



Performance from Experience

MYNAH™ System Administration Guide

A Module of Enterprise Testing Products

Telcordia Technologies System Documentation
BR 007-252-001
Issue 5, August 1999

Release 5.4

Telcordia Technologies, Inc. Confidential — Restricted Access
This document and the confidential information it contains shall be distributed, routed or made available solely to authorized persons having a need to know within Telcordia, except with written permission of Telcordia.

An SAIC Company

MYNAH™ System Administration Guide

Prepared for Telcordia Technologies by: Learning Support

Target audience: Testing administrators

This document replaces: BR 007-252-001, Issue 4

Where material has been added, changed, or deleted, the location of the change is marked by a vertical bar (|) in the outer margin next to the change.

Related document: BR 007-252-002, BR 007-252-004

For further information, please contact:

MYNAH Customer Service Center
1-(732) 699-2668, option 3

To obtain copies of this document, contact your company's document coordinator or call 1-800-521-2673 (from the USA and Canada) or 1-732-699-5800 (all others), or visit our Web site at www.telcordia.com. Telcordia employees should call (732) 699-5802.

Copyright © 1997-1999 Telcordia Technologies, Inc. All rights reserved.

Project Funding Year: 1999

Document Feedback

We at telcordia are constantly striving to meet your need for information. Once you've had a chance to use this document that we've written for you, please let us know if it met your needs. Please complete this form and either FAX it to us at (732) 336-3345 or return it to us at the address below.

Document No. BR 007-252-001	Issue No. Issue 5	Publication Date August 1999	Revision No.	Supplement No.
--------------------------------	----------------------	---------------------------------	--------------	----------------

1. In each of the following areas, how well did this document meet your need for information?

		Nearly Met	Met	Exceeded	Not Applicable
a. Relevance of the information to your work	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
b. Ease of finding the information that you need	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
c. Clarity of the information.....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
d. Accuracy of the information	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
e. Usefulness of the information	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
f. Thoroughness of the information	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
g. Level of detail of the information.....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
h. Availability of this document when you needed it	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
i. Overall quality of this document	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>

2. Please comment on any of the areas where this document *did not* meet or exceed your need for information.

3. Are there features of this document that you found particularly useful or informative? Please explain.

4. Are there other ways that we can improve this document? Please feel free to comment on any aspect of it.

5. For what purpose did you use this document?

- | | | |
|---|---|--|
| <input type="checkbox"/> As a technical reference | <input type="checkbox"/> To use a system | <input type="checkbox"/> To learn methods/procedures |
| <input type="checkbox"/> As an administrative reference | <input type="checkbox"/> To install/administer a system | <input type="checkbox"/> To be better informed |
| <input type="checkbox"/> Other (please specify) _____ | | |

6. Please tell us something about yourself.

Your company/employer _____ Your title _____

Your job responsibilities _____

If you would like us to let you know what we're doing in response to your feedback, please write your name and address (or telephone number) below.

Name _____ Telephone Number _____

Address _____

Thank you for your time and cooperation!

To return this form, please FAX it to (732) 336-3345, or mail it to Ken Berczik, telcordia Learning Support, 444 Hoes Lane, Room RRC 2B-183, Piscataway, NJ 08854.

NOTICE OF DISCLAIMER AND LIMITATION OF LIABILITY

This document is intended for use solely by customers of Telcordia Technologies who have licensed the Telcordia software described herein. The software, this document, and the information contained within this document may be used, copied or communicated only in accordance with the terms of a written license agreement with Telcordia. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording without the prior written permission of Telcordia. While the information contained herein has been prepared from sources deemed reliable, Telcordia reserves the right to revise the information without notice, but has no obligation to do so. Unless the recipient has been expressly granted a license by Telcordia under a separate applicable written agreement with Telcordia, no license, express or implied, is granted under any patents, copyrights or other intellectual property rights. Use of the information contained herein is in your sole determination and shall not be deemed an inducement by Telcordia to infringe any existing or later-issued patent, copyright or other intellectual property rights.

TELCORDIA PROVIDES THIS DOCUMENT “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR AGAINST INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS. FURTHER, TELCORDIA MAKES NO REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED WITH RESPECT TO THE SUFFICIENCY, ACCURACY, OR UTILITY OF ANY INFORMATION OR OPINION CONTAINED HEREIN. TELCORDIA EXPRESSLY ADVISES THE USER THAT ANY USE OF OR RELIANCE UPON SAID INFORMATION OR OPINION IS AT THE SOLE RISK AND LIABILITY, IF ANY, OF THE USER AND THAT TELCORDIA SHALL NOT BE LIABLE FOR ANY DAMAGE OR INJURY INCURRED BY ANY PERSON ARISING OUT OF THE SUFFICIENCY, ACCURACY, OR UTILITY OF ANY INFORMATION OR OPINION CONTAINED HEREIN. TELCORDIA, ITS OWNERS AND AFFILIATES SHALL NOT BE LIABLE WITH RESPECT TO ANY CLAIM BEYOND THE AMOUNT OF ANY SUM ACTUALLY RECEIVED IN PAYMENT BY TELCORDIA FOR THE DOCUMENTATION, AND IN NO EVENT SHALL TELCORDIA, ITS OWNERS OR AFFILIATES BE LIABLE FOR LOST PROFITS OR OTHER INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

Nothing contained herein is intended as a recommendation or endorsement of any product.

For further information, please contact:

The MYNAH Customer Service Center 8:00 AM and 7:00 PM ET Monday through Friday, (732) 699-2668, Option . If outside the 732 area, call (800) 795-3119, Option 3.

You can also contact support (for non critical problems) via e-mail at mynah-support@telcordia.com.

Copyright © 1999 Telcordia.
All Rights Reserved.

Trademark Acknowledgments

Adobe, Acrobat, and Acrobat Reader are registered trademarks of Adobe Systems Incorporated.

HP and HP-UX are trademarks of the Hewlett-Packard Company.

IBM is a trademark of International Business Machines.

Macintosh is a registered trademark of Apple Computer, Inc.

Microsoft, Windows, and NT are registered trademarks of Microsoft Corporation.

MYNAH is a trademark of Telcordia Technologies, Inc.

Ismonitor and Isdecode are copyrights of Viman Software.

Oracle is a registered trademark of Oracle Corporation

SunOS, Solaris, SPARC, SPARCstation, Sunlink, and NFS are trademarks of Sun Microsystems, Inc.

Telexel is a trademark of Telcordia Technologies, Inc.

Transarc is a registered trademark of Transarc Corporation

QA Partner is a trademark of Segue Software, Inc.

UNIX is a registered trademark of the Open Group in the US and other countries.

X Window System is a trademark of the Massachusetts Institute of Technology.

MYNAH System Administration Guide

Contents

Preface	Preface-1
Document Structure	Preface-1
Related Documents	Preface-2
On-line Versions of the MYNAH Documents	Preface-3
Formatting Conventions	Preface-4
1. Introduction.....	1-1
1.1 MYNAH System Overview	1-1
1.2 Architecture Overview	1-2
1.2.1 Script Engine (SE)	1-2
1.2.2 Script Dispatcher (SD)	1-2
1.2.3 Trigger Daemon (TD)	1-2
1.2.4 Boot Daemon (BD)	1-2
1.2.5 Graphical User Interface (GUI)	1-3
1.2.6 Command Line User Interface (CLUI)	1-3
1.2.7 Application-to-Application Main Collector Processes (CL)	1-3
1.2.8 Background Execution Environment (BEE).....	1-3
1.2.8.1 Why Background Execution.....	1-4
1.2.8.2 Why SE Groups	1-5
1.2.8.3 Why Multiple SDs	1-6
1.2.9 MYNAH Architecture Abstract View	1-7
1.3 System Requirements.....	1-8
1.4 Third Party Software Requirements.....	1-9
1.4.1 3270 Terminal Package.....	1-9
1.4.2 3270 Printer Package	1-10
1.4.3 Asynchronous Terminal Package.....	1-10
1.4.4 DCE Package	1-11
1.4.5 GUI Package	1-11
1.4.6 AppApp Package.....	1-12
1.4.7 Batch Package.....	1-12
1.4.8 Miscellaneous.....	1-13
1.5 Script Engine (SE).....	1-14
1.5.1 SE Types	1-14
1.5.1.1 Background.....	1-14
1.5.1.2 Embedded	1-14
1.5.1.3 Command-line	1-14
1.5.2 SE Execution Modes.....	1-15
1.5.2.1 Stateless Mode	1-15

