contain a substitution variable that passes the name of a component or other information along to the tool being started. See "Substitution variables in a Start Tool menu definition file" (page 47) for information about the use of substitution variables in these lines.

If this line is specified, the tMCommandLine line must be omitted.

```
tMSelectionPatterns:       <enabling patterns>
```

are pattern alternatives similar to those provided with the UNIX egrep command that are used to enable or disable the menu item for certain components. For example, the pattern ^PM.*|^OA.*enables the menu item for DPN-100 modules (PMs) and operations agents (OAs). If this line is omitted, the menu items is enabled for all components.

```
tMPresenceTest:            <presence test>
```

specifies a test to be used to determine if an entry should appear in a menu. The supported tests are:

`-f <file path>` checks that the specified file exists

`-r <file path>` checks that the specified file can be read

`-w <file path>` checks that the specified file can be written to

`-x <file path>` checks that the specified file can be executed

`-d <file path>` checks that the specified file is a directory

`-h <file path>` checks that the specified file is a symbolic link

`-e <script path and arguments>` executes the specified script, which should exit with a return code of 0 to indicate success

> ***Note:*** The <file path> and <script path> options can include the tilde (~) character, which resolves to your current home directory. Also, you can include the string $LANG in either of these options. See "Resolving the value of $LANG" (page 49) for more information.

`-E <environment variable> [<value>]` checks to see if the named environment variable exists and if a value is specified, whether the variable's value matches. Two special environment variables are supported that let many tools use a single menu definition file, but display different menu entries for different tools. For example, with these variables the entries displayed on a Start Tool menu can vary depending on which Fault tool is invoking the Start Tool menu. These two environment variables are:

- `$APPCLASS` identifies the tool loading the menu
- `$APPMENU` identifies the menu that is being built

The table "$APPCLASS and $APPMENU values" (page 39) lists the allowed values of $APPCLASS for Preside MDM window menus and Start Tool menus and the allowed values of $APPMENU for Start Tool menus. For Preside MDM window menus, the value of $APPMENU is the name of the toolset definition file, for example, Full.tsets.

You can set the value of $APPMENU in a toolset definition file such as Full.tsets using the tMMenuAppName parameter.

You can also use $@ to test the numerical user-ID value. For example, to allow certain tools to be invoked only by the root user (user-ID 0), use the following test:

-E $@ 0.

Other environment variables are substituted as usual.

`-L <product license number` or `name prefix>]` checks to see if the product identified either by license number or name prefix is enabled on the workstation. For example, `-L Global Data Manager` verifies if the Global Data Manager product is licensed on the workstation.

Multiple tests can be strung together with && (AND) and || (OR) operators. For example, the following test includes a tool in the menu if the executable is found and the current DISPLAY is not the main console:

```
tMPresenceTest: -x /opt/MagellanNMS/bin/svmadm \
          && ^ -E DISPLAY ":0.0"
```

`tMPresencePatterns:` `<pattern alternatives>`

are pattern alternatives that are used to make an menu entry visible or hidden. Unlike a presence test, a presence pattern does not hide a menu entry permanently. If a presence pattern is specified, the menu item is visible only if the target component ID matches one of the pattern alternatives. Presence patterns are only supported in Start Tool menus, where a component ID context exists.

`helpText:` `<help information>`

is used in icon bars to specify a short string of text that is to be displayed in a dialog when the user selects context sensitive help from a menu, then moves the cursor to the push-button and clicks the select mouse button.

`<Motif resource line for PushButtons>`

resources for Motif PushButtons that specify the appearance of the menu item or push-button. See “Useful OSF/Motif resources” (page 48).

Titles

Titles are actually Motif Label widgets and therefore support all of the corresponding resources. The general syntax for a title record is as follows:

```
labelString:     <title name>
tMMenuAppName:   <$appmenu env. variable>
tMPresenceTest:  <presence test>
<Motif resource line for Labels>
```

Only the labelString line needs to be specified in a title. Explanations for each of these lines are as follows:

`labelString:` `<title name>`

is the title that appears in the menu. Some Fault tools allow you to use a substitution variable in this string. See 241-6001-011 *Preside MDM Fault Management User Guide* for information about the use of substitution variables in these lines.

`tMMenuAppName:` `<$appmenu env. variable>`

is used to set the value of the environment variable $APPMENU. For more information, see the explanation given on page 37 with
-E <environment variable> [<value>].

`tMPresenceTest:` `<presence test>`

specifies a test to be used to determine if an item should appear in a menu. A number of presence tests are possible. For information about them, see “Menu entry descriptions (including icon bar push-buttons)” (page 39).

`<Motif resource line for Labels>`

are resources for Motif Labels that specify the appearance of the title. See “Useful OSF/Motif resources” (page 48).

Separators

Separators are actually Motif Separator widgets and therefore support all of the corresponding resources. The general syntax for a separator record is as follows:

```
tMSeparator:
tMPresenceTest:       <presence test>
<Motif resource line for separators>
```

Only the tMSeparator line needs to be specified. Explanations for each of these lines are as follows:

```
tMSeparator:
```

indicates only that the record is for a separator. By default, this is a single horizontal line.

```
tMPresenceTest:             <presence test>
```

specifies a test to be used to determine if an item should appear in a menu. A number of presence tests are possible. For information about them, see “Menu entry descriptions (including icon bar push-buttons)” (page 39).

```
<Motif resource line for Separators>
```

resources for Motif Separators that specify the appearance of the separator. See “Useful OSF/Motif resources” (page 48).

Submenus

Submenus begin with a submenu header record and end with a submenu footer record. Submenus can be nested one within another, each of which must be started with its own submenu header record and terminated by its own submenu footer record.

Submenu headers are in fact Motif CascadeButton widgets and therefore support all of the corresponding resources. The general syntax for a submenu header record is as follows:

```
tMSubMenu:              <submenu name>
labelString:            <submenu title>
tMMenuAppName:          <$appmenu env. variable>
tMSelection Patterns:   <enabling patterns>
```

```
tmPresenceTest:          <presence test>
tMPresencePatterns:      <pattern alternatives>
<Motif resource line for Cascade Buttons>
```

The general syntax for a submenu footer record is as follows:

```
tMEndSubMenu:            <submenu name>
```

Only the tMsubMenu and labelString lines need to be specified in a submenu header record. Explanations for lines in submenu header and footer records are as follows:

```
tMSubMenu:               <submenu name>
```

introduces the submenu. The submenu must be terminated by a tMEndSubMenu record.

```
labelString:             <title name>
```

is the name of the submenu as it is displayed in the menu button of the menu that opens the submenu. Some Preside Multiservice Data Manager (MDM) tools allow you to specify a substitution variable name in this string that is replaced by the appropriate value when the menu is displayed.

tMMenuAppName: <$appmenu env. variable>

is used to set the value of the environment variable $APPMENU. For more information, see the explanation given on page 37 with -E <environment variable> [<value>].

```
tMSelectionPatterns:     <enabling patterns>
```

are pattern alternatives similar to those provided with the UNIX egrep command that are used to enable or disable the entire submenu in certain contexts. For example, the pattern ^PM.*|^OA.*enables a command for DPN-100 modules (PMs) and operations agents (OAs). If this line is omitted, all components are enabled.

```
tMPresenceTest:             <presence test>
```

specifies an optional test to be used to determine if the entire submenu should be included in the menu, at all. The supported tests are identical to those for line tMPresenceTest described in "Menu entry descriptions (including icon bar push-buttons)" (page 39).

```
tMPresencePatterns:        <pattern alternatives>
```

are pattern alternatives that are used to make an menu entry visible or hidden. Unlike a presence test, a presence pattern does not hide a menu entry permanently. If a presence pattern is specified, the menu entry is visible only if the target component ID matches one of the pattern alternatives. Presence patterns are only supported in Start Tool menus, where a component ID context exists.

```
<Motif resource line for Cascade Buttons>
```

resources for Motif CascadeButtons that specify the appearance of the menu item that causes the submenu to appear. See "Useful OSF/Motif resources" (page 48).

```
tMEndSubMenu:                <submenu name>
```

ends the submenu begun by the tMSubMenu record.

Substitution variables in a Start Tool menu definition file

The following records are an extract from a menu definition file that defines the Start Tool menu that opens when you select a node in the Network Viewer and press the select mouse button. The records in the extract define the menu item that calls up the Component Provisioning tool:

```
labelString:              Component Provisioning
tMCommandLine:            /opt/MagellanNMS/bin/pui \
                          -defaultComponent "$DCOMP"
tMSelectionPatterns:      ^PM.*|^EM*
tMPresenceTest:           -x /opt/MagellanNMS/bin/
ppsxterm
```

Substitution variables, such as $DCOMP (shown in the extract) can be added to some records in Start Tool menu definition files. These variables are used for such things as replacing the name of a component in a startup command with the name of a component an operator has selected in the Network Viewer

main window. The substitution variables allowed depend on the type of record (labelString, tMCommandLine, or tMAction) and the tool. For a list of the substitution variables and the records in which they can be used for various Preside Multiservice Data Manager (MDM) tools, see the following tools in 241-6001-011 *Preside MDM Fault Management User Guide*:

- Network Status Bar
- Network Viewer
- Component Information Viewer
- Component Status Display
- Alarm Display

By convention, only the records in the extract are used in Start Tool menu definition files.

Useful OSF/Motif resources

The following OSF/Motif resources are common to the OSF/Motif widgets used in menus and icon bar push-buttons, titles, separators, and submenus:

```
background: <background color>
```

controls the background color of a tools menu entry or a push-button

```
foreground: <foreground color>
```

controls the foreground color of a tools menu entry or a push-button

```
fontList: <font list specification>
```

controls the font used for in tools menu entry or push-button

```
alignment:<alignment_beginning|alignment_center
|alignment_end>
```

controls the way in which the tools menu entry or push-button is displayed: left justified, centered, or right-justified

```
shadowThickness: <separator thickness>
```

is used in separators to control the thickness of the separator bar

Automatic search path

If you do not specify a full path for the menu definition files, the following standard directories are searched in the following order:

1 $HOME/MagellanNMS
This path is for customized menu definition files for a single Preside Multiservice Data Manager (MDM) user.

2 /opt/MagellanNMS/cfg/tsets/$LANG
This path is for customized menu definition files for the workstation.

3 /opt/MagellanNMS/lib/tsets/$LANG
This path contains the menu definition files provided with the Preside Multiservice Data Manager (MDM) software. The menu definition files that come with the MDM software should never be altered.

The tools menu system applies search paths, $LANG substitutions, and GLOB pattern matching and picks the first found match for each resolved filename. This search pattern means that user customizations take precedence over workstation customizations, which take precedence over MDM defaults.

See “Resolving GLOB patterns” (page 49) and “Resolving the value of $LANG” (page 49) for more information on how pathnames are resolved.

Resolving GLOB patterns

You can also use GLOB style patterns to identify file locations. When you use GLOB style patterns, all matching files are loaded in alphanumeric order as if they were included individually. The filenames are scanned until a match is found.

Resolving the value of $LANG

When the string $LANG appears in a pathname, it resolves to one of the following values in this lookup order:

1 The current setting of the LANG environment variable. The value of this environment variable can be C for the English language toolset files, ja for the Japanese language toolset files, or zh for the Chinese language toolset files.

2 The value C if the LANG environment variable exists but no match is found with the values described in the previous step.

3 Nothing, if the LANG environment variable does not exist. In this case, the string $LANG is ignored.

Finding menu definition files for Preside MDM window menus

The menu definition files for toolsets menus are grouped in several subdirectories. Use the following path to find the menu definition files:

```
<search path>/toolsets/fcaps/<area>/<filename>
```

where:

`search path` is one of the paths described in "Automatic search path" (page 49). The path /opt/MagellanNMS/lib/tsets/C contains the default menu definition files that come with the Preside Multiservice Data Manager (MDM) software.

`area` is one of the subdirectories that contain menu definition files for entries on the Preside MDM window primary menu. See "Toolset definition files" (page 51) for information on how these subdirectories are ordered for display purposes. The following subdirectories come with the MDM software:

`fault` contains menu definition files for fault menu items. For example Network Status Bar.

`configuration` contains menu definition files for configuration menu items. The tools are organized by device type.

`accounting` contains menu definition files for Management Data Provider (MDP) menu items.

`performance` contains menu definition files for performance menu items.

`security` contains menu definition files for security menu items.

`administration` contains menu definition files for the administration menu entries. For example System -> Administration -> System Log Display.

`utilities` contains menu definition files for utilities menu items. For example System -> Utilities -> Customer Data.

`help` contains menu definition files for help menu items.

`filename` is one of the menu definition files. A menu definition file contains the records that define one entry on the Preside MDM window primary menu and the submenus that cascade from the entry. The names of the menu definition files begin with a number, which determines the order in which the menu entries are displayed on the Preside MDM window primary menu. Menu definition files for the Preside MDM window menus have the extension *.tools*.

Example

/opt/MagellanNMS/lib/tsets/C/toolsets/fcaps/fault/30fault.tools contains the definition for the general fault tools.

/opt/MagellanNMS/lib/tsets/C/toolsets/fcaps/fault/50circuit.tools contains the definition for the Fault -> Circuit Viewer tool.

/opt/MagellanNMS/lib/tsets/C/toolsets/fcaps/configuration/60service.tools.

/opt/MagellanNMS/lib/tsets/C/toolsets/fcaps/fault/70hpovdt.tools contains the definition for the menu Fault -> HP OpenView NNM.

The entry Fault -> Circuit Viewer precedes the entry HP-OV NNM, but follows all other Fault tool entries on the Fault menu because of the alphabetical order of the filenames (for example, 30fault.tools would precede 50circuit.tools which would precede 70povdt.tools).

Toolset definition files

The order of the entries on the Preside MDM window primary menu is determined by the alphabetical order of the menu defintion files. The toolset definition file influences the order of the entries as well as what entries are visible. There is only one toolset definition file used in a Preside Multiservice Data Manager (MDM) session.

Toolset definition files are written using the same syntax as menu definition files. In a toolset definition file, the *area* subdirectories are listed in the order in which their menu definition files appear on the Preside MDM window primary menu.

Use the following path to find the toolset definition files:

```
<search path>/<filename>
```

where:

`search path` is one of the paths described in "Automatic search path" (page 49). The path /opt/MagellanNMS/lib/tsets/C contains the default toolset definition files that come with the MDM software.

`filename` is one of the toolset definition files. The default toolset definition used by MDM is /opt/MagellanNMS/lib/tsets/C/Full.tsets. By default, Full.tsets is used unless the environment variable NMSTSETS is defined. The MDM software provides a number of toolset definition files that you can use and they are:

`Full.tsets` is the default toolset definition file. `Admin.tsets`, `NMSAdmin.tsets`, and `User2.tsets` are the same as the Full toolset. These toolsets make all tolls available to the users.

`User.tsets` removes access to some administration tools.
`Web.tsets` removes access to UNIX by way of the System ->Utilities -> UNIX Access.

Finding menu definition files for Start Tool menus

The menu definition files for Start Tool menus are grouped in several subdirectories. Use the following path to find the menu definition files:

```
<search path>/tools/<application area>/<filename>
```

where:

`search path` is one of the paths described in "Automatic search path" (page 49). The path /opt/MagellanNMS/lib/tsets/C contains the default menu definition files that come with the Preside Multiservice Data Manager (MDM) software.