1.5.2.2	ConnOnly Mode	1-15
1.5.2.3	FullState Mode.....	1-15
1.6	Administrative Commands Overview	1-16
1.6.1	Administrative CLUI Commands	1-16
1.6.2	Administrative Tcl Commands	1-17
1.7	Hardware and Operating System Requirements	1-17
1.8	Networking.....	1-18
1.9	On-line Documentation	1-19
1.10	Customer Support.....	1-19
2.	Installation.....	2-1
2.1	Introduction	2-1
2.2	Preliminary Background Information	2-2
2.2.1	Assumptions and Recommendations	2-2
2.2.2	Environment Settings and xmyProfile and xmyLogin.....	2-3
2.2.3	Starting and Stopping MYNAH Software Packages on the Solaris Platform.....	2-5
2.2.4	Requirements	2-5
2.3	Preliminary Actions	2-5
2.3.1	Obtaining License Keys	2-5
2.3.2	Creating the mynah Group and MYNAH Administrator logid (madmin).....	2-7
2.3.3	Setting /tmp_mnt (HP-UX Only).....	2-7
2.3.4	BAIST Considerations	2-8
2.4	Installing the MYNAH System.....	2-11
2.4.1	MYNAH System Preinstallation Steps	2-11
2.4.2	MYNAH Software Installation	2-11
2.4.2.1	MYNAH Software File Archive Installation.....	2-12
2.4.2.2	MYNAH Post Installation Steps.....	2-13
2.4.3	Changes to /etc/services and /etc/system	2-13
2.5	Installing the Telexel System.....	2-14
2.5.1	Telexel System Preinstallation Steps	2-14
2.5.2	Telexel Installation.....	2-15
2.5.2.1	Telexel Software File Archive Installation.....	2-15
2.5.2.2	Telexel Post Installation Steps.....	2-16
2.5.3	Configuring the Telexel System	2-16
2.5.4	Verifying the Telexel System	2-18
2.6	Installing Oracle (Optional)	2-20
2.6.1	Oracle Preinstallation Steps	2-20
2.6.2	Oracle Installation	2-25
2.6.2.1	Verifying an Installation.....	2-27
2.6.2.2	Oracle Error Messages.....	2-27
2.6.3	TNS Network Configuration	2-28
2.6.4	Configuring the MYNAH System Oracle Database	2-28
2.6.5	Dropping the Oracle Database	2-33

- 2.6.6 Verifying Oracle2-34
- 2.7 Installing I/O Concepts' X-Direct TN3270 (Optional).....2-36
 - 2.7.1 Installing X-Direct TN32702-36
 - 2.7.2 Configuring X-Direct TN3270.....2-36
 - 2.7.3 Verifying X-Direct TN3270.....2-37
- 2.8 Installing TOPCOM (Optional)2-39
 - 2.8.1 TOPCOM System Preinstallation Steps2-39
 - 2.8.2 TOPCOM Installation2-40
 - 2.8.2.1 File Archive Installation2-40
 - 2.8.2.2 TOPCOM Post Installation Steps2-40
 - 2.8.3 TOPCOM Configuration.....2-41
- 2.9 Installing Third-Party GUI-Test Tools — QA/Partner (Optional)2-42
 - 2.9.1 QA/Partner Preinstallation Steps2-42
 - 2.9.2 Installing and Configuring QA/Partner2-42
- 2.10 Installing the License Keys2-43
- 2.11 Configuring a Minimal Working Installation2-44
- 2.12 Verifying the MYNAH System2-46
- 2.13 Installing the On-line Documents2-48
 - 2.13.1 Installing the PDF Files.....2-48
 - 2.13.2 Obtaining the Acrobat Reader.....2-48
 - 2.13.2.1 Obtaining the Acrobat Reader as a File Archive2-49
 - 2.13.2.2 Obtaining the Acrobat Reader from Adobe.....2-49
 - 2.13.3 Accessing the PDF Files2-49
- 3. Customizing a MYNAH Configuration3-1
 - 3.1 Introduction3-1
 - 3.1.1 The xmyConfig File3-1
 - 3.1.2 The xmyConfigOP File3-3
 - 3.2 Environmental Variables.....3-3
 - 3.2.1 XMYDIR3-4
 - 3.2.2 XMYHOME.....3-5
 - 3.2.3 XMYSD3-6
 - 3.2.4 XMYSEGROUP3-7
 - 3.2.5 XMYSUT3-7
 - 3.3 The xmyConfig File Syntax3-8
 - 3.3.1 General Configuration - xmyConfig.General3-11
 - 3.3.2 MYNAH Package Configuration3-14
 - 3.3.2.1 Asynchronous Configuration - TermAsync.....3-14
 - 3.3.2.2 3270 Configuration - Term3270.....3-16
 - 3.3.2.3 TOP/AppApp/PRT3270/TCP Configuration.....3-19
 - 3.3.2.3.1 Protocol Handler Configuration3-20
 - 3.3.2.3.2 Message Collector Configuration.....3-23
 - 3.3.2.4 GUI Test Configuration.....3-25
 - 3.3.2.4.1 QA Partner GUI Test Tool Configuration.....3-25

3.3.2.4.2	GUI Test Script Engine (GTSE) Configuration	3-27
3.3.2.4.3	MYNAH System Include Files and Accessing the Symbol Table	3-30
3.3.3	Defining Script Engines (SEs), Script Engine Groups, and Script Dispatchers (SDs)	3-31
3.3.3.1	Script Engine Configuration - Engine	3-31
3.3.3.2	Engine Group Configuration - EngineGroup	3-37
3.3.3.3	Script Dispatcher Configuration - Dispatcher	3-38
3.3.4	Complete xmyConfig Example.....	3-40
3.4	The xmyConfigOP File Syntax	3-43
3.4.1	General Entry	3-44
3.4.2	Process Entries	3-44
3.4.2.1	.xmyStartup.....	3-45
3.4.2.2	MYNAH Process Start, Stop, and Status Commands	3-46
3.4.2.2.1	BD Commands	3-46
3.4.2.2.2	TD Commands	3-46
3.4.2.2.3	SD Commands.....	3-46
3.4.2.2.4	CL (Collector) Commands	3-48
3.4.2.2.5	TN3270 (Prt3270) Printer Emulator Commands.....	3-48
3.4.3	OperabilityAgent.....	3-50
3.4.4	Example xmyConfigOP File.....	3-51
3.5	The .xmyMYNAHrc File	3-56
3.6	Specialized Configuration Concerns	3-56
3.6.1	Batch Package Configuration — The .netrc File	3-56
3.6.2	Maintaining a Tcl Procedures Library	3-57
3.6.3	Synchronizing Clocks for the MYNAH System.....	3-57
3.6.4	Overriding Default ASCII-EBCDIC Translations	3-59
4.	Operability Management — Starting and Stopping MYNAH Processes	4-1
4.1	Basic Steps	4-1
4.2	Overview	4-2
4.3	Operability Design	4-4
4.3.1	Operability Manager (OM)	4-4
4.3.2	Operability Agent (OA)	4-4
4.3.3	Operability xmyOM Subcommands	4-5
4.4	Licensing	4-7
4.4.1	MYNAH Licensing Commands.....	4-7
4.4.2	Starting the License Server	4-7
4.4.3	Stopping the License Server	4-8
4.4.4	License Utilities	4-8
4.4.4.1	Monitoring	4-8
4.4.4.2	Decoding.....	4-9

4.5	Starting Processes.....	4-10
4.5.1	Solaris Start-up Mechanism.....	4-10
4.5.2	HP-UX Start-up Mechanism.....	4-10
4.5.3	Starting at Boot Time.....	4-11
4.5.3.1	.xmyRemovePips.....	4-11
4.5.3.2	xmyStartUp.....	4-12
4.5.4	Starting a Specific Process.....	4-13
4.5.5	Starting Autostart Processes.....	4-13
4.6	Stopping Processes.....	4-14
4.6.1	Stopping a Specific Process.....	4-14
4.6.2	Stopping Autostart Processes.....	4-14
4.6.3	Stopping an OA and all Autostart Processes on the Local Host.....	4-15
4.6.4	Stopping an OA and all Autostart Processes on a Specific Host.....	4-15
4.6.5	Stopping an OA.....	4-15
4.6.6	Stopping and Restarting a Host.....	4-16
4.7	Obtaining Information About Processes.....	4-16
4.7.1	Obtaining the Status of Processes.....	4-16
4.7.2	Displaying Operability Configuration Settings.....	4-17
4.8	Starting and Stopping MYNAH Software Packages.....	4-20
4.8.1	Automatically Starting the MYNAH System.....	4-20
4.8.1.1	Starting on the Solaris Platform.....	4-20
4.8.1.2	Starting on the HP-UX Platform.....	4-21
4.8.2	Automatically Starting Oracle.....	4-21
4.8.2.1	Starting on the Solaris Platform.....	4-21
4.8.2.2	Starting on the HP-UX Platform.....	4-22
4.9	User Defined Processes.....	4-22
4.10	Operability Summary.....	4-23
5.	MYNAH Administrative Tasks.....	5-1
5.1	The xmyCmd Command.....	5-1
5.2	User Enum Values.....	5-2
5.2.1	Adding User Enum Values.....	5-4
5.3	Obtaining Current System and User Statistics.....	5-5
5.3.1	Obtaining an SDs Global Parameter Values.....	5-5
5.3.2	Obtaining a User's Parameter Value Statistics.....	5-6
5.4	Concurrency Levels.....	5-7
5.4.1	Setting System Concurrency Limits.....	5-7
5.4.2	Setting Concurrency Limits for a Specific User.....	5-8
5.4.3	Setting Concurrency Limits for New Users.....	5-9
5.5	Working with Script Queues.....	5-10
5.5.1	Checking the Queues on the Local Host.....	5-10
5.5.2	Checking the Queues on all Hosts.....	5-11
5.6	Setting Queuing Priority Levels.....	5-12
5.6.1	Setting Queuing Priorities for New Users.....	5-12
5.6.2	Setting Queuing Priorities for Specific Users.....	5-13

- 5.7 Changing the Number of SEs in an SE Group.....5-14
 - 5.7.1 Increasing the Number of SEs5-14
 - 5.7.2 Decreasing the Number of SEs5-15
- 5.8 Setting Administrative Privileges5-16
 - 5.8.1 Granting Administrative Privileges5-16
 - 5.8.2 Removing Administrative Privileges5-16
- 5.9 Returning the Status of SE Groups5-17
- 5.10 Creating New MYNAH Configurations5-18
- 5.11 Removing Locks on Database Objects5-19
- 5.12 Monitoring Processes5-20
- 5.13 Monitoring the MYNAH System.....5-21
 - 5.13.1 MYNAH Processes5-21
 - 5.13.2 Monitoring the System Status5-21
 - 5.13.2.1 Using xmyOM status5-21
 - 5.13.2.2 Using vxIpcProcesses5-23
- 5.14 Monitoring MYNAH Log Files5-24
- 5.15 Deleting Message Files5-26
- 5.16 Cleaning Queues5-28
- 6. Integrating Other Testing Tools.....6-1
 - 6.1 Wrapper Script Basics.....6-1
 - 6.2 Example.....6-2
- 7. The Person Object.....7-1
 - 7.1 Creating a Person Object.....7-2
 - 7.1.1 Editing Properties Attributes.....7-2
 - 7.1.2 Editing Information Attributes.....7-6
 - 7.2 Editing an Existing Person Object7-9
- 8. Setting up Tag Name Files.....8-1
 - 8.1 Introduction8-1
 - 8.2 Tag Names8-1
 - 8.2.1 Storing Tag Name Files8-2
 - 8.2.2 Using Tag Name Files.....8-2
 - 8.3 Tag Name Table Format and the Tag Name File.....8-2
 - 8.4 Enabling Tag Name Processing8-5
 - 8.5 Testing Multiple Releases and/or Multiple Applications.....8-5
 - 8.6 Undefined Tag Name Processing.....8-5
- 9. The Screen Identification File.....9-1
 - 9.1 Why the Screen Identification File is Needed.....9-1
 - 9.2 About the Screen Identification File9-1
 - 9.2.1 Requirements for Format/Screen Names and Data Lines9-2
 - 9.2.2 Screen Identification File Formatting Commands9-3
 - 9.2.2.1 format-pattern9-3
 - 9.2.2.2 format-name-mask.....9-5

9.3	How the MYNAH System Processes the Screen Identification File	9-6
9.4	Enabling the Use of the Screen Identification File	9-8
10.	Output Processing	10-1
10.1	Process Output - Errors and Tracing	10-1
10.2	Message-Based Tracing	10-1
10.3	Activity Logging	10-1
10.4	Script Output	10-3
10.4.1	Location of the Output Files	10-3
10.4.1.1	Output Directory Symbolic Link	10-3
10.4.1.2	Determining How Many Output Directories to Retain.....	10-4
10.4.1.3	Other Possible Locations	10-4
10.4.2	Date-Specific Testing and Output Directories	10-5
10.4.3	Content of the Output Directory	10-6
10.4.4	Output file	10-8
10.4.4.1	Child Script Events	10-9
10.4.4.2	Compare Events	10-9
10.4.4.3	Exception (error) Events	10-10
10.4.4.4	Language Events.....	10-10
10.4.4.5	Script Events	10-11
10.4.4.6	Summary Events	10-12
10.4.4.7	Sutimage Events	10-13
10.4.4.8	SUT Timing (suttiming) Events	10-13
10.4.4.9	Test Object Events	10-14
10.4.4.10	User Events	10-14
10.4.5	The result File	10-15
10.4.5.1	Run Section.....	10-15
10.4.5.2	Summary Section.....	10-16
10.4.5.3	result File Example	10-16
10.4.6	Setting Output Level	10-18
10.4.6.1	Recommended Standalone Engine Output Levels	10-18
10.4.6.2	Recommended Embedded Engine Output Levels	10-18
10.4.6.3	Recommended Background Engine Output Levels.....	10-19
11.	Creating DCE Executables.....	11-1
Appendix A:	Administrative Commands	A-1
A.1	Administrative CLUI Commands	A-1
A.1.1	CLUI Command Help Messages	A-2
A.1.2	xmyCmd.....	A-3
A.1.2.1	addEnumValue	A-4
A.1.2.2	checkAllQueues	A-6
A.1.2.3	checkQueues	A-7
A.1.2.4	dfltusrconc	A-8
A.1.2.5	dfltusrpri	A-9
A.1.2.6	sedecr	A-10

A.1.2.7	seincr.....	A-11
A.1.2.8	sestat	A-12
A.1.2.9	setadm	A-13
A.1.2.10	sysconc.....	A-14
A.1.2.11	unsetadm	A-15
A.1.2.12	usrmaxconc	A-16
A.1.2.13	usrpriority	A-17
A.1.3	xmyOM.....	A-18
A.1.3.1	autostart.....	A-18
A.1.3.2	autostop.....	A-19
A.1.3.3	query	A-20
A.1.3.4	readconfig	A-21
A.1.3.5	recycle.....	A-22
A.1.3.6	shutdown.....	A-23
A.1.3.7	start/stop/status	A-24
A.2	Administrative Tcl Commands	A-25
A.2.1	xmyMsgDel.....	A-25
Appendix B:	Example Installation Files.....	B-1
B.1	Example MYNAH System Files.....	B-1
B.1.1	Example MYNAH Installation Session	B-1
B.1.2	Example System Changes File.....	B-4
B.1.3	Example MYNAH xmyProfile File	B-5
B.1.4	Example MYNAH xmyLogin File	B-9
B.1.5	Example Solaris MYNAH Start-up File (S99mynah.eg).....	B-12
B.2	Example Telexel Installation Session	B-14
B.3	Delivered Example Oracle Files	B-16
B.3.1	Example Solaris Oracle Start-up File (S96oracle).....	B-16
B.3.2	Example Oracle Configuration File	B-18
B.3.3	Example Oracle Initialization Script (initmynah5.ora).....	B-19
B.3.4	Example Oracle crdbmynah5.sql File.....	B-22
B.3.5	Example Oracle crdb2mynah5.sql File.....	B-23
B.3.6	Example Oracle crdb3mynah5.sql File	B-26
B.3.7	Example xmyCreateSequences Execution.....	B-28
B.3.8	Example xmyCreateTemplates Execution	B-29
B.3.9	Example xmyDropSequences.....	B-31
B.3.10	Example xmyCreate Tables	B-31
B.3.11	Example xmyDropTables	B-31
B.3.12	Example root.sh Run.....	B-32
Appendix C:	Building the DCE Emulated Client and Emulated Server	C-1
C.1	Template Makefile	C-1
C.2	Manual Steps.....	C-5
Appendix D:	QA Partner Configuration Options	D-1

Appendix E: Architecture	E-1
E.1 Communications	E-1
E.1.1 Platform Processes	E-1
E.1.2 Channel Names	E-1
E.2 Administrative Logging	E-2
Appendix F: Auxiliary Termino File Information	F-1
F.1 Escape Sequences Syntax	F-1
F.1.1 Terminal Capabilities	F-2
F.1.2 Key Capabilities	F-3
Glossary	Glossary-1
Index	Index-1