`application area` is one of the subdirectories that contain menu definition files for entries on the Start Tool menus. See "Tools definition file" (page 54) for information on how these subdirectories are ordered for display purposes. The following subdirectories come with the MDM software:

`top` contains menu definition files for menu entries you add to the top of the Start Tool menu. Until you add menu definition files to it, this subdirectory is empty.

`surv` contains menu definition files for the Fault menu entry on the Start Tool category menu. For example, *Component Information Viewer* and *Alarm Help*.

`prov` contains menu definition files for the Configuration menu entry on the Start Tool category menu. For example, *Nodal Provisioning* and *Network Reporting System*.

`misc` contains menu definition files for the Utilities menu entry on the Start Tool category menu. For example, *Customer Data* and *Command Console*.

`custom` contains menu definition files for the Custom menu entry on the Start Tool category menu. Until you add menu definition files to it, this subdirectory is empty and the Custom menu entry is not displayed.

`bottom` contains menu definition files for menu entries you add to the bottom of the Start Tool menu. Until you add menu definition files to it, this subdirectory is empty.

`filename` is one of the menu definition files. A menu definition file contains the records that define one entry on the Start Tool menu and any submenus that cascade from the entry. The names of the menu definition files begin with a number, which determines the order in which the menu entries are displayed on the Start Tool menu. Menu definition files for the Start Tool menus have the extension *.menu*.

Example

In the Configuration menu entry of the Start Tool menu, /opt/MagellanNMS/lib/tsets/C/tools/prov/10arch_pp.menu contains the menu definition records for the entry *Nodal Provisioning,* and /opt/MagellanNMS/lib/tsets/C/tools/prov/10arch_pp.menu contains the menu definition records for the entry *Network Reporting System* menu entries.

The entry *Nodal Provisioning* precedes the entry *Network Reporting System* on the Start Tool menu because of the alphanumeric order of the menu definition files.

Tools definition file

The order of the entries on the Start Tool menu is determined not only by the alphanumeric order of the menu definition files.

The tools definition file is written using the same syntax as menu definition files. In a tools definition file, the application area subdirectories as described in "Finding menu definition files for Start Tool menus" (page 52) are listed in the order in which their menu definition files appear in the Start Tool menu. For example, in the figure "Sample Start Tool menu for the Component Status Display" (page 34), the tools definition file would list the application area subdirectories in this order: top (empty), surv, prov, misc, custom (empty), and bottom (empty). This order causes the entries for the surveillance tools to be listed first, followed by the provisioning tools, and the miscellaneous tools.

Use the following path to find the tools definition file:

```
<search path>/NMSToolsMenu.menu
```

where:

`search path` is one of the paths described in "Automatic search path" (page 49). The path /opt/MagellanNMS/lib/tsets/C contains the default tools definition file that comes with the MDM software.

Where to find icon bar definition files for the Network Viewer icon bars

Icon bars are defined in icon bar definition files, which use the same syntax as menu definition files. Network Viewer has two icon bars but only one is displayed at a time.

Use the following path to find the icon bar definition files:

```
<search path>/<filename>
```

where:

`search path` is one of the paths described in "Automatic search path" (page 49). The path /opt/MagellanNMS/lib/tsets/C contains the default icon bar definition files that come with the Preside Multiservice Data Manager (MDM) software.

`filename` is one of the icon bar definition files.

The default icon bar definition files that come with the MDM software are:

- /opt/MagellanNMS/lib/tsets/C/NDIconBar.menu
  This icon bar file defines the icon bar displayed while the Network Viewer is being used in surveillance mode.
- /opt/MagellanNMS/lib/tsets/C/NDIconBarEdit.menu
  This icon bar file defines the icon bar displayed while the Network Viewer is being used in Network Model Edit mode.

Customizing the toolsets menus

This section describes the procedures you follow when you want to customize the toolsets menus. When you customize the toolsets menus, it is important to reassess your changes when a new release of Preside Multiservice Data Manager (MDM) is installed because the new release may impact your changes. For example, the new release may have new tools that you want to add to your customized files. Also, some tools may not exist. See the MDM Release Supplement that accompanies the software release for details of such changes.

You can access your changes by using the diff or sdiff utilities that come with Solaris. For more information, refer to the man pages for diff and sdiff.

See the following sections for toolsets customization procedures:

- "Adding a new toolset entry" (page 56)
- "Changing an existing toolset entry or submenu" (page 56)
- "Changing the toolsets primary menu" (page 57)

Adding a new toolset entry

Follow this procedure to add a new entry to the toolsets primary menu and the submenu that cascades from the entry.

1 Create a new menu definition file and place it in one of the following directories:

   - $HOME/MagellanNMS/toolsets/<area>
     Use this directory for a new toolset submenu for a single Preside Multiservice Data Manager (MDM) user.
   - /opt/MagellanNMS/cfg/tsets/$LANG/toolsets/<area>
     Use this directory for a new toolset submenu for the workstation.

   See "Finding menu definition files for Preside MDM window menus" (page 50) for information on resolving the values of the variables in the pathnames.

   Make sure that the name of the new menu definition file begins with a number that places the entry where you want in the toolsets primary menu. For example, if you want to add an entry between Configuration -> DPN Devices (whose menu definition file is 20dpn_config.tools), the name of your new menu definition file must begin with a number between 30 and 35.

2 Add the menu definition records using the syntax described in "File syntax" (page 35).

Changing an existing toolset entry or submenu

Follow this procedure to change an existing menu definition file in order to either change an entry on the toolsets primary menu or the contents of a submenu.

1 Copy the menu definition file that you need to change to one of the following directories:

- $HOME/MagellanNMS/toolsets/<area>
  Use this directory for a change that affects a single Preside Multiservice Data Manager (MDM) user.
- /opt/MagellanNMS/cfg/tsets/$LANG/toolsets/<area>
  Use this directory for a change that affects all users of the workstation.

See “Finding menu definition files for Preside MDM window menus” (page 50) for information on resolving the values of the variables in the pathnames.

2 Edit the menu definition records using the syntax described in “File syntax” (page 35).

Changing the toolsets primary menu

Follow this procedure when you want to use a toolset definition file other than the default /opt/MagellanNMS/lib/tsets/C/Full.tsets. The toolset definition file lists the *area* subdirectories and these subdirectories contain the menu definition files that make up the entries on the toolsets primary menu.

1 Copy an existing toolset definition file or create a new file in one of the following directories:

- $HOME/MagellanNMS/toolsets
  Use this directory for a change that affects a single Preside Multiservice Data Manager (MDM) user.
- /opt/MagellanNMS/cfg/tsets/$LANG
  Use this directory for a change that affects all users of the workstation.

See “Finding menu definition files for Preside MDM window menus” (page 50) for information on resolving the values of the variables in the pathnames.

2 Edit the menu definition records using the syntax described in “File syntax” (page 35). If you are not longer using Full.tsets, make sure you adjust the value of the tMMenuAppName entry to match the name of the new file.

3 If you are no longer using Full.tsets, change the value of the environment variable NMSTSETS to the name of the new toolset definition file. See “Changing the default toolset definition file” (page 58).

Changing the default toolset definition file

The value of environment variable NMSTSETS determines the toolset that is displayed when you log in to Preside Multiservice Data Manager (MDM). The value of this variable is specified in set-up file .cshrc for UNIX user accounts that run C-shell and in set-up file .profile for accounts that run Bourne or Korn shell. By default, these set-up files provide you with access to the toolset included in the toolset definition file /opt/MagellanNMS/lib/tsets/C/Full.tsets. See "Toolset definition files" (page 51) for a description of this file and the other toolset definition files that come with the MDM software.

You can set the value of the NMSTSETS variable in the .cshrc or .profile set-up files so that it specifies any of the toolsets provided with the MDM software, or a toolset that you create yourself. The specification can be written in the following forms:

- For accounts that run Bourne or Korn shell:

NMSTSETS=<tset name>
export NMSTSETS

Examples:

```
NMSTSETS=Admin.tsets
export NMSTSETS

NMSTSETS=<absolute pathname>
export NMSTSETS
```

Example: Admin.tsets for the Japanese language,

```
NMSTSETS =/opt/MagellanNMS/lib/tsets/ja/Admin.tsets
export NMSTSETS
```

- For accounts that run C-shell:

setenv NMSTSETS <tset name>

For example: `NMSTSETS Admin.tsets`

setenv NMSTSETS <absolute pathname>

Example: Admin.tsets for the Chinese language,
`setenv NMSTSETS /opt/MagellanNMS/lib/tsets/zh/Admin.tsets`

Use the following procedure to specify the toolset that appears when you log in to an account that is set up to run the default MDM user environment.

1. Log in to the UNIX account that is set up with the default MDM environment.
2. Using a UNIX editor such as vi, open one of the following files for editing:
   - .cshrc for UNIX accounts that run C-Shell
   - .profile for accounts that run Bourne or Korn shell
3. Change the value for the NMSTSETS variable to the name of the toolset that is to appear when the user logs in. For a descriptions of the existing toolsets, see “Toolset definition files” (page 51).

   For example:

   - For accounts that run C-Shell:

   ```
   setenv NMSTSETS Admin.tsets
   or
   setenv NMSTSETS /opt/commonspecs/Harry.tsets
   ```

   - For accounts that run Korn or Bourne shell:

   ```
   NMSTSETS=Admin.tsets
   export NMSTSETS
   or
   NMSTSETS=/opt/commonspecs/Harry.tsets
   export NMSTSETS
   ```

4. Save the file and exit from the file.
5. Log out and log back in again. When logging in be sure to choose the language C, ja, or zh from the Options menu button on the login panel.

   The Preside MDM window opens.
6. Move to the Preside MDM window.
7. Using the menu mouse button, pull down each of the menus and submenus, and verify that the correct set of tools is displayed.

Customizing the Start Tool menus

This section describes the procedures you follow when you want to customize Start Tool menus. When you customize Start Tool menus, it is important to reassess your changes when a new release of Preside Multiservice Data Manager (MDM) is installed because the new release may impact your changes. For example, the new release may have new tools that you want to add to your customized files. Also, some tools may no longer exist. See the MDM Release Supplement that accompanies the software release for details of such changes.

You can access your changes by using the diff or sdiff utilities that come with Solaris. For more information, refer to the man pages for diff and sdiff.

See the following sections for Start Tools customization procedures:

- "Adding a new entry to a Start Tool menu" (page 60)
- "Changing an existing Start Tool menu" (page 61)

Adding a new entry to a Start Tool menu

Follow this procedure to add a new entry or entries to a Start Tool menu.

1. Create a new menu definition file and place it in one of the following directories:

   - $HOME/MagellanNMS/tools/<application area>
     Use this directory for a new Start Tool entry for a single Preside Multiservice Data Manager (MDM) user.
   - /opt/MagellanNMS/cfg/tsets/$LANG/tools/<application area>
     Use this directory for a new Start Tool entry for the workstation.

   See "Finding menu definition files for Start Tool menus" (page 52) for information on resolving the values of the variables in the pathnames.

   Make sure that the name of the new menu definition file begins with a number that places the entry where you want in the Start Tool menu. For more information see the example in "Finding menu definition files for Start Tool menus" (page 52).

2. Add the menu definition records using the syntax described in "File syntax" (page 35).

Changing an existing Start Tool menu

Follow this procedure to change an existing entry or entries on a Start Tool menu.

1. Copy the menu definition file that you need to change to one of the following directories:
   - $HOME/MagellanNMS/tools/<application area>
     Use this directory for change that affects a single Preside Multiservice Data Manager (MDM) user.
   - /opt/MagellanNMS/cfg/tsets/$LANG/tools/<application area>
     Use this directory for a change that affects all users of the workstation.

   See “Finding menu definition files for Start Tool menus” (page 52) for information on resolving the values of the variables in the pathnames.
2. Edit the menu definition records using the syntax described in “File syntax” (page 35).

Customizing an icon bar

The main window of the Network Viewer has an icon bar. See “Where to find icon bar definition files for the Network Viewer icon bars” (page 55) for information on the location of the files that define the contents of the icon bars.

Bitmap images for icons that are provided with the Preside Multiservice Data Manager (MDM) software are contained in directory /opt/MagellanNMS/lib/nvs/icons. You can use these as a starting point and edit them with a bitmap editor such as CDE’s Icon Editor, or the one provided with Solaris, located in file /usr/openwin/bin/bitmap. Or you can use your own bitmaps. The bitmaps must be in X-11 monochrome format.

The two following sets of records are extracted from file /opt/MagellanNMS/lib/tsets/C/NDIconBar.menu. The first set of records defines a pushbutton that is equipped with an icon to indicate its function, and the second set defines a push-button that is equipped with a stringLabel to indicate its function.

```
! Expand in Place icon:
! This item expands the selected nodes in place.
!-------------------------------------------------
```

```
labelString:        Expand in Place
labelType:          pixmap
labelPixmap:        /opt/MagellanNMS/lib/nvs
                    /icons/IBExpand.xbm
tMAction:           expand
helpText:           selecting this item will ...if any


! CIV item:
! The following item will invoke the Component
Information
! viewer for the targeted node. The -Q option will
start
! CIV with the auto-context turned on
!-------------------------------------------------
labelString:        CIV
tMCommandLine:      /opt/MagellanNMS/bin/civ -Q
helpText:           Selecting this item will...if any.
```

These extracts are only intended as a representative examples. For information about other records that can be inserted icon bar definition files, see “Menu definition records” (page 35).

For the Network Viewer, several actions can be included in tMAction records for push-buttons. When a user of the Network Viewer selects a node in the Network Viewer main window, then selects the push-button in the icon bar that bears the expand in place, the tMAction: expand record shown in the extract expands the node that is selected in place.

Substitution variables can also added to some records in icon bar files. These variables are used for such things as replacing the name of a component in a startup command with the name of a component an operator has selected in the Network Viewer main window.

For information about substitution variables that can be inserted into icon bar definition files, and for a list and an explanation of the actions that can be inserted in tMAction records for push-buttons, see the section below, “Fields in the icon bar definition files” (page 63).

Fields in the icon bar definition files

The syntax for the fields in the icon bar definition files is described in the following paragraphs.

> ***Note:*** Only information unique to the icon bar definition files is included here. The common fields are described "File syntax" (page 35).

Menu items: The following lines are used to define a menu item:

> ***Note:*** [] indicates optionality and []* means 0 or more times.

```
labelString: <menu item name>
[labelType: pixmap]
[labelPixmap: <pixmap path>]
tMCommandLine: <command line to invoke>,
or tMAction: <NV action>
[tMPresenceTest: <presence test>]
[helpText: <help information>]
[<MOTIF resource line for PushButtons>]*
```

where:

`labelType: pixmap` is a constant that is present if the item is displayed as an icon. In this case, a label pixmap icon always needs to be present.

`<pixmap path>` is the absolute path name for the file that contains the pixmap image of the icon displayed in the icon bar. If label type and label pixmap are not provided, the item's name is used for its push button.

`<NV action>` is an internal action to be performed. The following actions are supported:

`expand`	expands the selected nodes in place
`full_expand`	expands the selected nodes to module level in place
`expand_excl`	expands the selected nodes exclusively

`full_expand_excl`	expands the selected nodes to module level exclusively
`compress`	compresses the selected nodes
`restore_view <view name>`	restores the named view. The view name can contain the $NAME string that replaces the name of the selected node to automatically open a view corresponding to it.
`open_finder`	opens the finder dialog
`select_org`	shows the organization's roots
`open_shelf_dlog`	opens the shelf dialog for the selected node
open_subcomp_dlog <allltrbld>	opens the subcomponent or component dialog in either all or filtered mode as specified for the selected node

Creating a customized icon bar definition file

1. If you are customizing a push-button in an icon bar definition file and you need a new bitmap image for the push-button, go to step 2. If you are not doing this, skip to step 5.
2. Create your own bitmap image or choose a bitmap image in file /opt/MagellanNMS/lib/nvs/icons and modify it with a bitmap editor, such as bitmap. The bitmap image must be in X11 bitmap format.
3. Save the customized bitmap image.

   If your change is for a single Preside Multiservice Data Manager (MDM) user, save the image in $HOME, the user's home directory.

   If the change is for all MDM users, save the image in /opt/MagellanNMS/cfg/nvs/icons.
4. Change file permissions to allow read access by the group and others, and read-write by the owner.

   `chmod 644 <new file>`

5 Choose one of the files listed in "Where to find icon bar definition files for the Network Viewer icon bars" (page 55) as the starting point for your new customized file.

6 Copy the file into one of the following directories:

- an MDM user's home directory, if you are only going to make the new file available to one MDM user:

cp <file> /<users home directory path>/ /MagellanNMS/<new file>

- /opt/MagellanNMS/cfg/tsets/<language>, if you are going to make the new file available to all users of the workstation:

cp <file> /opt/MagellanNMS/cfg/tsets/<language>/<new file>

7 Access the directory that contains the newly copied file:

```
cd /<users home directory path>/MagellanNMS
```

```
cd /opt/MagellanNMS/cfg/tsets/<language>
```

8 Change file permissions to allow read access by the group and others, and read-write by the owner.

```
chmod 644 <new file>
```

9 Using a UNIX editor, modify the contents of the new icon bar definition file to suit your needs.

If you are customizing an icon bar definition file to add a new icon to a push-button in the Network Viewer, be sure to specify the new bitmap image file in the appropriate labelPixmap record.

10 Save the file and exit from the file.

11 Have users of the tool exit from the tool then restart it to obtain access to the new customization.

Customization examples

See the following sections for examples of customizing menus and setting resources in startup options:

- "Example 1, Setting the startup options in a menu definition file" (page 66)
- "Example 2, Creating a submenu item that invokes a macro" (page 66)

Example 1, Setting the startup options in a menu definition file

This section contains two examples that show how to create records that set resources in startup options for a Preside Multiservice Data Manager (MDM) tool in a menu definition file. The tool used in the examples is the Network Viewer.

You can also set resources in other places, including the user account's .Xdefaults set-up file and the Network Viewer default resource definition file /opt/MagellanNMS/cfg/app-defaults/C/ND. See "Customizing resources used by MDM tools" (page 15) for information about setting resources by other means.

Starting the Network Viewer with labels and icons turned on

When starting the Network Viewer you can specify whether labels and type icons are displayed by default. The following example shows the record needed to start the tool with the labels shown by default and all icons turned off.

```
tMCommandLine:    /opt/MagellanNMS/bin/nd \
                  -xrm 'ND*showAllabels: True' \
                  -xrm 'ND*showAllIcons: False'"
```

Setting the Network Viewer window position and dimensions as a startup option

The following example shows the record needed to start the Network Viewer, position the main window of the tool at coordinates 50 and 60, and set the window size to a width of 600 and a height of 500:

```
tMCommandLine:     /opt/MagellanNMS/bin/nd \
                   -geometry +50+60 \
                   -xrm "ND*mainWindow.width: 600" \
                   -xrm "ND*mainWindow.height: 500"
```

Example 2, Creating a submenu item that invokes a macro

The following example shows the records needed to create a submenu item called Current Path that invokes a macro. The macro invoked in this example, inactive, uses the current connection to display the inactive Processing Elements (PEs) of a DPN module. The record that invokes the macro also opens an X-Terminal window that displays the response to the macro. It also display the words *press return* in the X-Terminal window to remind you to

press the return key when you have finished reading the output from the macro. For information about writing macros, see “Creating and using macros” (page 69).

```
labelString:          List Inactive Components
tMCommandLine:        /opt/MagellanNMS/bin/nmxterm \
                      -c “inactive r72”; \
                      echo “press return” ; read <a>
```

where

<a> is user defined text

Chapter 3
Creating and using macros

This section contains information about creating and using macros in the Preside Multiservice Data Manager (MDM) software environment.

The information in this section is directed at programmers who have a working knowledge of programming, of UNIX utilities, who have experience creating scripts for UNIX-based shells, and who wish to create macros for MDM.

This section contains the following sections:

- "About macros for MDM" (page 69)
- "Macro structure" (page 74)
- "Where macros are stored" (page 75)
- "Using the cmccmd program" (page 75)
- "Writing macros for the Command Console" (page 83)
- "Setting the Command Console mode" (page 81)
- "Writing stand-alone command macros" (page 92)
- "Making use of the utilities and macros provided with the MDM software" (page 95)

About macros for MDM

A macro is a file that contains a program which runs complex or repetitive commands on network elements, parses the results if need be, and reports back to standard output (the screen by default).

Two examples of macros are the allcalls and ppDeltaDiag macros supplied with the Preside Multiservice Data Manager (MDM) software. The allcalls macro displays a list of the calls that are currently active on a port of a DPN switch, and the ppDeltaDiag macro queries counter statistics information on Passports and automatically computes the differences in values returned by the queries.

Macros can be written in a number of languages including: C, C++, Perl, and TCL/Tk. However, they are most commonly written in one of the UNIX shell languages. These include: Bourne Shell, C-Shell, Korn Shell, and desktop Korn shell (dtksh).

Command macros can be run within a user's MDM session or from outside a user's MDM session, as described in the following sections.

Within a user's MDM session

A Preside Multiservice Data Manager (MDM) session starts when a user logs in to an account that is set up to run the default MDM environment. When the user's MDM session starts, a number of servers are started, including session servers Command Functional Process (CMCFUN) and Connection Manager (CM). See the figure "MDM servers and programs associated with command macros" (page 73). The CMCFUN session server owns and manages the connections to the network for the user's MDM session.

> ***Note:*** An MDM session is identified by the value of its DISPLAY environment variable.

To establish a connection to the network, the CMCFUN server directs the CM server to make use of the network access servers to set up the connection as follows:

- It uses the HGDS and NCSMGR servers to create a CCIF process to handle messages for a connection between the CMCFUN process and an OA.
- It uses the HGDS and FDTM servers to create an FDTR process to handle messages for a connection between the CMCFUN process and a Passport switches in a Passport group.

Once the connection is established, the CMCFUN server routes operator commands to their intended destination: to an OA (through the CCIF process) or to a Passport switch in a Passport group (through the FDTR process).

The main tool used to access destinations in network is the Command Console. This tool allows you to connect to one or more destinations (OAs or Passport groups). Once the destinations are connected, an operator can use the Command Console to send one or more operator commands concurrently to network elements that can be accessed through the connected destinations.

The Command Console is a user interface to the CMCFUN server. A macro can also be used to send commands to the CMCFUN server and have the CMCFUN server route them to their intended destinations. The destination used by a macro can be one of the destinations connected by an operator the Command Console, or by the macro itself.

To issue a command to an OA or to a Passport switch, a macro must use the cmccmd program. Commands available through the cmccmd program can be inserted into a macro to connect to a destination, issue operator commands to a network element that is accessible through that destination, and disconnect from a destination.

Within a user's MDM session, command macros can be run from

- the Command Console
- the Component Information Viewer
- a UNIX shell
- a customized tools menu entry
- from another macro

Note: You could run a macro by a cron job from within a user's MDM session, but this is not recommended. If the user logs out before the cron job runs, the CMCFUN and CM servers cease to exist. Without these servers, the cron job has no means of routing commands that are contained in the macro to their intended destination.

Outside a user's MDM session

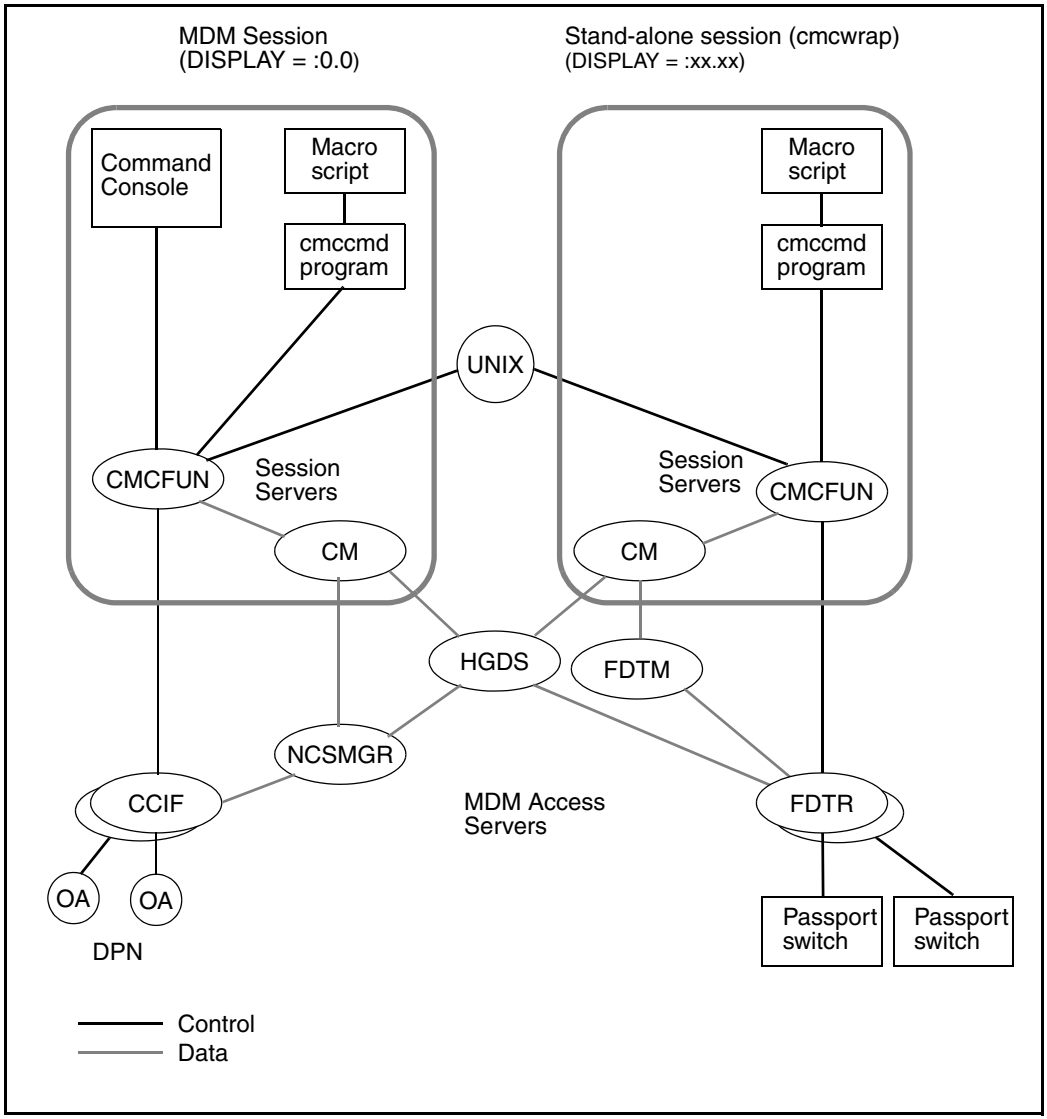
A macro can also be run from outside a user's Preside Multiservice Data Manager (MDM) session. Running a macro from outside a user's MDM session is needed for the following reasons:

- the macro needs exclusive access to a connected destination. For example, to provision Passport switches, a macro requires an NCS Capability ID and password with privileges that you may not wish to make available to an operator who is using the Command Console.
- the macro is to be run from a cron job set up to run automatically according to a command contained in a crontab file in directory /var/spool/cron/crontabs.

To establish a connection to a destination in the network, a macro requires a session and the CMCFUN and CM session servers that go with it. These servers, shown in the figure "MDM servers and programs associated with command macros" (page 73), can be provided to the macro by an MDM command wrapper program called cmcwrap.

Command macros that are constructed to run outside a user's MDM session are stand-alone macros. For information about the structure of a stand-alone command macro, see "Macro structure" (page 74). For information about writing stand-alone macros that call the MDM command wrapper program, see "Writing stand-alone command macros" (page 92).

Figure 5
MDM servers and programs associated with command macros



Macro structure

Macros that run from within a user's Preside Multiservice Data Manager (MDM) session are constructed to perform the following tasks:

1 Parse the command line and analyze the environment.

2 Make sure there is a route for the commands.

3 Issue command(s) to the Network Elements.

4 Parse, analyze, and filter the replies.

5 Display the results on the standard output stream.

6 Repeat these tasks as needed.

This structure takes for granted that the commands are to be issued to the Command Console's current destination or to a known destination.

Command macros that are intended to be run outside a user's MDM session (as stand-alone macros) are constructed to perform the following tasks:

1 Start the session servers required.

2 Parse the command line and the analyze the environment.

3 Connect to the appropriate Passport Group(s) and/or NCS OA(s).

4 Issue command(s) to the Network Elements.

5 Parse, analyze, and filter the replies.

6 Display the results on the standard output stream.

7 Repeat these tasks as needed.

8 Stop the session servers.

This structure is described further in "Writing stand-alone command macros" (page 92).

Where macros are stored

When a command macro is run, the operating system automatically searches for the command macro in the following directories:

1 $HOME

This is the user's home directory. This directory contains macros written by a customer that are only available to a single user's UNIX account.

2 /opt/MagellanNMS/cfg/macros/user

This directory contains macros written by the customer that are available to all Preside Multiservice Data Manager (MDM) users.

3 /opt/MagellanNMS/lib/macros/nms

This directory contains macros that are provided with the MDM software.

On a UNIX user account that is set up to run the default MDM user environment, you can run a command macro stored in any of these directories without specifying its absolute path name. If you wish to run a command macro that is stored in another directory, you must run it by means of its absolute path name, or modify the search path in the user account's set-up files to locate the macro automatically.

Using the cmccmd program

The cmccmd program lets you to establish connections to a destination (an OA or a Passport group), to issue operator commands to network elements that are part of the destination, and to drop connections to a destination.

To function properly, the cmccmd program must be run with the same DISPLAY environment variable as the CMCFUN and CM servers for a given Preside Multiservice Data Manager (MDM) session. Invoking the macro from the Command Console guarantees that this is the case.

Establishing new a connection with cmccmd

To establish a connection to a destination, run the cmccmd program as follows:

```
/opt/MagellanNMS/bin/cmccmd connect [OA|GROUP] \
<route name> <user ID> <password>
```

Note: connect can be abbreviated as conn.

where:

`OA|GROUP`

is an optional parameter that indicates the type of destination to which the <route name> applies: an OA or a Passport Group.

`<route name>`

is the name of an OA or a Passport group defined in Preside Multiservice Data Manager (MDM) software.

If the intended destination is an OA, <route name> must correspond to a valid WS-MDI name defined in file /opt/MagellanNMS/cfg/HGDS.cfg.

If the intended destination name is a Passport group, <route name> must correspond to a valid Passport group name defined in file /opt/MagellanNMS/cfg/HGDS.cfg.

`<user ID>`

If the intended destination is an OA, <user ID> must be a valid NCS capability ID on the OA that has the WS-MDI.

If the intended destination is a Passport group, <user ID> must be a valid userid defined on the Passport switches in the group.

`<password>`

If the intended destination is an OA, <password> must be the valid NCS password associated with the NCS capability ID.

If the intended destination is a Passport group, <password> must be the valid password for the userid.