List of Figures

Figure 1-1.	Background Script Execution Environment	1-4
Figure 1-2.	MYNAH System 5.4 Architecture Diagram	1-7
Figure 1-3.	MYNAH System Hardware Architecture.....	1-17
Figure 1-4.	Networked Services	1-18
Figure 2-1.	Recommended Oracle Software Directory Structure	2-3
Figure 2-2.	BAIST Opening Screen	2-9
Figure 2-3.	BAIST Product Menu	2-9
Figure 3-1.	xmyConfig General Entry	3-11
Figure 3-2.	xmyConfig TermAsync Entry	3-14
Figure 3-3.	xmyConfig Term3270 Entry	3-16
Figure 3-4.	Example xmyConfig ProtocolHandler Entries	3-20
Figure 3-5.	Example xmyConfig MsgCollector Entry	3-23
Figure 3-6.	Example GuiTool_qap Entry	3-25
Figure 3-7.	QA Partner X Resources.....	3-27
Figure 3-8.	Example GuiEngine GTSE Configurations.....	3-27
Figure 3-9.	Example xmyConfig.GT File	3-29
Figure 3-10.	xmyConfig Engine Entries	3-32
Figure 3-11.	xmyConfig Engine Group Entry.....	3-37
Figure 3-12.	xmyConfig Dispatcher Entry	3-38
Figure 3-13.	Example xmyConfig File.....	3-40
Figure 3-14.	xmyConfig Process Entry Structure	3-44
Figure 3-15.	Example xmyConfigOP Process Entry.....	3-45
Figure 3-16.	xmyConfigOP OperabilityAgent Entry Structure	3-50
Figure 3-17.	Example xmyConfigOP OperabilityAgent Entry	3-50
Figure 3-18.	Example MYNAH xmyConfigOP File	3-53
Figure 4-1.	Operability Architecture	4-4
Figure 4-2.	Operability Feature Summary	4-23
Figure 5-1.	SUT Object's Type Drop-down Menu User Enum Values	5-2
Figure 7-1.	Creating a New Person Object.....	7-2
Figure 7-2.	New Person Object Properties View	7-3
Figure 7-3.	Edited Person Object Properties View	7-4
Figure 7-4.	Duplicate UNIX ID Error Box	7-5
Figure 7-5.	Close Dialog Box.....	7-5
Figure 7-6.	Person Object Information View	7-6
Figure 7-7.	Database Browser - Keyword Object	7-7
Figure 7-8.	Properties View with Keyword Box in "Focus"	7-8
Figure 7-9.	Properties View with Keyword Added.....	7-9
Figure 7-10.	Database Browser - List of Person Objects	7-10

Figure 7-11. Edited Person Object — Granting Administrative Privileges	7-11
Figure 7-12. Refreshed Database Browser	7-12
Figure 8-1. Example Tag Name File	8-4
Figure 9-1. Example Screen Identification File	9-7

List of Tables

Table 1.	MYNAH Administration Guide Sections.....	Preface-1
Table 1-1.	Required Software Packages	1-9
Table 1-2.	TN3270 Client Packages (Solaris and HP-UX)	1-9
Table 1-3.	PRT3270 Emulation Products	1-10
Table 1-4.	Solaris DCE Libraries.....	1-11
Table 1-5.	HP-UX DCE Libraries.....	1-11
Table 1-6.	GUI Testing Tools	1-11
Table 1-7.	Compatible PC/UNIX QA-Partner Versions.....	1-12
Table 1-8.	TOPCOM Requirements	1-12
Table 1-9.	FTP Requirements	1-12
Table 1-10.	Recommended Font Packages	1-13
Table 1-11.	Optional Encryption Kit	1-13
Table 2-1.	Installation Steps.....	2-1
Table 2-2.	Licensed Features	2-6
Table 2-3.	Oracle Installation Items and Responses	2-26
Table 2-4.	Oracle Software Asset Manager Packages	2-27
Table 4-1.	xmyOM Sub-commands.....	4-5
Table 5-1.	Delivered User Enum Values	5-2
Table 5-2.	Required Telexel Processes	5-20
Table 5-3.	vxFilterLogFile Field Specifier IDs	5-24
Table 9-1.	Example format-pattern Regular Expressions	9-4
Table 11-1.	DCE C++ Compilers	11-1
Table C-1.	DCE Makefile variables	C-1
Table 11-2.	Escape Sequences	F-1
Table 11-3.	Terminal Capabilities	F-2
Table 11-4.	Key Capabilities	F-3

Preface

Document Structure

Table 1 list the sections in this document with a brief description of each section.

Table 1. MYNAH Administration Guide Sections (Sheet 1 of 2)

Section Number	Section Name	Description
Section 1	Introduction	This section contains general information on this guide including an overview of some basic MYNAH.
Section 2	Installation	This section contains information on installation, including required third-party software. This section also contains information on configuring a minimal working installation.
Section 3	Customizing a MYNAH Configuration	This section describes the processes for customizing MYNAH System configuration.
Section 4	Operability Management—Starting and Stopping MYNAH Processes	This section describes the procedures used to start, stop, and get status of all of the MYNAH processes.
Section 5	MYNAH Administrative Tasks	This section describes the tasks used to administer the MYNAH System.
Section 6	Integrating Other Testing Tools	This section provides information on integrating third-party (or home-grown) testing tools other than the specified set of third party tools.
Section 7	The Person Object	This section provides information on the administrative use of the GUI's Person Object.
Section 8	Setting up Tag Name Files	This section contains information on creating Tag Name Files, which are used for processing locations on a 3270 (synchronous) terminal.
Section 9	The Screen Identification File	This section contains information on creating a Screen Identification File, which the MYNAH GUI uses to identify screens when emulating a 3270 (synchronous) terminal.
Section 10	Output Processing	This section describes the structure and location of the log and trace files.

Table 1. MYNAH Administration Guide Sections (Sheet 2 of 2)

Section Number	Section Name	Description
Section 11	Creating DCE Executables	This section contains the instructions on how to create the emulated client or server executables for the DCE Package.
Appendix A	Administrative Commands	This appendix describes the CLUI and TCL commands used to perform administrative processes.
Appendix B	Example Installation Files	This appendix contains examples of the delivered files used to install the MYNAH System.
Appendix C	Building the DCE Emulated Client and Emulated Server	This appendix describes the steps required to manually build the DCE emulated client and the emulated server executables using the tools developed by the MYNAH System if you do not use the program described in Section 11 .
Appendix D	QA Partner Configuration Options	This appendix lists available QA Partner configuration options
Appendix E	Architecture	This appendix contains some background information on the MYNAH System architecture.

Related Documents

- *BR 007-252-002, MYNAH System Users Guide*
- *BR 007-252-004, MYNAH System Scripting Guide*

On-line Versions of the MYNAH Documents

This document is available on-line in the Adobe[®] Acrobat[®] PDF format. The PDF file is accessible from either the local file system or from an internal URL, depending on where you install the PDF files.

Viewing the PDF file requires that you have installed the Adobe Acrobat Reader[®]. See [Section 2.13](#) for information installing the PDF files and obtaining the Acrobat Reader. In addition, you can download the Acrobat Reader directly (off the Internet) from Adobe at www.adobe.com.

Formatting Conventions

The following formatting conventions are used in this document.

Textual Conventions

In the body of this document, the following conventions are used

- Literal strings (command, subcommand, and option names) appear in **bold Times**. For example, we will talk about the **xmyOM** command and the **start** subcommand.
- Substitutable or user supplied items (e.g., option and argument values) appear in *boldItalics Times*.
- File and directory names appear in *italics Times*.
- The first time a term is used, it will appear in **bold Helvetica**, as when we first mention **Script Engines**.
- When we explain the syntax for a MYNAH administrative Command Line User Interface (CLUI) subcommand, the syntax will appear in the form:

```
xmyCommand_name subcommand_name -option_name option_value\  
?-option_name option_value? argument
```

where:

- The CLUI command, subcommand, and option names appear in the *Courier* font.
- All substitutable or user-supplied items(e.g., option and argument values) appear in *Courier Italics*.
- All optional entries are delimited by question marks.

For example

```
xmyOM status ?-o oa_name? logical_process_name
```


CLUI Command Conventions

Appendix A.1 contains descriptions of each of the administrative CLUI commands and subcommands. Each CLUI subcommand entry contains sections detailing the syntax for, and containing descriptions of, the subcommand. These entries use the following formatting conventions:

Syntax

```
xmyCommand_name subcommand_name -option_name option_value\n      ?-option_name option_value? argument
```

The syntax section uses the following formatting conventions:

- CLUI command, subcommand, and option names appear in the *Courier* font.
- All substitutable or user-supplied items (e.g., option and argument values) appear in *Courier Italics*.
- All optional entries are delimited by question marks.

Description

The description section explains the reasons and uses of the subcommand; this section uses the following conventions:

- Literal strings (class command, subcommand, and option names) appear in **bold Times**.
- Substitutable or user supplied items (e.g., option and argument values) appear in ***boldItalics Times***.
- File and directory names appear in *italic Times*.