If the connection is successful, the cmccmd program outputs a message, as follows:

- For a connection to an OA:

```
CC_STATUS  1041 The following NCS destination has been
established: <route name>

<capability set information>
```

- For a connection to a Passport group:

```
CC_STATUS  1042 The following Passport destination has
been established: <route name>

<scope and impact information>
```

In addition, the cmccmd program returns an exit code of 0 (available as the $? variable in Bourne shell and the $status variable in C shell). See also information about the -v option in "Identifying individual command responses" (page 80).

If the connection already exists, the following error message appears and the cmccmd program returns an exit code other than 0:

```
Connection to <route name> already exists. Request
discarded.
```

If the connection cannot be established, a number of error messages can be displayed to indicate the source of the error, but the exit code associated with them is always a value other than 0.

Dropping an existing connection with cmccmd

To drop an existing connection, run the cmccmd program as follows:

/opt/MagellanNMS/bin/cmccmd disconnect <route name>

Note: disconnect can be abbreviated as disc.

If the disconnect is successful, the cmccmd program outputs a response similar to the one that follows, and outputs a return code of 0:

```
CC_STATUS  1040 The following connection has been
terminated: <route name>
```

If the disconnect fails, the cmccmd program outputs an error message and outputs an exit code other than 0.

Issuing operator commands with cmccmd

To issue a single operator command, run the cmccmd program as follows:

/opt/MagellanNMS/bin/cmccmd command <route name> <operator command...>

Note: command can be abbreviated as cmd.

where:

<route name>

is the name of a destination to handle the command. This destination must already be connected.

You can use the Command Console specialized routes that include the UNIX macro route ($), the SNMP command route (@), and the Passport wild-card route (*). When using these specialized routes, enclose the route character in quotes (either single or double), or use the Escape character before the route character, for example, '$', "$", or \$. Using quotes or the Escape character ensures that UNIX command shells do not interpret the route characters as special characters. You do not need to connect before using the specialized routes.

To use the Command Console's current default route, <route_name> can be specified as the value of the CMC_CURRENT_ROUTE environment variable. When specifying this variable, precede it with a dollar ($) and enclose it within quotes as follows: "$CMC_CURRENT_ROUTE"

If you specify the current route, be aware that the current route can be an unexpected route, such as a Passport route for NCS commands.

<module name>

is the name of the DPN-100 module or Passport module within the OA group or the Passport group for which the command is intended

```
<operator command...>
```

is the command to execute on the module that is specified with the <module name> parameter. The operator command name typically begins with the name of the destination module.

Guidelines for issuing operator commands are as follows:

- You can specify a command as a single string argument or as a set of individual arguments.
- If multiple commands are to be issued to the same destination, you can either run the cmccmd program multiple times or you can input the commands on separate lines as shown in the following example. If you are entering the commands interactively, you can indicate the end of file shown in the following example, by pressing Control and d together.

```
/opt/MagellanNMS/bin/cmccmd command <module name> \
<route name>
<module name> <command...>
<module name> <command...>
...
<end-of-file>
```

In this example, the cmccmd program reads the commands one per line on its standard input and issues them through the named route, one at a time, until it reads the end-of-file. This allows a command macro to pipe, or redirect from a file, a list of commands to execute.

It is also possible to omit the <route name> argument from the first line. If you do this, the cmccmd program expects <route name> to be the first token of each subsequent line on its standard input until the end-of-file, as follows:

```
/opt/MagellanNMS/bin/

cmccmd command
<route name> <module name> <command...>
<route name> <module name> <command...>
...
<end-of-file>
```

In both these cases, it is assumed that the *-v* option is being used to append an ENDCOMMAND to each command being executed. See "Identifying that a connection is available" (page 81) for a description of the *-v* option.

Listing the available routes with cmccmd

You can get the list of available routes, similar to the destination list in the Command Console Connection Manager dialog, by issuing the following command:

```
/opt/MagellanNMS/bin/cmccmd list [OA|GROUP|ALL]
```

where:

`OA, GROUP`, and `ALL`

indicate the types of routes to be displayed

The routes are listed in the following format:

```
<name> OA|GROUP CONN|AUTH|-
```

where:

`OA` and `GROUP`

indicate the type of the route

`CONN, AUTH` and -

indicate whether this route already has a connection, authentication, or is not being used

Identifying individual command responses

To help you determine that a command is issued properly, and to determine that the response to a command is complete in the cases described previously, you can use the -v option as follows:

```
/opt/MagellanNMS/bin/cmccmd -v connect ...
/opt/MagellanNMS/bin/cmccmd -v disconnect ...
/opt/MagellanNMS/bin/cmccmd -v command ...
```

When the -v option is used or implied, the following line is appended to the response to each command that is executed:

```
ENDCOMMAND <exit code>
```

where:

`<exit code>`

is the command's exit code. When multiple commands are being issued to the cmccmd program by means of a single call that includes the -v option, each command is followed by its own ENDCOMMAND.

Note: Exit codes provided by the cmccmd program only indicate whether an operator command is dispatched. They do not indicate whether an operator command has been successfully executed by the network element itself.

Identifying that a connection is available

To help you determine whether a connection is available, you can issue an empty command (" ") to the connection, as follows:

```
/opt/MagellanNMS/bin/cmccmd command <route name> ""
```

If the route is currently connected, the command returns a blank line and an exit code of 0. However, if the connection is not available, the command produces the following response and returns an exit code other than 0:

```
CM_ERROR   1014 Unknown destination route for command.
```

Note: Other errors messages are possible.

Setting the Command Console mode

By default the Command Console opens in Command Console mode. You can override that default and have the Command Console open in Connection Console mode or Authentication Console mode. To set your own defaults, use the following command:

```
/opt/MagellanNMS/bin/cmcui \
[-CC|-AC] \
[-desttype OA|GROUP] \
[-dest <default destination name>] \
```

```
[-ccmsg <prompt string>] \
[-printauth] \
[-comp<component ID>]
```

where:

`-CC|-AC` indicates the mode. -CC specifies Connection Console mode; -AC specifies Authentication Console mode. If you do not specify a mode, the standard Command Console opens. If you select Authentication Console mode, you must also specify the `-desttype` option.

`-desttype OA|GROUP` specifies the type of destination that displays. You can display OAs, GROUPs, or both. If you do not specify the destination type, both OAs and GROUPs display. If you select Authentication Console mode, you must specify the `desttype`.

`-dest <default destination name>` specifies a default selection for the dialog if the destination exists.

`-ccmsg <prompt string>` specifies a string to add to the dialog message area to prompt the user for input.

`-printauth` adds the user ID and password to the `connected`: and `authenticated:` standard output lines.

`-comp <component ID>` is an option to the plain Command Console only. This option enters a default component in the Command Console command area.

In the context of the Component Information Viewer diagnostics, the Connection Console should not be directly invoked. Instead, use the `execWithDest` utility macro. This macro identifies the default destination, invokes the Command Console when needed, and executes a specified command in the context of the default connection. For details, see “execWithDest” (page 133).

Actions that you perform on any tools or scripts that you run are sent to standard output. The Connection Console actions consist of connecting, disconnecting, and selecting. The output corresponding to these actions are as follows:

- `connect: <OA or GROUP> <destination name> [<userid> <password>]`
- `disconnect: <OA or GROUP> <destination name>`
- `selected: <OA or GROUP> <destination name>`

The output `selected: <OA or GROUP> <destination name>` indicates the last selected destination. If this output does not display, no destination is selected and the Connection Console dialog closes.

Writing macros for the Command Console

This section explains how to run a macro from the Command Area of the Command Console, describes the running environment for such macros, and contains examples of macros that are designed to be run from the Command Console from within a user session.

Invoking a macro from the Command Console

A macro can be run by entering a dollar sign ($) followed by the name of the macro in the Command Area of the Command Console. The dollar sign is the <route name> for commands that are to be directed to UNIX. For a description of the Command Console and the instructions to use it, see 241-6001-804 *Preside MDM Workstation Utilities User Guide*.

For example, to run the whichmodels command macro that is supplied with the Preside Multiservice Data Manager (MDM) software, enter the following command in the Command Area. This macro executes a set of UNIX commands that lists all of the Network Models that are available on the workstation.

```
$ whichmodels
```

In response, a list of the Network Models is displayed in the response area of the Command Console window, similar to the list in the following example:

```
The available network models are (* = Fast Load):
10model
8lab
8model
DEMO
empty
```

Running environment for command macros run from the Command Console

When a macro is run from the Command Console, its running environment can be described as follows:

- The command line arguments are those entered after the macro name in the command area of Command Console main window. That is

  ```
  $ my_macro p1 p2 p3 p4...
  ```

  The arguments p1, p2, and p3 are available as $argv[1], $argv[2]... in C and C++, and as $1, $2... in UNIX shells.

- The standard output and error streams are copied to the response area of the Command Console main window. Therefore everything output by the script (that is printf in C and C++, and echo in the UNIX shells) is displayed to the operator. Because there is no standard input stream, it is not possible for a macro to interact with the operator, unless a separate window such as a UNIX Access window is opened for that purpose. Interact refers to such actions as prompting the operator for a parameter.

- The CMC_CURRENT_ROUTE environment variable holds the name of the current route with a value as follows:

  ```
  <DISPLAY environment variable value> <current route>
  ```

  For example: :0.0 OA_WEST

This variable is available as "$CMC_CURRENT_ROUTE" in UNIX shells and through the command getenv(“CMC_CURRENT_ROUTE”) in the C and C++ programming languages. Because the value of the variable contains a blank, you must place it within quotes so that UNIX shells do not treat each item in the value as if it were a separate token.

- The DISPLAY environment variable holds the name of the current X-Windows display, which also happens to be the name of the current Preside Multiservice Data Manager (MDM) session. This is important because this is how the cmccmd program locates the correct CMCFUN server.
- The rest of the environment is the same as that of the UNIX user who owns the MDM session, including other environment variables such as NMSXTERM. For information about global environment variables, see 241-6001-303 *Preside MDM Administrator Guide*.

A macro can execute any command that an operator’s access privileges allow the operator to execute. The following points should be taken in consideration though:

- A macro can be run multiple times simultaneously so be careful to manipulate common files correctly. If you are not careful to do so, corrupted data can ensue should several macros write to the same file at the same time. Temporary files should be given unique names. For example, using $$ in UNIX shells or mktemp in C or C++. These files should reside typically in the /tmp directory and should be removed when they are no longer needed.
- Because command macros use the same connections as those established through the Command Console, avoid disconnecting a destination, unless the macro established the connection to that destination in the first place.

  For an example of a macro that establishes a connection to a destination when the connection is not already established by the Command Console, runs commands, and on completion only disconnects from the destination to which it established a connection, see “Example 4, Making an explicit connection” (page 88). If the macro needs a dedicated connection, write the macro as a stand-alone macro. For the instructions to write a stand-alone macro, see “Writing stand-alone command macros” (page 92).

Examples of macros to be run from the Command Console

The following examples demonstrate the use of cmccmd program commands in macros that are designed to be run from the Command Console.

Example 1, Using the current connection

The following macro, called /opt/MagellanNMS/cfg/macros/user/inactive, uses the current connection to display the inactive Processing Elements (PEs) of a DPN module:

```
#/bin/sh
# This macro should be run from the Command Console
# with the appropriate OA as the default route.
# $1 is a module name

/opt/MagellanNMS/bin/cmccmd cmd "$CMC_CURRENT_ROUTE"\
$1 d | grep inactive
```

This macro can be run from the Command Console while the current route is an appropriate WS-MDI, as follows:

```
$inactive r72
pe 13 inactive
pe 30 inactive
```

This example shows how one can pass arguments to a command macro and how standard UNIX utilities (grep) and piping mechanisms (|) can be used to enhance the output that results from an operator command to a network element.

For a more specific example of this, use a UNIX editor to look at macro /opt/MagellanNMS/cfg/macros/nms/dpnup and its associated awk script /opt/MagellanNMS/cfg/macros/nms/dpnupa.cfg. The awk script parses the output of the dir command to display only the destinations that are available.

Example 2, Using a specific connection

The macro in “Example 1, Using the current connection” (page 86) can be modified to allow you to optionally specify the OA MDI to use, as follows:

```
#/bin/sh
# This macro should be run from the Command Console
# with either the OA as first argument or as the
```

```
# default route.
# With a single argument, $1 is a module name and use
# the default route

if [ $# == 0 -o $# > 2 ]
     then
     echo "Usage: $0 [<route>] <module>"
     exit 1
elif [ $# = 2 ]
then
     route=$1
     mod=$2
else
     route="$CMC_CURRENT_ROUTE"
     mod=$1
fi

/opt/MagellanNMS/bin/cmccmd cmd "$route" $mod d \
| grep inactive
```

This macro in this example also includes some basic error checking. The error checking ensures that the correct number of arguments are provided and that the errors produce error messages.

Example 3, Detecting the availability of a route

We can further enhance the macro in "Example 2, Using a specific connection" (page 86), to make sure that the desired route is connected and to issue proper error messages if it is not, as follows:

```
#/bin/sh
# This macro should be run from the Command Console
# with either the OA as first argument or as default
# route.
# With a single argument, $1 is a module name and use
# the default route

if [ $# == 0 -o $# > 2 ]
then
     echo "Usage: $0 [<route>] <module>"
     exit 1
elif [ $# = 2 ]
then
     route=$1
     mod=$2
```

```
else
     route="$CMC_CURRENT_ROUTE"
     mod=$1
fi
if /opt/MagellanNMS/bin/cmccmd "$route:\
"" > /dev/null 2>&1
then
     /opt/MagellanNMS/bin/cmccmd cmd "$route" $mod d \
     grep inactive
else
     echo "Please connect to $route
     echo "before invoking this macro."
     exit 1
fi
```

The output and error streams of the first cmccmd invocation are redirected to /dev/null so that they are not displayed to the operator (that is, in the response area of the Command Console main window).

An alternative way to verify the route's availability is to use the *list* command of the cmccmd program, as follows:

```
if /opt/MagellanNMS/bin/cmccmd list ALL \
| grep "^$route\>.*CONN" > /dev/null 2>&1
then
# we are connected, issue command
...
```

Example 4, Making an explicit connection

The following is an example of a macro that can be run from within the Command Console or outside the Command Console. The macro checks that the connection needed is available and if the connection is not available, the macro establishes the connection, then drops the connection when it is no longer needed.

```
#!/bin/sh
# First check if the network connection is available

if /opt/MagellanNMS/bin/cmccmd command myroute ""
then
     # it is therefore we do not own it
     own_it=0
else
     # We must try to establish it ourselves
```

```
        own_it=1
        # Passwords should not be coded in scripts as
        # follows
        /opt/MagellanNMS/bin/cmccmd connect GROUP \
        myroute myid mypwd > /dev/null
        if [ $? != 0 ]
        then
             echo "Could not connect to myroute.
             exit
        fi
fi

# issue the necessary commands
/opt/MagellanNMS/bin/cmccmd command myroute mypp \
d -c -o lp/\* osi,activeCard

/opt/MagellanNMS/bin/cmccmd command myroute mypp \
d -c -o sh card/\* osi,currentLp

if [ $own_it = 1 ]
then
        /opt/MagellanNMS/bin/cmccmd disc myroute

fi
```

Do not encode network authentication in scripts like the one in this example, unless the authentication has a very low scope and impact (passive).