Example

When applicable, examples of the use of each subcommand are included. The examples use the following conventions:

- All examples appear in *Courier*
- Return values in the examples are in *Courier*.

1. Introduction

This document describes the procedures for installing and administering MYNAH™ System Release 5.4.

1.1 MYNAH System Overview

MYNAH System Release 5.4 supports the following general features:

- The ability to represent user-defined test cases, provide metrics on those test cases and use those defined test cases as a basis for test plans
- The ability to automate both test interactions with a **System Under Test** (SUT) and tasks on the following interfaces:
 - Application-to-Application contract interface
 - DCE/IDL client-server interface
 - Synchronous printer interface
 - Synchronous 3270 terminal interface
 - Asynchronous terminal interface
 - Batch MVS jobs
 - X Window System Graphical User Interface (GUI) through integration of vendor tools
 - Microsoft® Windows® GUI through integration of vendor tools.

Automation is accomplished by “scripting” the interactions with the SUT.

The MYNAH System scripting Language is **Tcl**, pronounced “tickle,” which stands for tool command language. To the basic set of Tcl commands, we have added **Extensions**, which are commands and procedures that expand Tcl’s capabilities. In addition, the MYNAH scripting language also supports extended Tcl (TclX) language. For information on the MYNAH scripting language, see the *MYNAH System Scripting Guide*.

1.2 Architecture Overview

The primary processes in the MYNAH System are the following:

1.2.1 Script Engine (SE)

A **Script Engine** (SE) is an extended Tcl interpreter that runs user scripts. Scripts can contain any core Tcl commands as well as any commands implemented in the set of MYNAH Tcl extension packages.

An SE must be running as part of an **SE Group** under the control of a Script Dispatcher (SD). The managed SEs are organized into SE Groups.

You can also configure a special SE called an SE Lite, which does not require a database. If your installation is configured without a database, all SEs are automatically SE Lites. If you do use a database but you want some SEs to be SE Lites, you can configure one or more SE Groups to use SE Lites.

1.2.2 Script Dispatcher (SD)

Script Dispatchers manage some number of SEs. An SD's primary function is to "dispatch" script execution requests to an up and running SE. Part of the dispatching job is to manage **concurrency** of script execution, which is the ability to synchronize between concurrently executing scripts. All script requests that come to an SD process are managed as part of one concurrency group. There can be more than one SD in a MYNAH configuration.

1.2.3 Trigger Daemon (TD)

The **Trigger Daemon** (TD) sends data from the database to the MYNAH processes. When an update is made to the MYNAH database, the TD broadcasts the update to such processes as the GUI and SDs.

1.2.4 Boot Daemon (BD)

The **Boot Daemon** (BD) "manages" the SEs running on its machine: it starts the SEs and it knows when an SE dies. A BD runs on each machine where a MYNAH process runs.

If the SE dies abnormally, the SE's exit code tells the BD why the termination occurred. If the SE died due to certain types of errors, the BD re-starts it.

1.2.5 Graphical User Interface (GUI)

The Graphical User Interface (GUI) provides user access to all MYNAH functionality, including:

- Interactive script execution through the use of an Embedded Script Engine (ESS)
- Test management functions, such as scheduling tests on several releases of an application.

See the *MYNAH System Users Guide*, for information on using the MYNAH GUI.

1.2.6 Command Line User Interface (CLUI)

The Command Line User Interface (CLUI) provides access to a subset of MYNAH functionality via a series of commands that are execute from the UNIX prompt. These functions include

- Starting and stopping MYNAH processes.
- Letting users send requests to the SD to execute scripts in the background.
- Letting the MYNAH Administrator send requests to the SD to change administrator-tunable parameters in the SD (such as the overall system concurrency level or the number of SEs in an SE group).
- Letting the MYNAH Administrator create or update database entities, such as User Enum Values, and user entries.
- Giving users a facility to encrypt a **des** key.

1.2.7 Application-to-Application Main Collector Processes (CL)

The Application-to-Application Main Collector Processes (CL) stores all incoming Application-To-Application messages into a special area (the **Message Response Directory**) on disk.

1.2.8 Background Execution Environment (BEE)

While not itself an actual MYNAH Process, the **Background Execution Environment (BEE)** is a powerful part of the MYNAH System in that it lets you execute scripts as background processes. That is, you can submit the script to the MYNAH System, which will schedule the script for execution.

The combination of SDs and SE Groups constitutes the MYNAH BEE. The relationship between SDs and SE Groups is depicted in [Figure 1-1](#).

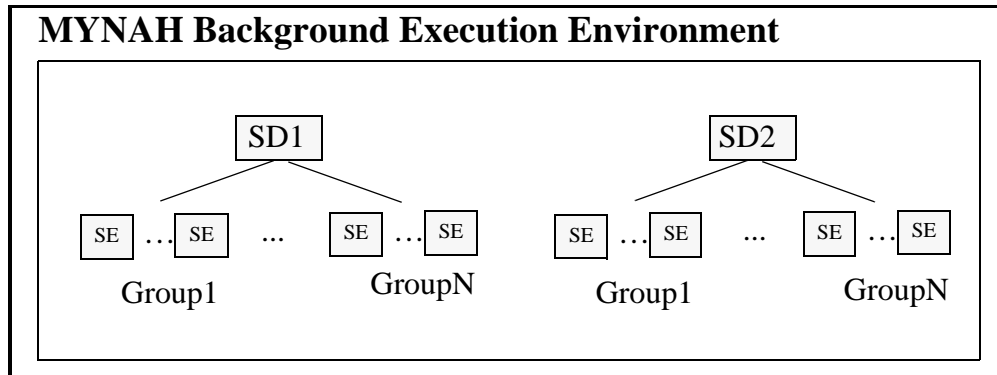


Figure 1-1. Background Script Execution Environment

A simple BEE is composed of one SD and a single SE Group.

To understand why background execution is desirable, why multiple SE groups are desirable, and why multiple SDs may be needed, the following sub-sections describe a MYNAH new-user scenario that occurs over a period of time.

1.2.8.1 Why Background Execution

When you first start to use the MYNAH System you will most likely use the interactive embedded SE to create and run scripts. For awhile, this is the only type of execution that will be needed. (If you only have one or two scripts, then it is no problem to run those scripts in the foreground interactively).

However, you will quickly begin to accumulate more and more automated scripts. Lets say you now have 30 scripts. At this point you may wonder if there's a way of submitting these scripts for unattended execution. The answer is yes. The automated scripts may be submitted to the BEE.

NOTE — For this example, let's assume your installation has one SD that is configured to use one SE Group that contains three SEs.

Let's assume that you submit the 30 scripts to the BEE. The SD will schedule the first three scripts on the three available SEs and queue the remaining 27 requests. When one script finishes and an SE becomes available, the SD will "dispatch" another script to that now non-busy SE. In this way the SD, in combination with the SEs, provides concurrency control on the number of executing scripts.

You may ask why not use 30 SEs to run the 30 scripts. There are two reasons why this may not be feasible. First of all, the SUT may not be able to handle 30 concurrent scripts accessing it. Secondly, the hardware where MYNAH is running may not be able to handle 30 concurrently running SEs. Information about these two limitations will yield the ideal number of SEs that can/should be in the default SE group.

1.2.8.2 Why SE Groups

All of the 30 scripts have to log into the application that is being accessed. It might be more efficient if the SEs remain logged into the application so that each new script begins from the main screen. Can this be done? The answer is yes. You would need to use the **Connection Only** mode for the SEs that are in the SE Group. This is easily done by changing the value of the **Mode** parameter for the SE in the *xmyConfig* file to **ConnOnly**.

So now you have even more efficient execution of the 30 scripts in the BEE. However, up to this point there is still no need for multiple SE groups—that is, until now.

If the scope of your duties expand to include validating data in a second application, this is certainly not a problem during the creation of the scripts phase. However, these new scripts log into a different application than the first 30, so you now need a new SE group. The new group will be used to maintain connections to the new application, therefore this new group will also be run in the **Connection Only** mode. Your MYNAH Administrator will configure this new group and give it a name. Let's call it **SEgroupApp2**. The MYNAH Administrator will probably at this time also create an SE group for the original application. Let's call that SE group **SEgroupApp1**. Note that the original SE group will remain as well.

If your installation uses a database, the MYNAH System database contains **Script Objects**, which are used to document script code. For this new task, the Script Objects for the original 30 scripts will be updated to indicate that they should run on **SEgroupApp1**. (See the *MYNAH System Users Guide*, for information on using Script Objects.)

Let's assume you now create 20 scripts that interact with the new application. For these 20 scripts, you will indicate the fact (in the Script Objects) that they should run on **SEgroupApp2**. You now have 50 different scripts. Can you submit all 50 scripts, or any subset of them, to the BEE for execution? Yes.

The SD will handle dispatching the scripts to the correct SE group and will keep all SEs in both groups busy.

1.2.8.3 Why Multiple SDs

A single SD provides concurrency control for all scripts that are submitted to it, one criteria for concurrency levels is the ability of the SUT to handle that concurrency. Well, you may need to submit a large number of scripts that interact with one SUT and another set of scripts that interact with a different SUT, and you may want to do both of these things at the same time. Since concurrency must be controlled for both SUTs, two SDs can be used to handle this situation.

1.2.9 MYNAH Architecture Abstract View

Figure 1-2 depicts an abstract view of the architecture of the MYNAH System.

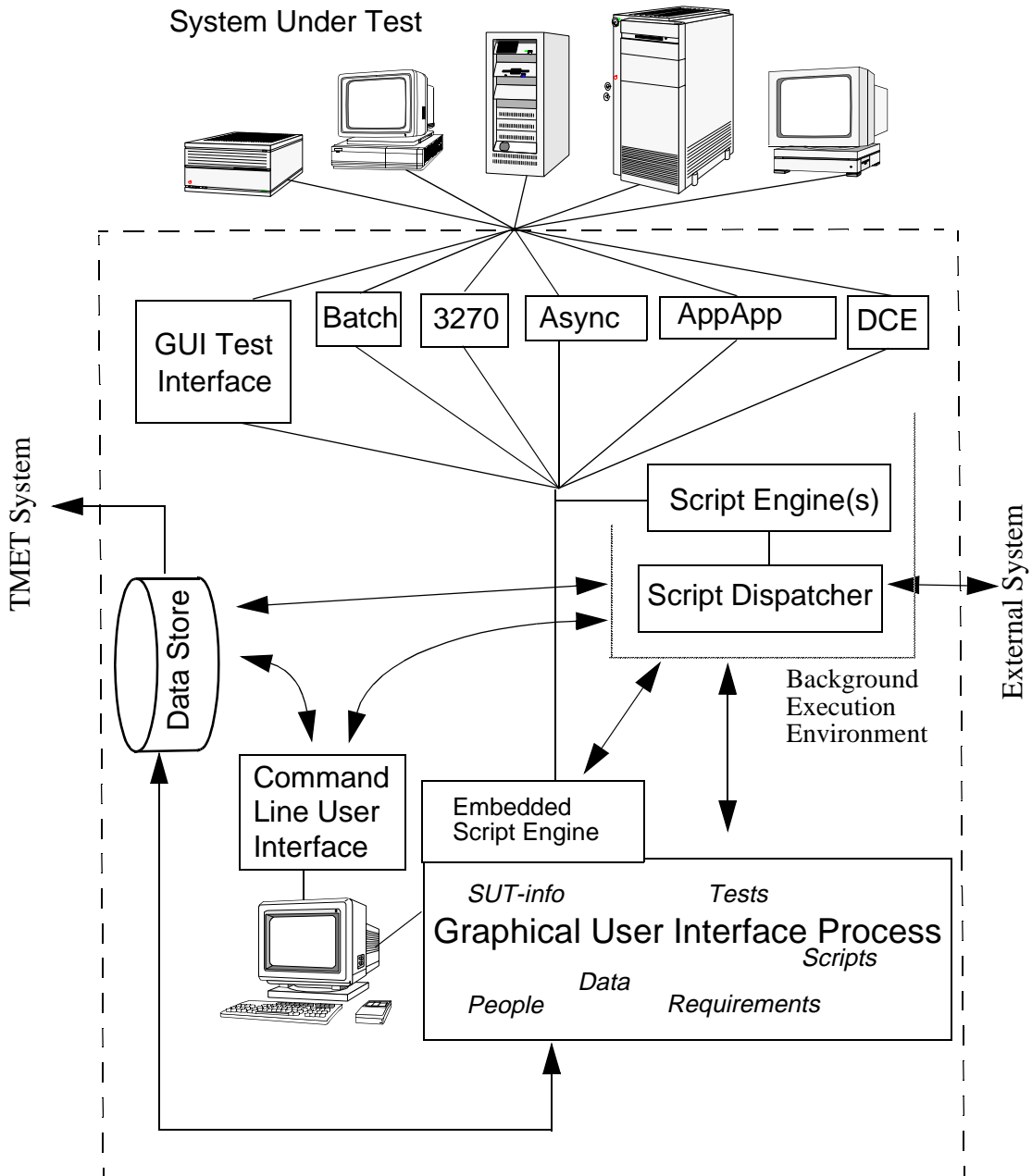


Figure 1-2. MYNAH System 5.4 Architecture Diagram

From the GUI, the user can interactively create or run automation scripts by using an embedded interactive SE.

The user can also use the GUI to request scripts to be run noninteractively (i.e., in the background) by using the SD and associated SEs. The SD allows you to group SEs to be dedicated to a specific purpose.

Using the GUI the user may create, or modify, any data objects supported by the system. Data objects are persistent testing artifacts that are stored in the database. Some examples are user-defined tests, scripts, test data, requirements information, and SUT information.

The CLUI provides commands that can be invoked from the UNIX[®] prompt that support script execution in the BEE and the monitoring of that execution.

NOTE — In addition, the system supports nontesting use. The nontester will be able to create, save, and execute scripts.

1.3 System Requirements

MYNAH System Release 5.4 requires a SPARC machine running the Solaris (Release 2.6) operating system or an HP machine running the HP-UX (Release 11.0) operating system. Test management functions require an Oracle[®] database.

NOTE — If you are running the Oracle software on the same machine (Solaris or HP-UX) on which you are running the MYNAH System, then operating system patches may also be needed. Refer to the Oracle7 Server Installation Guide for your operating system.

The MYNAH System requires 1.6 GB of disk space.

1.4 Third Party Software Requirements

In order to use MYNAH 5.4, a number of software packages must first be installed. The set of packages needed depends upon how the MYNAH System will be used. A minimal MYNAH configuration requires the software systems shown in [Table 1-1](#).

Table 1-1. Required Software Packages

Product	Version	Vendor
Oracle DBMS	8.0.5	Oracle (optional)
TELEXEL IPC ^a	7.3	Telcordia
X Window System™	X11R6	Sun or HP

a. TELEXEL IPC is provided with the MYNAH 5.4 distribution. Users must install it separately.

NOTE — The Oracle database is required when you need to perform test management functions. If you want to run the MYNAH System for task automation only, you do not need to install the Oracle database, and you will only be able to run the GUI Script Builder via the **xmyRunMynah -b** command.

WARNING — SQL *should not* be used to manipulate the MYNAH database.

1.4.1 3270 Terminal Package

Accessing synchronous systems through the TN3270 protocol or BISYNC/SNA over TCP/IP requires a TN3270 client. The MYNAH 5.4 System uses the IBM™ specification for 3270 access, High Level Language Application Program Interface (or HLLAPI). A vendor's HLLAPI API works in concert with a vendor's TN3270 client to facilitate automated interactions with the 3270 host.

If you plan to perform 3270 scripting you must install the vendor TN3270 client and HLLAPI API shown in [Table 1-2](#).

Table 1-2. TN3270 Client Packages (Solaris and HP-UX)

Product	Version	Vendor
X-Direct TN3270 (Solaris)	9.1.7	I/O Concepts

NOTE — If you want to use Version 8.4 of the X-Direct TN3270 software, please contact the MYNAH Customer Service Center. See [Section 1.10](#) for information on contacting the service center.

The MYNAH 5.4 3270 domain is driven primarily through the TN3270 protocol. For MYNAH users who do not have a host that supports TN3270 directly but do have a Sunlink™ BSC or SNA gateway installed, the Sunlink TN*Server can be used in conjunction with their existing SunLink gateways. Through these servers, MYNAH 5.4 users will be able to access the host through the TN3270 protocol. They must have the TN*Server piece installed with their SunLink package, as well as some patches for both the gateways and the TN*Server piece.

NOTE — The TN3270 package mentioned above is still required for Sunlink TN*Server.

1.4.2 3270 Printer Package

The MYNAH 5.4 3270 printer package (Prt3270) is driven by the print emulation functionality provided by the TN3270E protocol. As a result, MYNAH 5.4 3270 printer domain users must have access to a TN3270E server to complete 3270 printer domain functionality. Using the 3270 printer domain requires both a TN3270E client and a TN3270E server. Alternately, if you have Sunlink 3270 BSC or SNA Gateway software, you can use the 3257 module.

Use one of the IBM 3270 emulation products listed in [Table 1-3](#).