A backslash character (\) is used to escape the wildcard symbol (*) and to prevent the command from globbing into all the matching file names as a result of the way in which UNIX shells interpret the command before passing it to the cmccmd program. The same should be done for occurrences of the dollar sign ($) and question mark (?). For the wildcard symbol (*) and question mark (?) we could also have turned file globbing off at the beginning of the script (set noglob in csh, set -f in sh and ksh).

Another useful macro is one that automatically establishes network connections when the user logs in, thereby avoiding the need to establish the connections manually. Here is an example of this macro:

```
#!/bin/sh
/opt/MagellanNMS/bin/cmccmd connect OA myoa1 \
mycap mypwd
```

```
/opt/MagellanNMS/bin/cmccmd connect OA myoa2 \
mycap mypwd

/opt/MagellanNMS/bin/cmccmd connect GROUP \
mygroup1 myuid mypwd

/opt/MagellanNMS/bin/cmccmd connect GROUP \
mygroup2 myuid mypwd
```

This macro can be run in the user's .xsession or .dtprofile setup file just after the statement that starts the Preside Multiservice Data Manager (MDM) session. Like the previous example, this macro has the disadvantage of coding authentication information in a file. However, the security risk caused by including this information can be minimized by setting file permissions for the macro file (chmod 700 <macro file>) so that only the owner of the macro can read and execute its contents.

Example 5, Issuing multiple commands

The macros in this section are equivalent examples that show how to issue multiple commands with a single invocation of the cmccmd program. The macros issue a d (display) command for a list of modules:

```
#!/bin/sh
# this version uses input redirection from the script

/opt/MagellanNMS/bin/cmccmd cmd "$CMC_CURRENT_ROUTE"\
<<ENDOFIT
r70 d
r71 d
r72 d
r73 d
r74 d
ENDOFIT
```

Another alternative is as follows:

```
#!/bin/sh
# this version used a shell routine to compose the
# command from a hard-coded list of modules and a UNIX
# pipe to send these commands to the network

pipecmd()
{
for i in r70 r71 r72 r73 r74
     do
```

```
          echo "$i d"
     done
}

pipecmd \
| /opt/MagellanNMS/bin/cmccmd cmd "$CMC_CURRENT_ROUTE"
```

Another alternative is as follows:

```
#!/bin/sh
# this version is similar to the preceding one but
#this time the list of modules is taken from the file
# named as argument to the shell routine

pipecmd()
{
     while read name rest
      do
           echo "$name d"
      done < $1
}

pipecmd modulelist \
| /opt/MagellanNMS/bin/cmccmd cmd "$CMC_CURRENT_ROUTE"
```

Example 6, Invoking cmccmd from a non-script language

Most UNIX shells allow you construct macros to perform all the operations you need but there are circumstances in which it is necessary use a classic programming language like C or C++. If you use one of these programming languages, use the UNIX *popen* mechanism to interact with the cmccmd program, as follows:

```
#include <stdio.h>
char line[256];
FILE *fp;
/* open an output pipe to the cmccmd command */
fp = popen("/opt/MagellanNMS/bin/cmccmd -v command
          myroute mycommand",
          "r");

/* read each output line from cmccmd */
while ( fgets(line, 255, fp) > !=null)
{
/* check if we have reached the end of the response */
if ( strncmp(line, "ENDCOMMAND", 10) == 0 )
         break;
```

```
/* handle the output line */
     ...
}
```

Invoking macros (that are not stand-alone macros) from outside the Command Console

Because the session servers are available to any tool that is started from within a Preside Multiservice Data Manager (MDM) session (that is any tool that shares the same DISPLAY environment variable value), it is possible to run macros from outside of the Command Console but still within the same MDM user session. For example, from a customized tools menu entry or manually from a UNIX Access window. These macros share any existing connections with the Command Console. An examples of this type of macro is the Operator Command tool. The Operator Command tool is a powerful interactive macro that is started from tools menu entries (in NV, CSD, AD, and CIV) and that runs within the current MDM user session.

Writing stand-alone command macros

Writing stand-alone macros is not much more complex than writing macros to be run within a Preside Multiservice Data Manager (MDM) session. The only difficulty to overcome is that the cmccmd program needs the CMCFUN and CM session servers to function and these servers are not started unless a user logs in to the workstation with a user account that is set up to run the default MDM environment.

One reason to create a stand-alone macro is to run the macro as a cron job. Because a cron job usually runs outside of any user environment, steps must be taken to start a dedicated set of session servers for the duration of the cron job. You may use the MDM Command Wrapper program (cmcwrap) for this purpose. Because stand-alone macros operate with new independent session servers, the only available connections are those that the macros establish themselves. Stand-alone macros are therefore just like those that are intended to run within a user's MDM session, except that they must establish their own connections. For this reason, it is necessary to provide authentication information inside the macro.

There is an exception to this: if the macro is not meant to be run as a cron job but is to be run interactively, then the macro can prompt the operator for the authentication information and can be run along with independent session servers.

The command wrapper program (cmcwrap)

The cmcwrap program runs a given macro in an environment that corresponds to a user session; that is, a session in which the standard session-servers are running. These session servers are terminated when the program exits. Its usage is:

```
/opt/MagellanNMS/bin/cmcwrap \
[-i] [-gp] "<macro command line>"
```

where:

`-i`

is an optional parameter that must be provided if your macro is to prompt the user through the standard input stream

`-gp`

is an optional parameter to prevent the Generic Prober session server from starting. This session server is used by the Data Viewer and is typically not needed to run macros.

`"<macro command line>"`

is the command line containing the macro name that is to be executed

When the cmcwrap program runs, it first makes sure that the DISPLAY environment variable is set to a unique value. Remember that the session servers are uniquely identified with the value of the DISPLAY environment variable. Redefining the DISPLAY variable with a unique value ensures that a new set of independent session servers can be started and used by the programs started by the macro.

Note: The DISPLAY variable does not correspond to any valid X-Windows display and therefore no X-Windows based tools should be started by the macro.

The next step the cmcwrap program performs is to start a new set of session servers (CM, CMCFUN, and GP). These servers are started and maintained by a Preside Multiservice Data Manager (MDM) utility called loop (/opt/MagellanNMS/bin/loop). The loop utility runs its command line argument as a program and restarts it if it terminates, to make sure that the program is always available. The loop utility terminates by itself and the program it maintains, once its parent process exits, or if it is terminated by means of a HUP signal. The cmcwrap program then runs the macro provided on its command line. Once the macro terminates, cmcwrap automatically terminates the session servers it started.

For example, to run the DUMPALL macro, which triggers a DPN-100 Data Spooling Dump for all data types as a cron job, once every hour at the following entry to your crontab file:

```
0 * * * * /opt/MagellanNMS/bin/cmcwrap -gp \ /opt/
MagellanNMS/cfg/macros/nms/DUMPALL
```

The first 5 parameters in the entry specify the time at which the command (beginning with /opt) is to run. These 5 parameters are as follows:

- minutes, with a value of 0 to 59
- hours, with a value of 0 to 23
- day of the month, with a value of 1 to 31
- month, with a value of 1 to 12
- day of the week, with a value of 0 to 6, where 0 is Sunday

In these parameters you can use commas (,) and hyphens (’) to specify a range of numbers or you can use an asterisk (*) to specify all possible values, as has been done in the example.

Starting a session and ending a session can take a long time. Therefore call the cmcwarp program near the beginning of the program and call it as few times as possible in the macro to minimize the time consumed by setting up and ending a session.

Making use of the utilities and macros provided with the MDM software

This section contains information about a number of utilities and macros provided with the Preside Multiservice Data Manager (MDM) software that can prove useful when writing macros or API-based scripts.

MDM utilities grouped by function

This section contains descriptions of utilities that are provided with the Preside Multiservice Data Manager (MDM) software. Descriptions of the utilities are arranged according to the function that they provide. Each description includes one or more examples of code in Bourne shell syntax to show how the utility can be run.

Invoking a macro in a separate UNIX Access window

To run a macro in a new UNIX Access window, use the following code segment. This code segment runs the macro from the user's preferred UNIX Access window (according to the NMSXTERM environment variable).

```
/opt/MagellanNMS/bin/nmsxterm [<X options...>] \
-e <macro invocation>
```

Invoking a macro in a new UNIX Access window is often used when you want to run a command macro from within a tool or a menu.

The following example runs the Operator Command tool is taken from the NV Node Tools Menu:

```
/opt/MagellanNMS/bin/nmsxterm -geometry 80x24-20+220 \
-fg Navy -bg '#e5e5e5' \
-e /opt/MagellanNMS/cfg/macros/nms/cmdshell \
cmdshell.menu $COMP
```

Popping up a message window

Your script or macro may need to warn the user of some event. Preside Multiservice Data Manager (MDM) software provides a utility which pops up a message dialog. Its command line syntax is as follows:

```
/opt/MagellanNMS/bin/xmsg "<message>" \
[-title <title string>] \
[-buttons <buttons specification string>] \
[-pixmap <bitmap filename>] \
```

```
[-labelMargin <margin>]
[-textHeight <height in characters>] \
[-textWidth <width in characters>] \
[-delay <delay timer>] \
[-bell] \
[<X options...>] \
[-h]
```

where:

`<message>`

is the string or the minus character (*-*). The minus (-) character indicates that the message is to be read on standard input and displayed in a scrolled window. The dimensions of the scrolled window can be specified using the -textHeight and -textWidth options.

`<title>`

is the title of the message dialog.

`<buttons specification string>`

are button specifications separated by commas in the form <label>:<exitcode>,<label>:<exitcode>,... . A maximum of three buttons can be specified.

`<bitmap filename>`

is the full pathname to a bitmap file for the dialog.

`<delay timer>`

is the number of seconds before the dialog closes. When 0 is specified, the dialog is present until one of the buttons is selected.

`-bell`

rings a bell to alert the user.

`<X options...>`

are Xresources to be applied to the dialog. For example: colors.

-h

displays help information for the xmsg command.

The following example presents a warning dialog to the operator each time a DPN-100 service data activation alarm is received when a script invoking this code segment is used as the command in the AlarmTrigger utility. The use of this utility is described in "Triggering a macro on receipt of an alarm" (page 99).

```
/opt/MagellanNMS/bin/xmsg "Node $5 has just activated
new service data." \
-pixmap /usr/dt/include/bitmaps/xn_warning \
-delay 30 \
-bg orange
```

This program can also be used to ask users simple questions:

```
/opt/MagellanNMS/bin/xmsg "Are you sure you want to
delete MCF MC.KEEP01.4034.0?" \
-title Question \
-buttons Yes:0,No:1 \
-pixmap /usr/dt/include/bitmaps/xm_question \
-delay 0\
-bell \
```

If the user selects the *Yes* button, the program exits with status 0; if the user selects the *No* button, the program exits with a status of 1.

Converting component ID formats

It is sometimes necessary to convert a Preside Multiservice Data Manager (MDM) Component ID from its canonical form (as provided by the APIs and in the COMP_ID Context, and its display format (and vice versa). The following command does this transformation for you:

/opt/MagellanNMS/bin/surcompxlate [-d] <component ID>

where:

<component ID>

is a component ID either in display format or in API format. Both are recognized. If the -d option is specified, the component's display name is produced on output. If the -d option is not specified, it is the API format that is output.

The following example extracts the current component ID from context and presents it in display format:

```
comp=`/opt/MagellanNMS/bin/ctxcmd -g COMP_ID`
if [ ! -z "$comp" ]
then
     disp=`/opt/MagellanNMS/bin/surcompxlate $comp`
     echo "The component is ${disp}."
fi
```

Identifying or changing the current component ID context

The current value of the Preside Multiservice Data Manager (MDM) component context can be extracted as follows:

```
/opt/MagellanNMS/bin/ctxcmd -g COMP_ID \
| /bin/sed -ne 's/value is ://p'
```

To get the Hot Context component ID, use DPN_QUICK_STEP instead of COMP_ID.

You can also track changes of one of these two variables with the following command:

```
/opt/MagellanNMS/bin/ctxcmd \
-r <COMP_ID or DPN_QUICK_STEP> \
| /bin/sed -ne 's/value is ://p'
```

Changing the values for these variables in a current user session affects tools that are running as if a component were selected. For example, as if a component were selected in the Network Viewer.

The following example extracts the component ID, ensures that it identifies a Passport, and extracts its name for use in commands:

```
COMP=`/opt/MagellanNMS/bin/ctxcmd -g COMP_ID \
     /bin/sed -ne 's/value is ://p'`
NAME=`/bin/expr "EM/\([^ ]*\).*" : "$COMP"`
if [ ! -z "$NAME" ]
```

```
then
     # it's a Passport, send commands to $NAME
fi
```

Triggering a macro on receipt of an alarm

It is possible to have a command macro run automatically on receiving an alarm. The AlarmTrigger script allows you to specify various filters and a command to run each time a matching alarm is received:

```
/opt/MagellanNMS/cfg/macros/nms/AlarmTrigger \
    [-fault "<faultcode>"]... \
    [-pfault "<faultcode pattern (post filtered)>"]* \
    [-pref "<component prefix>"]... \
    [-comp "<component>"]... \
     [-pcomp "<component pattern (post filtered)>"]* \
    [-cfile "<component pattern file (post
                            filtered)>"]* \
    [-event set|clear|message]... \
    [-sev critical|major|minor|warning|cleared|
                    indeterminate]... \
    [-host <GMDR hostname>] \
    [-serv <GMDR service name>] \
    [-or (alternative filters)] \
    -cmd "<macro invocation>"
```

The following example launches a script every time a DPN-100 service data activation alarm is received:

```
/opt/MagellanNMS/cfg/macros/nms/AlarmTrigger \
     -fault "FFFF30FF" \
     -cmd mymacro
```

where:

`-fault "<faultcode>"` filters on the specified fault code (left match)

`-pfault "<faultcode pattern>"` post-filters on the (grep style) fault code

`-pref "<component prefix>"` filters on the component ID (left match)

`-comp "<component>"` filters on the component ID (full match)

`-pcomp "<component pattern>"` post-filters on the (grep style) component ID

`-cfile "<component pattern file>"` is like -pcomp except component patterns are (one per line) in the named file

`-event set|clear|message` filters on the alarm event

`-sev critical|major|minor|warning|cleared|indeterminate` filters on the alarm severity

`-host <GMDR hostname>` uses GMDR on the specified Preside Multiservice Data Manager (MDM) host

`-serv <GMDR service name>` uses an alternate GMDR server

`-or` splits between two filter groups. By default, filters on different attributes are ANDed and these ORed with filters on different attributes. Using -or indicates that the following filters are to be interpreted as a different group (sieve) ORed with the other ones allowing for complex filters [for example (fault = x or fault = y and event = z) or (fault = a or fault = b and event=c)].

`-cmd "<command script name>"` is the command to execute upon reception of an accepted alarm.

Ensuring that a program is always running

You may sometimes need to ensure that a program is always running, something like the SVMDMN server. Unlike a server, a program must be restarted every time it terminates. To ensure that a program is always running, the Preside Multiservice Data Manager (MDM) software includes the *loop* utility, which can be run as follows:

```
/opt/MagellanNMS/bin/loop [-forever] \
<program invocation>
```

The program is started and every time it terminates (other than with a *KILL* (*9*) signal or an exit code of 50) it is automatically restarted up to 50 times. However, if you specify the -forever option, the 50 times limit and the exit code of 50 do not apply. Should the loop program itself be terminated or sent a non-KILL signal, the loop utility automatically ensures that the program it

controls is terminated first by sending it a HUP (1) signal). Also, if the process that started the loop utility terminates without terminating the loop, the *loop* utility detects this event within 30 seconds and automatically terminates its program and itself.

The following example is an extract from the scripts that maintain the MDM session servers while a user is logged in. The process ID of the loop is remembered to ensure proper cleanup later:

```
/opt/MagellanNMS/bin/loop -forever \
/opt/MagellanNMS/bin/icm &
cmpid=$!
```

Identifying the current Service Selection for a user

To extract the current value of the Service Selection setting that a user has made with the Service Selection tool, use the output of the following command pipe:

```
/opt/MagellanNMS/bin/ctxcmd -s <var> \
| /bin/sed -ne 's/value is ://p'
```

where:

`<var>`

is one of the Preside Multiservice Data Manager (MDM) service areas:

SURSERVERHOST for surveillance
NMSERVERHOST for the Network Model
DPNACCSERVERHOST for DPN access
EMACCSERVERHOST for Passport access
ARCHSERVERHOST for DPN provisioning

For information about the Service Selection tool, and how a user can select an MDM Service Area with it, see 241-6001-303 *Preside MDM Administrator Guide*.

The following example sets *DEST* to the current Service Selection setting which happens to be for the Network Model, then uses the name of the host that provides access to the Network Model in a Network Model API shortcut query:

```
DEST=‘/opt/MagellanNMS/bin/ctxcmd -s <var> \
| /bin/sed -ne ‘s/value is ://p’‘
/opt/MagellanNMS/bin/nmapi -h "$DEST" -node PM R78 \
-attr_id all
```

Changing the user Service Selection

To change the value of the current Service Selection setting for a user, issue the following command:

```
/opt/MagellanNMS/bin/ctxcmd -p <var> <hostname>
```

where:

`<var>`

is one of:

SURSERVERHOST for surveillance
NMSERVERHOST for the Network Model
DPNACCSERVERHOST for DPN access
EMACCSERVERHOST for Passport access
ARCHSERVERHOST for DPN provisioning

`<hostname>`

is an appropriate MDM hostname

The following example realigns the DPN Access Service Selection within a cmcwrap script to force the session to use that selection:

```
/opt/MagellanNMS/bin/cmcwrap \
“/opt/MagellanNMS/bin/ctxcmd -p DPNACCSERVERHOST \
myserver ; \
mymacroscript and its args”
```

Sending a log to the MDM Log Collector and Display

To create a simple log message that is displayed in the System Log Display tool along with other logs generated by Preside Multiservice Data Manager (MDM) software, run the following utility:

```
/opt/MagellanNMS/bin/nmsgenlog <fault code> \
"<log comment>"
```

where:

`<fault code>`

is a string of 8 hexadecimal digits

The following example generates a log which indicates that a macro has just started:

```
/opt/MagellanNMS/bin/nmsgenlog 00000000 \
"MyMacro is now waiting for alarms."
```

Identifying the Passport nodes in a group

It is sometimes necessary to identify the Passport hosts part of a given group. This can be obtained with the following command:

```
/opt/MagellanNMS/bin/hgdsapi [-h <hostname>] \
-child <Passport group name> \
| /bin/sed -ne 's/_obj_id: FMemberId S //p'
```

where:

`<hostname>`

is the name of a Preside Multiservice Data Manager (MDM) host that supports Passport network access. For a code segment that can be used to identify the current service selection, see "Identifying the current Service Selection for a user" (page 101).

The same operation can be done for DPN switches only by parsing the output of the OA DIR command (DPNUP) macro.

The converse operation, obtaining the names of groups for a given Passport host is also possible, as follows:

```
  /opt/MagellanNMS/bin/hgdsapi [-h <hostname>] \
     -parent <Passport host> \
> | /bin/sed -ne 's/_obj_id: FGroupId S //p'
```

The following example contains code that loops over a Passport's parent groups and identifies the first one that is already connected (if any):

```
groups='/opt/MagellanNMS/bin/hgdsapi -parent mynode \
| /bin/sed -ne 's/_obj_id: FGroupId S //p'`
for i in $groups
do
    if /opt/MagellanNMS/bin/cmccmd cmd $i "" \
    /dev/null 2>&1
     then
          # we're connected to this one
          group=$i
          break
    fi
done
```

Other MDM utilities

Preside Multiservice Data Manager (MDM) provides a number of specialized utilities, tools, and interfaces that can prove useful in writing macros and scripts. These include the following items:

- cdbextract

  This utility extracts data from the Customer Database. See 241-6001-804 *Preside MDM Workstation Utilities User Guide* for information about this utility.

- API providers

  MDM offers a number of API interfaces to provide you with access to the management data collected and managed by its servers. See the following Nortel Network technical publications (NTPs) for information about these interfaces:

  — 241-6001-200 *Preside MDM Application Programming Interface Primer*

- 241-6001-201 *Preside MDM Network Model API Reference Guide*
- 241-6001-203 *Preside MDM Alarm and Status API Reference Guide*
- 241-6001-204 *Preside MDM DPN Provisioning API Reference Guide*
- 241-6001-207 *Preside MDM Passport Provisioning API Reference Guide*
- 241-6001-209 *Preside MDM Provisioning Command Filter API Reference Guide*

- nmslog

  This utility lists the most recent MDM logs stored in the log buffer then outputs other logs as they are generated.

- NRS

  The Network Reporting System provides the base elements to create macros and tools that interact with DPN and Passport configuration data. See 241-6001-022 *Preside MDM Network Reporting System User Guide* for information about these macros and tools.

- NM Utilities

  The Network Model system also provides a number of macros that are useful when writing scripts to manage MDM. See 241-6001-015 *Preside MDM Network Model Administrator Guide* for information about these macros.

- rntp

  Like its graphical equivalent, Alarm Help, this utility displays the text describing the alarm identified on its command line according to its fault code.

- snmpsh

  This utility allows you to interact with SNMP compliant devices notably by issuing GET and SET commands to them.

Macros provided with the MDM software

This section summarizes the purposes of some of the macros provided with the Preside Multiservice Data Manager (MDM) software. All of the macros listed in this section, with the exception of ppDeltaDiag, are located in directory /opt/MagellanNMS/cfg/macros/nms. The macro ppDeltaDiag is located in the directory /opt/MagellanNMS/bin. The macros in this section can be run from the Command Console. You can run the macros from the Command Console as follows:

```
$<macro name> <macro arguments>...
```

Some of these macros can also be run from a UNIX Access window or from within a cron job. For more information on their behavior, use a UNIX editor to examine the contents of these macros.

The macros are as follows:

- DUMPACC, DUMPALA, DUMPALL, DUMPLOG, DUMPSTAT

  These macros allow you to trigger the dumping of DPN-100 spooling files. They expect a destination OA name, an NCS Capability ID, and a password as arguments. The list of modules to dump from must be created and stored in file /opt/MagellanNMS/cfg/dumpmodules.cfg. Refer to the macros themselves for usage instructions.

- LPDAMLA, LPDAMST, LPDARCON, LPDASLF, LPDASTSP, LPDATRR, LPDAWCON

  These macros allow administrators to control LPDA-compliant MODEMs and are described in 241-6001-011 *Preside MDM Fault Management User Guide*.

- acknowledge, dumpmodel, maintenance, makecurrent, savemodel, whichcurrent, whichmodels

  These macros relate to the Network Model system and are described in 241-6001-015 *Preside MDM Network Model Administrator Guide*.

- allcall

  This program displays all calls on a DPN-100 port. Run this macro from the Command Console with one of the following arguments:

```
$allcall <module> <pi> <po>

$allcall <OA> <module> <pi> <po>
```

- autosync

  This macro waits for a disk out of sync alarm from a DPN-100 module and automatically resynchronizes the disks. To be used, this macro must first be copied to the /opt/MagellanNMS/cfg/macros/user directory then modified to provide the capability and password to use to connect to the switch. Refer to the macro itself for usage instructions.

- cmdshell, cmdshell.menu

  The cmdshell macro is in fact the Operator Command tool that is run from the various fault management tools. The file cmdshell.menu defines the menus offered by the tool and can be customized by first copying it either into directory /opt/MagellanNMS/cfg/tsets or into your $HOME/MagellanNMS directory. Look at the contents of file cmdshell.menu file for customization instructions.

- dbnl_passwd_chg

  This macro allows you to change setup password for Dial Backup Network Links (DBNLs) on multiple DBNL DPN-100 ports at once. Look at the macro itself for usage instructions.

- dpndelta

  This macro issues two commands that are provided as its arguments and highlights the differences between the two resulting responses using the diff UNIX utility. Despite its name, this command can also be used to produce the same effect with Passport commands. See the information about the pprepdelta macro in this section for another form of Passport delta macro.

  For example, when used as follows, this macro highlights the difference between two DPN ports:

  ```
  $dpndelta "r78 6 3 d" "r72 4 4 d"
  ```

- dpnmcfhi

  This macro finds the highest matching MCF file on a DPN-100 module and, optionally, activates it. This macro is run as follows:

```
$dpnmcfhi [-a] <module> <MCF-key>
```

If `-a` is specified, the MCF is activated.

- dpnscan

  This macro executes the provided command and only displays response lines that match the specified pattern. It can be run from the Command Console as follows:

```
$dpnscan <command> <pattern>
```

- dpnup

  This macro executes the `DIR` command on the current OA route and displays the devices it contains that are up. This macro can be run from the Command Console as follows:

```
$dpnup
```

- ManClear

  This macro manually clears alarms matching a set of specifications from the command console. ManClear connects to a specified GMDR and extracts the list of active alarms based on the specified pattern and date ranges. Depending on the options specified with ManClear, you can list active alarms, locally or globally clear active alarms, and log output to a file. Note that the order of arguments is important when executing the macro. ManClear can be run from the command console as follows:

```
ManClear [help|?]
[-l] [-g] [-C] [-h <GMDR host>]
[-s <GMDR name>] <fault code pattern> <severity
pattern> <from: dd/mm/yy or *> [<to: dd/mm/yy>]
[-L [<log file>]] [-c <component ID pattern>]
```

where:

`help|?` displays information about the ManClear macro

`-l` lists the active alarms matching the specifications and exits the macro without clearing alarms. If the -l option is not specified, the matching alarms are automatically cleared.

`-g` performs a global clear on DPN or Passport alarms. If you don't use the -g option, or you have alarms other than DPN, clear the alarms in the Alarm Display or the Component Information Viewer. If a global clear fails, ManClear automatically performs a local clear.

`-C` prevents a local clear from automatically occurring when a global clear fails.

`-h <GMDR host>` specifies the GMDR Service Selected host.

`-s <GMDR name>` specifies the GMDR server.

`<fault code pattern>` specifies the fault code pattern. Patterns are specified in the GLOB style with an asterisk (*) or question mark (?). The fault code pattern matches the 8 hexadecimal digit format with FFs interpreted as wildcards.

`<severity pattern>` specifies the alarm severity. The alarm severity can be critical, major, minor, warning, or indeterminate.

`<from: dd/mm/yy or *>` and <to: dd/mm/yy> specifies the date range. The two-digit year is interpreted as follows: years 91 to 99 (inclusive) represent the twentieth century; years 00 to 90 (inclusive) represent the twenty-first century. If you specify the "from" date as an asterisk (*) for all alarms, do not specify the "to" date.

`-L [<log file>` logs the ManClear output and operations to a file. If you do not specify a log file name, ManClear uses the default file $HOME/ManClear.log.

-c <component ID pattern> specifies the component ID pattern. The component ID pattern can be API, canonical, or format.

- mi8, mi8_profiles

  This macro allows you to control and configure the MODEMs of a DPN-100 MI-8 Integrated MODEMs card. For a description of this macro, see 241-6001-011 *Preside MDM Fault Management User Guide*.

- ppDeltaDiag

  This macro lets you query Passport attributes at regular intervals and have the difference between the results of two consecutive queries computed and displayed automatically. It is especially useful for querying counter statistics. This macro can be run from the command console as follows:

```
/opt/MagellanNMS/bin/ppDeltaDiag <timeout>
[<repeat>][<attributes, space or comma separated>]
<Passport compoment ID>
```

An alternate call syntax can also be specified:

```
/opt/MagellanNMS/bin/ppDeltaDiag <timeout>[,<repeat>]
<Passport component ID>" \ ["<attributes, must be comma
separated>"]
```

where:

`<timeout>` is the amount of time to wait between the two samples.

`<repeat>` is the number of samples to take. By default <repeat> is one. If this value is specified, there must be a comma, and no space between the delay and repeat.

`<attributes, space or comma separated>` is an optional list of attributes and groups to display. Ensure that each attribute is separated or a space to separate the attributes and groups. All attributes are in CAS format, and are displayed by default.

`<attributes, must be comma separated>` is an optional list of attributes and groups to display. Ensure that each attribute is separated by a comma to separate the attributes and groups. All attributes are in CAS format, and are displayed by default.

`<Passport component ID>` is the component name to probe in display format. The component name can contain wildcards following the Passport Component Administration System (CAS) wildcarding conventions. Usual issues in Unix with wildcard components which should be avoided, or the whole component name should be in quotes.

You can also run ppDeltaDiag from the Component Information Viewer, as described in 241-6001-011 *Preside MDM Fault Management User Guide*.

Chapter 4
Creating snmpCmd macros

This section contains information to help programmers create snmpCmd command macros for devices that are monitored using SNMP protocol. These devices include Passport 4400 series devices.

This section contains the following information:

- "What you need to know" (page 113)
- "Runtime environment" (page 114)
- "About snmpCmd macros" (page 118)

What you need to know

Information in this section is directed at programmers who have:

- programming experience using Tool Command Language (TCL) as a scripting language and Scotty for writing interface routines to Simple Network Management Protocol (SNMP) devices.

  You are not limited to writing command macros using TCL and Scotty. Other languages such as Bourne shell can be used, provided that your macro provides the capability of setting up an SNMP session to the managed device. TCL and its extension language Scotty were chosen for creating the snmpCmd macros supplied with the Preside Multiservice Data Manager (MDM) software because Scotty has the advantage of loading in the standard ASN.1 MIB directly without the need to run the MIB through a MIB compiler.

- knowledge of the hardware structure and of the Management Information Bases (MIBs) for the SNMP devices being managed

For the location of directories in which these MIBs are stored, see also "Runtime environment" (page 114).

Runtime environment

Tool Command Language (TCL) and Scotty are used to create the snmpCmd macros that are provided with the Preside Multiservice Data Manager (MDM) software package.

TCL and Scotty executables are part of the Magellan Contrib MagTcl software package, which must be installed on the workstation along with the MDM software package to be able to run snmpCmd macros.

Paths to the TCL and Scotty executables are set up automatically when the Magellan Contrib MagTcl package is installed. Paths to these executables are as follows:

- TCL: /opt/MagellanContrib/bin/tclsh
- Scotty: /opt/MagellanContrib/bin/scotty

Environment variables related to Scotty and TCL

Environment variables TCL_LIBRARY and TCLLIBPATH need to be set in the set-up files of the UNIX user account from which macros are to be run.

When the account is set up to run the default Preside Multiservice Data Manager (MDM) environment, the values of these variables are set automatically and no intervention is required. The method used in the default MDM environment for setting up the values of the environmental variables is as follows:

- For user accounts that run C-shell, the values are set by script /opt/MagellanNMS/bin/nmscsh which is called by the .cshrc file in the user's home directory.
- For user accounts that run Korn or Bourne shell, the values are set by script /opt/MagellanNMS/bin/nmssh which is called by the .profile file in the user's home directory.

If the account is not using the default MDM user environment, you must take steps to ensure that these environment variables are set either when the user logs in or once login is complete.