Table 1-3. PRT3270 Emulation Products

Product	Version	Vendor
SunLink SNA 3270	8.0 or 9.0	Sun Microsystems
SunLink BSC 3270	8.0	Sun Microsystems
X-Direct TN3270E printer client	1.69	I/O Concepts

1.4.3 Asynchronous Terminal Package

No software packages are required.

1.4.4 DCE Package

The MYNAH DCE test environment requires the use of DCE run time libraries and software shown in Tables 1-4 (Solaris) and 1-5 (HP-UX).

Table 1-4. Solaris DCE Libraries

Product	Version	Vendor
DCE	2.0	Transarc®
Tools.h++ header files and libraries	7.0.9	RogueWave
SparcWorks C++ Compiler	4.2	Sun Microsystems

Table 1-5. HP-UX DCE Libraries

Product	Version	Vendor
DCE	1.7	HP
aC++	A.03.10	HP

NOTE — While the MYNAH DCE test environment was developed using Transarc's DCE version 1.1 development and runtime libraries, the DCE domain does not rely on Transarc's implementation specifically; it will work with any implementation of the DCE specification.

NOTE — While other compilers may work, the DCE component was tested using the SparcWorks compiler.

You must also have access to a version of the **make** program. Both **gnumake** and the Solaris **make** will work.

1.4.5 GUI Package

To test X11 GUIs, install one of the GUI test tools shown in Table 1-6.

Table 1-6. GUI Testing Tools

Product	Version	Vendor
QA Partner™	4.0.1 Distributed	Segue

You can test Windows GUIs (Windows 3.x, Windows NT[®], or Windows95) using the Distributed edition of the Segue QA-Partner™ software. This works by running the QA-Partner software on the UNIX workstation and having it communicate with an agent on the Windows platform.

To do this, you must install the QA-Partner software on the PC and a compatible version of the QA-Partner software on the UNIX workstation, as detailed in [Table 1-7](#).

Table 1-7. Compatible PC/UNIX QA-Partner Versions

Product	Windows Version	UNIX Version	Vendor
QA-Partner	4.0	4.0.1	Segue

1.4.6 AppApp Package

The Application-to-Application Package (TOP) requires either a Customer developed Application specific Interface Collector or Telcordia's TOPCOM program shown in [Table 1-8](#).

Table 1-8. TOPCOM Requirements

Product	Version	Vendor
BAE TOPCOM	5.4.1	Telcordia

NOTE — The appropriate version of BAE TOPCOM is provided with the MYNAH 5.4 distribution if purchased. Install it separately.

NOTE — The General Application-to-Application (AppApp) does not require TOPCOM. It is totally independent of TOP or TOPCOM.

1.4.7 Batch Package

Batch testing requires an FTP server running on the mainframe, as shown in [Table 1-9](#).

Table 1-9. FTP Requirements

Product	Version	Vendor
FTP	OS/390 2.5	IBM

1.4.8 Miscellaneous

The MYNAH System has been tested using the font packages listed in [Table 1-10](#).

Table 1-10. Recommended Font Packages

Package	Description
SUNWi1of	ISO-8859-1 (Latin-1) Optional Fonts
SUNWi2of	X11 ISO-8859-2 optional fonts
SUNWi2rf	X11 ISO-8859-2 required fonts
SUNWi4of	X11 ISO-8859-4 optional fonts
SUNWi4rf	X11 ISO-8859-4 required fonts
SUNWi5of	X11 ISO-8859-5 optional fonts
SUNWi5rf	X11 ISO-8859-5 required fonts
SUNWi7of	X11 ISO-8859-7 optional fonts
SUNWi7rf	X11 ISO-8859-7 required fonts
SUNWi9of	X11 ISO-8859-9 optional fonts
SUNWi9rf	X11 ISO-8859-9 required fonts
SUNWxglft	XGL Stroke Fonts
SUNWxwft	X Window System common (not required) fonts
SUNWxwfa	X Window System Font Administrator
SUNWxwfnt	X Window System platform required fonts
SUNWxwfs	Font server
SUNWxwoft	X Window System optional fonts

United States Solaris users may optionally use the encryption kit shown in [Table 1-11](#) with MYNAH 5.4.

Table 1-11. Optional Encryption Kit

Product	Version	Vendor
U.S. DES Encryption Kit	11.6.0	Sun

If the U.S. DES Encryption Kit is installed, the directory path to where the software is installed must be added to the MYNAH System's shell PATH so that MYNAH scripts can access **des**.

1.5 Script Engine (SE)

The SEs come in three varieties: Background, Embedded, and Command-line.

In addition, SEs run in one of three modes: *Stateless*, *ConnOnly*, and *FullState*. The mode cannot be changed after start-up.

1.5.1 SE Types

The MYNAH System contains the following SE types.

1.5.1.1 Background

Background SEs are used when script is submitted using the CLUI's **xmyCmd submit** subcommand. (See the *MYNAH System Users Guide*, for information on **xmyCmd submit**.) Background SEs are processes that do not have a GUI and do not offer graphical displays of script execution. Background SEs are faster than the GUI and are intended to be used to process large numbers of unattended scripts.

1.5.1.2 Embedded

Embedded SEs are started from within the GUI, and these SEs are part of the GUI's Script Builder. (See the *MYNAH System Users Guide* for information on the Script Builder.)

1.5.1.3 Command-line

Command-line SEs are started with the **xmytclsh** tool. A Command-line SE accepts Tcl commands from **stdin** and produces results on **stdout**. It does not interface with the MYNAH System database.

1.5.2 SE Execution Modes

MYNAH System SEs run in one of the following modes.

1.5.2.1 Stateless Mode

In the *Stateless* mode, when a script completes

- All open connections are closed.
- All Tcl file descriptors are closed.
- The Tcl interpreter is deleted.

The Tcl start-up script is rerun each time a new interpreter is created.

1.5.2.2 ConnOnly Mode

In the *ConnOnly* mode, when a script completes

- All Tcl file descriptors are closed.
- The Tcl interpreter is deleted.
- Connections to the SUT are maintained.

The *ConnOnly* mode is used when users want to maintain connections between scripts.

1.5.2.3 FullState Mode

In the *FullState* mode, when a script completes

- Connections to the SUT are maintained.
- All Tcl file descriptors remain open.
- The Tcl interpreter remains open.

In the *FullState* mode, all scripts submitted to an SE Group go to the first available engine in that group. No reinitialization is performed between each script execution.

When users submit scripts to an SE group running in *FullState* mode, they should ensure that the Tcl interpreter in each SE in a *FullState* Group is in the same state (e.g., same variables defined). If not, scripts submitted to such a group may behave differently depending on which SE they run in.

NOTE — Standalone SEs (i.e., those submitted using **xmytclsh**) are always in *FullState* mode regardless of the *xmyConfig* file setting, since they run individual statements, not entire scripts.

1.6 Administrative Commands Overview

The MYNAH System provides CLUI and Tcl based commands that let you perform various administrative functions.

1.6.1 Administrative CLUI Commands

The administrative CLUI commands let you

- Perform **Operability** tasks (start, shutdown, and obtain status of MYNAH processes), and get information on the set of processes under Operability control.
- Send requests to the SD to change administrator-tunable parameters in the SD (such as the overall system concurrency level or the number of SEs in an SE group).

The administrator uses the CLUI's **xmyCmd** and **xmyOM** commands. In actuality, these are the names of families of commands, each with control over specific areas or categories of MYNAH functionality. Each CLUI command has a series of subcommands that are analogous to the MYNAH extension **methods**. (See Section 1.2 of the *MYNAH System Scripting Guide*, for an overview of the MYNAH extensions.) These subcommands perform the actual command activities, accomplishing the requested operation, and these must be used in conjunction with a CLUI command.

These commands use the following syntax:

```
xmyCommand_name subcommand args
```

where *xmyCommand_name* is either **xmyCmd** or **xmyOM**.

For example, to display Operability Management configuration information, you can type

```
xmyOM query
```

NOTE — See [Section 4](#) for information on Operability Management.

Complete explanations of the administrative CLUI commands can be found in [Appendix A.1](#).

1.6.2 Administrative Tcl Commands

The administrative Tcl commands let you perform administrative tasks for a specific testing package (e.g., Async or 3270). These commands are in fact MYNAH extension methods that must be executed using a Tcl interpreter, such as **xmytclsh** or the Script Builder. In addition, you can include these commands in scripts, which can be executed using the **xmyCmd submit** command.

1.7 Hardware and Operating System Requirements

Figure 1-3 shows a distributed hardware architecture for MYNAH 5.4. It contains a server and multiple workstations for running the MYNAH System and an ethernet network for communications between the MYNAH hosts and the SUT. The SUT may consist of any combination of hardware devices.