TCL_LIBRARY

The value of variable TCL_LIBRARY is set to path /opt/MagellanContrib/lib/tcl17.5. This path provides access to the high level init script for TCL and to high-level utilities and error checking routines.

TCLLIBPATH

The value of variable TCLLIBPATH is set to the following paths:

- /opt/MagellanContrib/lib/tnm2.1.7
- /opt/MagellanContrib/lib/tkined1.4.7
- /opt/MagellanNMS/cfg/snm

If the 4400 Configuration software package is installed, an additional path of /opt/MagellanMPA/lib is appended to the values for this variable.

General purposes of the paths are as follows:

- Path /opt/MagellanContrib/lib/tnm2.1.7 provides access to the high level Management Information Bases (MIBs) that are common across all SNMP devices.
- Path /opt/MagellanContrib/lib/tkined1.4.7 provides access to TK routines for windowing and graphical user interface applications. This path is not currently used for snmpCmd macros but is provided for future use and for creating your own windowing applications.
- Path /opt/MagellanMPA/lib provides access to the init.tcl file, which specifies the location of the Passport 4400 series MIBs.

snmpCmd macros

A number of snmpCmd macros are provided with the Preside Multiservice Data Manager (MDM) software. These are located in the following subdirectories of MDM software library directory opt/MagellanNMS/lib/cfg/snm/Commands:

- GEN contains general command macros.

- MPA contains device-specific command macros for Passport 4400 series devices.

Any snmpCmd macros you write are stored in subdirectories GEN or MPA in customer configuration directory /opt/MagellanNMS/cfg/snm/Commands.

When attempting to locate a macro, the MDM software first searches the subdirectories of the customer configuration directory, then subdirectories of the MDM software library directory. If there are two macros with same name in these directories, the macro in the customer configuration directory takes precedence.

If there is a user-defined macro that overrides an MDM-supplied macro, an asterisk (*) appears after the macro name when you display the available commands by running the *showcmds* macro.

MIBs

A Management Information Base (MIB), is a set of files that list of all the variables you can use to retrieve management information about SNMP devices and their interfaces. Locations of the MIBs for SNMP devices used by snmpCmd macros are as follows:

- standard high-level MIBs provided with the Magellan Contrib software package: /opt/MagellanContrib/lib/tnm2.1.7/mibs
- MIBs for Passport 4400 series devices: /opt/MagellanMPA/mibs

Utilities

A number of TCL and Scotty utilities have been created for Preside Multiservice Data Manager (MDM) supplied macros. These are located in file /opt/MagellanNMS/lib/cfg/snm/Commands/libraryRoutines.tcl. A summary of these utilities is as follows:

- getIpAddr <device type> <device name>

  This utility attempts to get the IP address of an SNMP device from a number of locations, according to the <device type>. For example, Passport 4400 uses GMDR as the primary source of the IP address. If the attempt to get the address is unsuccessful, the utility returns a dash (-).

- getDevNames <device type> <device name>

  This utility obtains a list of devices for the specified <device type>. This utility can also be used to get subcomponent names of a particular device by specifying the additional <device name> parameter. For sample usage of this utility, look at files showdev and showcomp in directory /opt/MagellanNMS/dev/lib/cfg/snm/Commands/GEN.

- getIndexStringFromGMDR <comp id>

  This utility accepts a <comp id> parameter, which is used to fetch the SNMP index from DCD (through GMDR) for a specified component. An example of a <comp_id> is MPA MPADEV1 CARD 1 PO 1.

  The indexes are in the form <index_name>/<index_val>.

  The index name is actually the name of a MIB table which indicates that the index value can be applied to any element in the table. For example, if the index string returned is ifTable/1.1., it would be possible for the command macro to get the state of the component by doing an SNMP GET on ifOperStatus.1.1.

  This utility returns the index string returned from GMDR, or "-" if the index name was not found, in which case the invoking script can choose another course of action.

- getReadComString <device type> <device name>

  This utility gets a read community string value for the specified device type and/or device name. This value, if present, is stored in an environment variable in the form:

  <device_type> <device_name>_READ_COMMU_STR =<value>

  This variable can be set using the setreadcom general macro. If no such environment variable exists, the utility returns the default value of public.

  getWriteComString <device type> <device name>

  This utility gets a write community string value for the specified device type and/or device name. If present, this value is stored in an environment variable of the form:

  <device_type> <device_name>_WRITE_COMMU_STR=<value>

  This variable can be set using the setwritecom general macro. If no such environment variable exists, the utility returns the default value of *private.*

- getSnmpPort <device type> <device name>

  This utility gets the UDP port value used to receive SNMP messages for the specified device type and or device name. If present, this value is stored in an environment variable of the form:

  <device_type> <device_name>_SNMP_PORT=<value>

  This variable can be set using the setsnmpport general macro. If no such environment variable exists, the utility returns the default value of 161.

About snmpCmd macros

Writing *snmpCmd* macros requires a knowledge of existing Preside Multiservice Data Manager (MDM) supplied *snmpCmd* macros, the snmpCmd processor, and how snmpCmd macros interact with the command processor. This section provides a high-level view of this information.

A set of snmpCmd command macros is provided with the MDM software. These macros are of two types:

- general command macros (GEN) that can be applied to more than one type of SNMP device
- device-specific command macros that can only be applied to a specific type of SNMP device

General command macros

General command macros supplied by Preside Multiservice Data Manager (MDM) reside in directory /opt/MagellanNMS/lib/cfg/snm/Commands/GEN.

Commands for general snmpCmd macros

Commands to run these macros have the structure:

```
<command verb> [optional parameters]
```

where:

<command verb> is the name of an snmpCmd macro in subdirectory GEN of customer configuration directory /opt/MagellanNMS/cfg/snm/Commands or MDM software library directory /opt/MagellanNMS/lib/cfg/snm/Commands.

[optional parameters] define what the command verb is to act on. These depend on the specific implementation of the macro whose name is <command verb>.

Examples of general commands:

- help
- help gen showcmd
- showdev mpa
- showcomp mpa m31

To display the syntax of commands to run MDM-supplied *snmpCmd* macros, see 241-6001-804 *Preside MDM Workstation Utilities User Guide*.

Summary of general command macros

Because general command macros are equipped with built-in help that explains their purpose and command syntax, we have only summarized the command macros in this section.

The general command macros are as follows:

- help provides help for a specific Preside Multiservice Data Manager (MDM) supplied snmpCmd macro. The help information displayed includes the purpose of the macro and the command syntax for running the macro.
- showcmds displays a list of commands for a specified category of commands (GEN, MPA).
- showdev displays the IP address and state of all SNMP devices of a specified type (MPA) or of a specified SNMP device.
- showcomp lists all subcomponents of a specified SNMP device or of a specified component for an SNMP device.
- setreadcom creates an environment variable for a specified read community string for all SNMP devices of a specified type, or for a single specified SNMP device. This environment variable can then be used by any macro that wishes to establish read access to the SNMP device (or devices).

  If the command is entered without a read community string, the corresponding environment variable is removed and the default value of public is assumed.

- setwritecom creates an environment variable for a specified write community string for all SNMP devices of a specified type, or for a single specified SNMP device. This environment variable can then be used by any macro that wishes to establish write access to the SNMP device (or devices).

  If the command is entered without a write community string, the corresponding environment variable is removed and the default value of private is assumed.

- setsnmpport creates an environment variable for the port number used for establishing sessions to all SNMP devices of a specified type, or to a single specified SNMP device. This environment variable can then be used by any macro that wishes to establish a session with the SNMP device (or devices).

  If the command is entered without a read community string, the corresponding environment variable is removed and the default value of port 161 is assumed.

- setenvironment contains a template that allows you to run multiple setreadcom, setwritecom, and setsnmpport commands at a time to set environment variables at the beginning of a command console session. To use the macro, edit the macro and add the commands to the macro then enter setenvironment to run it.
- native allows you to direct commands in native SNMP protocol format to a specified SNMP device.

Device-specific snmpCmd macros

Device-specific command macros supplied by Preside Multiservice Data Manager (MDM) reside the following directory:

- macros for PP4400 : /opt/MagellanNMS/lib/cfg/snm/Commands/MPA

Commands for device-specific snmpCmd macros

Commands to run device-specific snmpCmd macros have the structure:

```
<device type> <device name> <command verb> \
[optional parameters]
```

where:

`<device type>` is the type of SNMP device such as MPA.

`<device name>` is the name of the SNMP device.

`<command verb>` is the name of an snmpCmd macro in subdirectory MPA in customer configuration directory /opt/MagellanNMS/cfg/snm/Commands or in the Preside Multiservice Data Manager (MDM) software library directory /opt/MagellanNMS/lib/cfg/snm/Commands.

`[optional parameters]` define what the command verb is to act on. These depend on the specific implementation of the macro whose name is <command verb>.

Example of device specific commands:

- mpa mpadev1 showstate card1 en1

To list the device-specific MDM-supplied snmpCmd macros available for each of the SNMP devices, to find the syntax of the commands to invoke them, and to list the optional parameters, see 241-6001-804 *Preside MDM Workstation Utilities User Guide*.

What happens when you run an snmpCmd macro

Commands to run *snmpCmd* macros can be entered at:

- the Command Console tool

  See the section on entering commands in 241-6001-804 *Preside MDM Workstation Utilities User Guide*.

- any application that uses the command console API (cmcmd)
- the RNCS tool

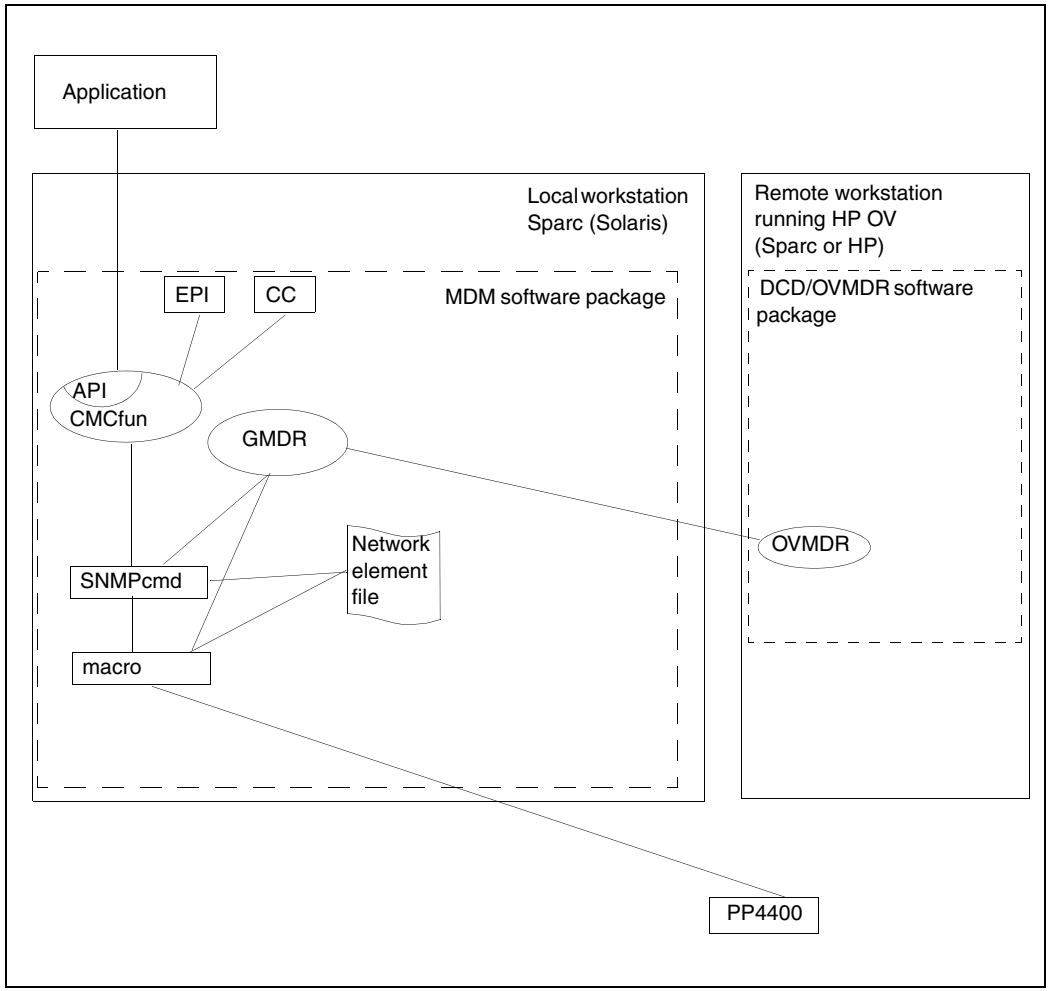
  See the section on starting RNCS in 241-6001-013 *Preside MDM Remote Network Communication System User Guide*.

- the Embedded Programming Interface (EPI)

  See the command access section in 241-6001-211 *Preside MDM Embedded Programming Interface Reference Guide*.

Commands must be directed to the @ route to ensure that they are directed to snmpCmd processor which then locates the macro and invokes it with the required variables.

Figure 6
Dataflow diagram for snmpCmd command macros



Running an snmpCmd macro involves the following steps (See the figure "Dataflow diagram for snmpCmd command macros" (page 123)):

1 The user sets the command route to @ to indicate that the command is to run an snmpCmd macro, in one of the following ways:

If you use the Command Console, select @ with the Route menu button.

If you use the API of the CMCFun server (cmcmd), prefix the command with @.

If you use the RNCS tool, prefix the command with @.

2 The user enters the prefix (if required) followed by the command to run the macro, along with the required parameters.

3 The CMCFUN server recognizes the @ route and passes the command to the snmpCmd processor.

4 The snmpCmd processor determines the command category (GEN, MPA).

5 The snmpCmd processor converts the command verbs to lowercase. This means that command verbs can be entered in lowercase, uppercase, or mix of uppercase and lowercase.

6 The snmpCmd processor searches for the macro according to the command category, as follows:

It searches in one of the customer configuration directories (/opt/MagellanNMS/cfg/snm/Commands/<category>)

Then it searches in one of the Preside Multiservice Data Manager (MDM) software library directories (/opt/MagellanNMS/lib/cfg/snm/Commands/<category>).

If it finds two macros with the same name, the macro in the customer configuration directory takes precedence.

7 If the macro is device-specific, the snmpCmd processor obtains the IP address of the device as follows:

For PP4400, the snmpCmd processor sends a request to the GMDR server. GMDR sends a property request to OVMDR running on a remote workstation through the PSERVER. OVMDR automatically obtains and updates information about SNMP components, including the IP addresses of PP4400s. This notification requests injection of device and component information, including the IP address. The IP address is injected back to IMDR through the PSERVER and is supplied to GMDR and to the snmpCMD processor.

8 The snmpCmd processor calls the macro and passes it the required options and parameters as follows:

For a general macro the command structure is <command verb> <optional parameters>. The snmpCmd processor passes the optional parameters to the macro.

For a device-specific macro the command structure is <device type> <device name> <command verb> <optional parameters>. The snmpCmd processor passes arguments to the macro in the following order:

- the device type
- the device name
- the IP address or "-" if the IP address cannot be obtained

  Having the snmpCmd process pass "-" to indicate that the IP address cannot be found provides the macro with the opportunity to take alternative action, such as displaying the response: no such IP address.
- the optional parameters

9 The macro runs. If a device-specific macro is involved, the macro gets the SNMP device index from GMDR for PP4400 devices, translates the parameters into an SNMP command, and routes the command to the managed device.

10 Replies and error messages are returned to the originating application.

Chapter 5
Extracting alarms in text format

This section contains instructions for extracting alarms in text format by means of the rncsalarm utility. This section contains the following information:

- "About the rncsalarm utility" (page 127)
- "Command syntax" (page 128)

About the rncsalarm utility

The rnsmclarm utility can be used to extract alarms from Preside Multiservice Data Manager (MDM) and display them, redirect them to a file or pipe them to another system. You can also use the Alarm Display tool to save alarms to a file, however, an operator must be logged in to MDM.

Consider the following alternatives to the rncslarm utility for extracting alarms from MDM:

- using the Alarm and Status API described in 241-6001-203 *Preside MDM Alarm and Status API Reference Guide*

  Because of its field selection and filtering capabilities, and its regular (parsable) output format, the Alarm and Status API is a better source of information, especially for alarms that need to be processed by other applications, such as an umbrella management system.

- using alarm on-switch alarm spooling facilities

  For devices that support it, spooling of alarms is a more reliable way of collecting alarms for later analysis.

When it is run without the -d and -h options (the default situation), the rncsalarm utility continuously receives matching alarms from a specified GMDR server and displays them in a specified format and mode. When it is run with the -d and -h options, the rncsalarm utility dumps the matching contents of the Active Alarm list (-d option) or the Alarm History list (-h option).

Command syntax

The command syntax for the rnscalarm utility is as follows:

```
/opt/MagellanNMS/bin/rncsalarm \
[-d] \
[-h] \
[-f TERSE|NORMAL|FULL] \
[-m DPN|COMMON] \
[-i "<component name>"]... \
[-n <NTP index>]* \
[-s <severity>]* \
[-l <server name>] \
[-H <server host>]
```

where:

[-d] dumps the matching Active Alarms

[-h] dumps the matching alarms from the Alarm History list

[-f TERSE|NORMAL|FULL] displays the alarms in TERSE, NORMAL, or FULL format. By default, alarms are dumped in TERSE format. For an explanation of the common alarm format see 241-6001-011 *Preside MDM Fault Management User Guide*.

[-m DPN|COMMON] displays the alarms in DPN or COMMON mode. By default the mode is DPN. For an explanation of these modes, see 241-6001-011 *Preside MDM Fault Management User Guide*.

[-i "<component name>"]... displays only the alarms for components that match the specified component name. Adding an asterisk (*) after the component name displays all alarms matching the specified component name. You can specify up to 5 -i options. When you specify more than one

component name, alarms matching any of the component names are displayed. The component name must be specified in API format, with blank separators and no slashes. For example, EM TOTO LP 2, not EM/TOTO/LP/2.

[-n <NTP index>]* displays only the alarms whose NTP index matches the specified NTP index. You can specify up to 5 -n options.

[-s <severity>]* displays only the alarms whose severity matches a specified alarm severity. Valid severities in DPN mode display are: NONE, DEGRADE, OVERLOAD, MAJOR, and MINOR. For COMMON mode display, valid severities are: UNKNOWN, WARNING, MAJOR, MINOR and CRITICAL. You can specify up to 5 -s options.

[-l <server name>] lets you name an alternate GMDR server to connect to. By default, the server name used is GMDR.

[-H <server host>] lets you override the host on which the connected GMDR server is running. By default, the host that has been selected for Surveillance Access with the Service Selection tool is used as the host on which the GMDR server is running.

> ***Note:*** If -i, -n, and/or the -s options are specified, only alarms matching all specified sets of filters are displayed.

Chapter 6
Customizing the Component Information Viewer diagnostics

This section describes how to customize the Component Information Viewer (CIV) Diagnostic menu and details available CIV diagnostic utilities. This section contains the following information:

- "Diagnostic menu management" (page 131)
- "CIV diagnostic utilities" (page 133)

Diagnostic menu management

You can customize the diagnostic commands that appear in the CIV diagnostic menus. You customize the diagnostic menu the same as any other Start Tool menu or tool. Customizing the diagnostic menu lets you specify the following:

- your own command line using substitution variables for the target component
- labels
- patterns to determine when the command applies
- tests to determine whether or not a command appears in the menu

When customizing, you can use wild cards and presence test patterns.

CIV Diagnostics supports a resource line that lets you specify the use of an input dialog. The resource line is as follows:

```
queryParameter:    [<substitution variable name>] \
<prompt string>
```

Example

```
queryParameter:    $VAL Nb seconds between polls:

queryParameter:    Lock the component?
```

If you specify a substitution variable name using the `$` prefix, an input dialog opens and prompts for a value. The input value is then substituted in the command line where the identified variable name is used. If you do not specify a substitution variable, a confirmation dialog opens and prompts to continue or abort the command execution.

The CIV diagnostic menus support the existing tMAction resource. This resource specifies that a command is to be executed by the Preside MDM window, and not directly. Consequently, no output from the command is expected or displayed in the CIV window. The syntax is as follows:

```
tMAction:  startTool <command and arguments>
```

The default CIV Diagnostic menu is configured in the file */opt/MagellanNMS/lib/tsets/C/CIVDiagnostics.menu*. The file includes some default items and all item description files found in the following file:

```
<search path>/dial/*.menu>
```

where:

`<search path>` is the usual search path sequence. (see “Automatic search path” (page 49)).

To add new diagnostic commands, you need only add a file in the directory

- *$HOME/MagellanNMS/diag* to customize a single user
- */opt/MagellanNMS/cfg/tsets/$LANG/diag/* to customize a workstation, (where $LANG is C for standard-ANSI/English language)

The default CIV diagnostic menus provided with Preside Multiservice Data Manager (MDM) are located in the directory /opt/MagellanNMS/lib/tsets/$LANG/diag/ and consist of the following files:

- 10_PPDiagnostics.menu
  for Passport related commands
- 15_PPTests.menu
  for Passport related tests and traces
- 20_PPInventory.menu
  for Passport Inventory reports
- 30_DPNDiagnostics.menu
  for DPN related commands
- 40_DPNInventory.menu
  for DPN Inventory reports

CIV diagnostic utilities

To assist in the creation of diagnostic commands based on device command macros, use the following utilities:

- "execDPNCommand" (page 135)
- "execDPNCommand" (page 135)
- "execPPCommand" (page 136)
- "ppCompSelector" (page 138)
- "execWithIPAddr" (page 138)

execWithDest

The execWithDest utility performs the following activities:

- identifies the default destination as specified in the CIV Diagnostic command *Select Command Route*
- opens the Connection Console to establish the connection when needed
- executes a specified command in the context of that connection

The $CMC_CURRENT_ROUTE environment variable stores the selected destination name. This environment variable is provided at script run time. It is not a substitution variable of the MDM menu system. If you specify $CMC_CURRENT_ROUTE as a command line argument to a diagnostic command, then you must ensure that the command is escaped (\\$CMC_CURRENT_ROUTE) so that it will substitute only at final execution time.

The command line is as follows:

```
/opt/MagellanNMS/bin/execWithDest
[-inline]
[-ask|-noask]
[-oa|-group]
[-def <default dest>]
[-mod <module name>]
[-name <command name>]
[<command and args]
```

where:

`-inline` is used when execWithDest is invoked from a DtKsh script. The utility is sourced rather than invoked as a subprocess (. /opt/MagellanNMS/bin/execWithDest -inline…). This option directs the utility not to drop the EPI Command Interface connection so that the connection can be used again by the sourcing script.

`-ask|-noask` controls whether the Connection Console is invoked. If you specify -ask, the Connection Console always opens. If you specify -noask, the Connection Console never opens and the command does not execute if there is no current default applicable.

`-oa|group` specifies the destination type

`-def <default destination>` specifies the default destination name.

`-mod <module name>` specifies the name of the module and is used in selecting an appropriate connected destination, if possible (Passport)

`-name <command name>` specifies a meaningful name for the command to be executed. This option is used only for prompts and messages.

<command and args> specifies the command line to execute, in context of the selected destination. When you source the utility into a DtKsh script, usually with the -inline option, do not specify the command and arguments.

execDPNCommand

The execDPNCommand utility executes a single DPN command as a macro in the context of an OA route similar to “execWithDest -OA”. The command line is as follows:

```
/opt/MagellanNMS/bin/execDPNCommand
[-msg <message>]
[-pe|-pi|-po]
“<command>” “<arguments>” “<component>”
```

where:

-msg <message> outputs the value of a message on the standard output stream before the command output.

-pe|-po|-pi identifies the level to which the component ID is interpreted. Superfluous component ID elements are dropeed and the utility converts the remainder to the appropriate component ID format.

<command> specifies the command to execute, in context of the selected destination

<arguments> specifies the command line arguments for <command>

<component> specifies the component ID in canonical format ($COMP, space separator) that will be used by <command>

The execDPNCommand analyzes the component ID and determines the module name and subcomponent to be used for the specified command. The execDPNCommand executes the specified command against the identified component ID as follows:

```
<module> [<subcomponent spec.>] <command> <arguments>
```

The component ID is interpreted as a subcomponent. If you specify a -pe, -pi, or -po option, the utility interprets the component ID only to the specified level. For example, the following command invokes the Display Hardware command on the corresponding PE, even if the target component is more specific.

```
/opt/MagellanNMS/bin/execDPNCommand -pe "d" \
"hard" "$COMP"
```

Using the example above with a $COMP value of `PM TOTO PE 1 PI 2 PO 2`, the following command executes:

```
<OA dest> TOTO PE 1 d hard
```

where:

`<OA dest>` is the OA destination selected by `execWithDest`

execPPCommand

The execPPCommand utility executes a single Passport CAS command as a macro in the context of a group route similar to "execWithDest -group". The command line is as follows:

```
/opt/MagellanNMS/bin/execPPCommand
[-msg <message>]
[-level <category>]
[-selector <patterns>]
"<command and options>" "<arguments>" "<component>"
```

where:

`-msg <message>` outputs the value of a message on the standard output stream before the command output

`-level <category>` identifies the category to which the component ID is interpreted. Superfluous component ID items are ignored

`-selector <patterns>` appends the specified patterns (separated by a | character) to the interpreted subcomponent to produce a set of component patterns. These patterns are used by the ppCompSelector utility to open a dialog listing all the matching components existing on a target node. The component selected from the dialog is used for the command execution. This option allows you to create diagnostic commands for components that are not

managed by fault tools and do not appear in the CIV Related Components List. For a description of ppCompSelector, see “ppCompSelector” (page 138).

The execPPCommand utility analyzes the component ID and determines the module name and subcomponent to be used for the specified command. The utility executes the specified command against the identified component ID as follows:

```
<module> [<subcomponent spec.>] <command> <arguments>
```

The component ID is interpreted as a subcomponent. If you specify a -level option, the utility interprets the component ID only to the specified level. For example, the following command executes the ping command on the IP ICMP component of the named Virtual Router (VR), even if the target component is a VR subcomponent:

```
/opt/MagellanNMS/bin/execPPCommand -level VR \
“ping -ipAddr ($VAL) -traceRoute” “IP ICMP” “$DCOMP”
```

Using the example above with a $COMP value of `EM TOTO VR 12 PP FRDTE102`, the following command executes:

```
<group dest> TOTO ping -ipAddr (<value>) -traceRoute \
VR/12 IP ICMP
```

where:

`<group dest>` is the destination selected by execWithDest

`<value>` is the result of a variable substitution specified through a query parameter dialog

The $VAL is a prompting dialog substitution variable. The execPPCommand arguments are used to construct the actual component ID that will be used for the command after the -level option interpretation.

ppCompSelector

The `ppCompSelector` utility provides a selection dialog with a list of Passport components matching the specified patterns. The command line is as follows:

```
/opt/MagellanNMS/bin/ppCompSelector
[-title <dialog title>]
<group name> "<component patterns>"
```

where:

`-title <dialog title>` is the title in the dialog bar

`<group name>` specifies a group to be used for command access

`<component patterns>` specifies full Passport component IDs in Preside Multiservice Data Manager (MDM) display format. Specify components using Passport CAS wild cards. Separate multiple patterns with the | character. If you use wild card in the module name, all matching nodes within the specified group are used. The group is also used to query the target modules for the matching subcomponents.

For example, the following command opens a dialog with all LPs of all nodes in the UNIVERSE group:

```
ppCompSelector UNIVERSE "EM/* LP/*"
```

In the following example, all circuits of a specific ATM interface are used:

```
ppCompSelector UNIVERSE "EM/NODEA05 ATMIF/122 VPC/
*|EM/NODEA05 ATMIF/122 VPT/*|EM/NODEA05 ATMIFf/122
VCC/*|EM/NODEA05 ATMIF/122 VPT/* VCC/*"
```

execWithIPAddr

The execWithIpAdd utility extracts an IP address or community string from the Preside Multiservice Data Manager (MDM) system for a specified component ID. The IP address is then substituted in the command line and the command is executed. The command line is as follows:

```
/opt/MagellanNMS/bin/execWithIPAddr
[-warn|-best-effort] <component ID> <command line>
```

where:

`-warn` displays a warning dialog when the IP address or community string cannot be identified and the command line does not execute. By default, the execWithIPAddr utility does not display warnings when the properties cannot be identified.

`-best-effort` executes the command line when the IP address and community string cannot be identified, substituting an empty string for the the matching parameter in the command line.

`<component ID>` specifies the component or subcomponent for which the IP address and/or community string is required. If the component ID identifies a sub-component and the property cannot be extracted at that level, execWithIPAddr automatically tries to fetch them at the module level.

<command line> specifies the command line to be executed. The utility substitutes any occurrence of %IPADDR% and %COMMUNITY% in the command line to the extracted IP address or community string, respectively. If `-best-effort` is specified and the property cannot be identified, nothing is substituted in the command line.

You can use this utility to start a Telnet session in context of a selected component ID, for example

```
/opt/MagellanNMS/bin/execWithIPAddr -best-effort \
"$COMP" "/bin/telnet %IPADDR%"
```

To enable context launching, specify the preceding command line in the MDM device-type specific tools-menu configuration files. For details on customizing the toolsets and Start Tool menus, see 241-6001-301 *Preside MDM Customization Administrator Guide*.

Index

B

bindings 18

C

cmccmd program 75
cmcwrap program 93
Customizing
- icon bars 61
- Network Viewer 66
- push-buttons 61
- Start tools menus 35
- startup options 66
- Toolsets 31

customizing
- MDM fonts 22
- MDM resources 15
- Network Viewer 21

M

Macros
- cmccmnd program 75
- cmcwrap program 93
- Command Console examples 86
- creating a connection 76
- definition 70
- determining if a connection exists 81
- dropping a connection 77
- in menu item definition files 66
- invoking from Command Console 83
- issuing operator commands 78
- listing available routes 80
- outside an MDM session 72
- provided with MDM software 95
- servers and programs for 73
- stand-alone 92
- storage directories 75
- structure 74
- within an MDM session 70
- …See also snmpCMD macros

R

resources
- definition files 20
- directories 19
- guidelines for customizing 20
- in a command line 27
- in the Xdefaults file 26
- predominance 17
- syntax 16
- utilities for modifying 27

S

Scotty 113, 114
snmpCMD macros
- environment variables 114
- GEN 115, 120
- how they work 123
- MIBs 116
- MPA 116, 121

T

TCL 113, 114
tool startup options 66

W

widgets 18

Preside Multiservice Data Manager

Customization

Administrator Guide

R14.3

Publication: 241-6001-301
Document status: Standard
Document version: 14.3RSUP
Document date: August 2003
Printed in Canada

NORTEL NETWORKS