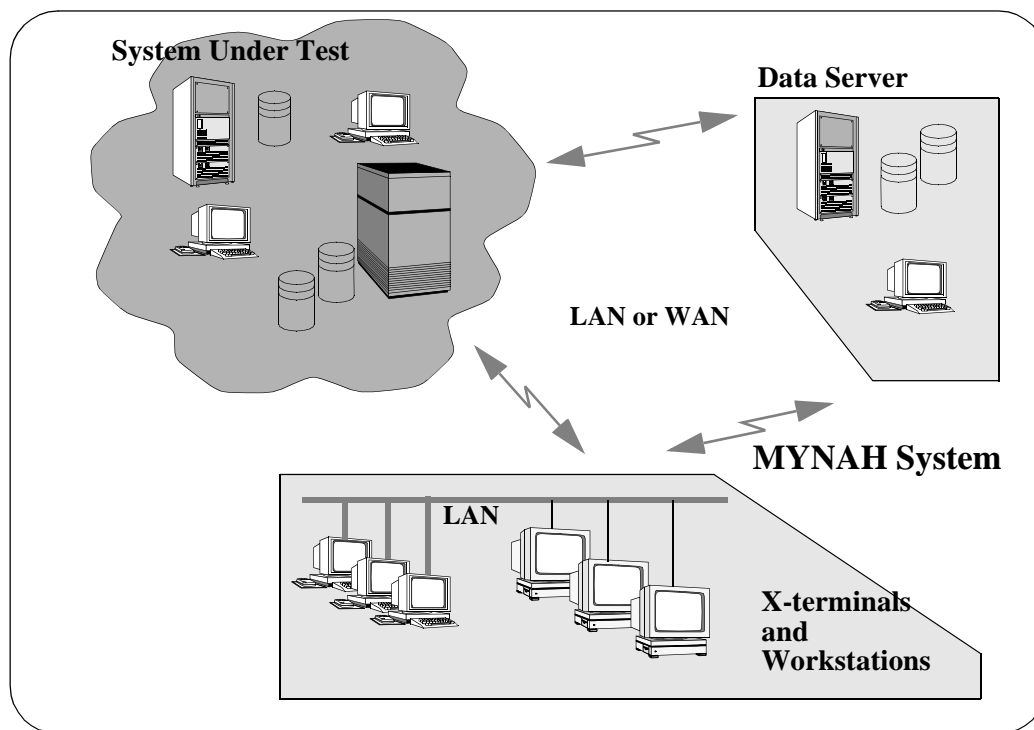


Figure 1-3. MYNAH System Hardware Architecture

The end-user display devices (X-terminals, workstations running X, or PCs running an X emulator) provide graphical/windowing capabilities and a UNIX command-line interface to MYNAH functionality. The display device should support a minimum resolution of 1024 by 768 pixels.

If the display device is a workstation, it may also be running a background MYNAH component.

For MYNAH Release 5.4, any host in the configuration that is running a MYNAH component must be a SPARC™ machine running the Solaris™ operating system (Release 2.5.1) or an HP™ machine running the HP-UX™ (Release 10.20) operating system.

The MYNAH System may be installed and run on a single SPARC2™ or HP workstation with 128 Mb of memory with the following configuration:

- One (1) MYNAH GUI
- One (1) Script Dispatcher
- Three (3) Script Engines
- All Oracle processes (required if you need to perform test management functions).

This is a minimal configuration and is typically for single user, low volume usage. The typical installation involves multiple users and multiple hosts.

1.8 Networking

MYNAH processes can be distributed across multiple hosts. All MYNAH processes use the network services of the Database Management System for database access and the network services of Telexel IPC processes for MYNAH interprocess communications and logging, as shown in [Figure 1-4](#).

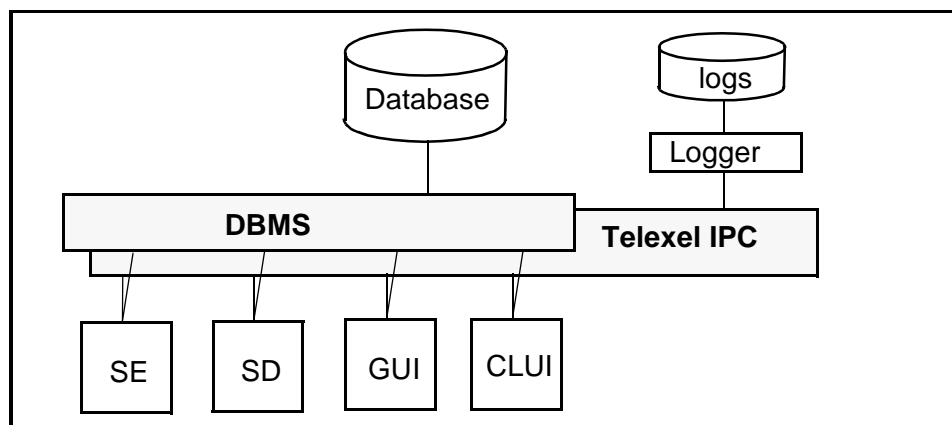


Figure 1-4. Networked Services

All MYNAH process executables reside in a single location designated as \$XMYDIR, but configuration information can reside in a different location, designated as \$XMYHOME; each user can have a different \$XMYHOME. The file system where this information is

stored must be accessible (typically via NFS) by every host machine that runs a MYNAH process.

1.9 On-line Documentation

This document is available on-line in the Adobe[®] Acrobat[®] PDF format. Viewing the PDF file requires that you have installed the Adobe Acrobat Reader[®].

NOTE — See [Section 2.13](#) for information on installing the on-line documents and obtaining the Acrobat Reader.

Once you have installed the Acrobat Reader, you can read the file

- Using the Acrobat Reader directly if the MYNAH System has been installed on a local system.
- Using the Acrobat Reader as plug-in to a browser if the MYNAH System has not been installed on a local system, such as if the system has been installed on a UNIX Solaris server and you are using an X-windows emulator to access the system on a PC. Consult your browser's documentation for information on how to install plug-ins.

If you access the PDF file via a browser, you may wish to download the file to your local system, which will give you direct access to the file the next time you need to read the file, rather than waiting for the browser to load it.

1.10 Customer Support

You can get support from the MYNAH Customer Service Center between 8:00 AM and 7:00 PM ET Monday through Friday by calling (732) 699-2668, Option 3; if outside the 732 area, call (800) 795-3119, Option 3.

During non-standard support hours and holidays, critical problem coverage is provided.

You can also contact support (for non critical problems) via e-mail at mynah-support@telcordia.com.

2. Installation

This section discusses the steps for installing the MYNAH System Release 5.4 and related software packages, e.g., Telexel.

2.1 Introduction

For a minimal installation (i.e., if you want to use the MYNAH System for task automation only), of the MYNAH System you must also install the Telexel software package, which the MYNAH System uses for interprocess communications and logging. If you also want to use the MYNAH test management capabilities, you must install an Oracle database.

The following subsections explain the steps you need to install the MYNAH System for both task automation and test management. If you do not need test management, you can skip [Section 2.6](#) on installing the Oracle software. In addition, we include example installations of I/O Concepts' X-Direct TN3270 (for 3270 scripting), TOPCOM, and the third-party GUI testing tool software packages. These example installations can serve as guidelines for installing other third-party packages.

[Table 2-1](#) lists, in order, the steps needed to create our example MYNAH installation.

Table 2-1. Installation Steps

Step	Section
Installing the MYNAH System	Section 2.4, Page 2-11
Installing the Telexel System	Section 2.5, Page 2-14
Installing Oracle (Optional)	Section 2.6, Page 2-20
Installing I/O Concepts' X-Direct TN3270 (Optional)	Section 2.7, Page 2-36
Installing TOPCOM (Optional)	Section 2.8, Page 2-39
Installing Third-Party GUI-Test Tools — QA/Partner (Optional)	Section 2.9, Page 2-42
Configuring a Minimal Working Installation	Section 2.11, Page 2-44
Verifying the MYNAH System	Section 2.12, Page 2-46

2.2 Preliminary Background Information

There is background information you need to know before you install the MYNAH System.

2.2.1 Assumptions and Recommendations

The following subsections make certain assumptions about your installation. If you do not follow these assumptions, remember to make the necessary changes while installing the software.

WARNING — Do not create any directories or links at this time.

1. All packages are installed in */opt*.
2. Under this directory you should create a directory named *XXXXxxx*, where *XXXX* is *SUNW* or *HPUX*, depending on your platform and *xxx* is the first three letters of the package, e.g., *SUNWora* for the Oracle software and *SUNWmyn* for the MYNAH System on a Sun platform or *HPUXora* for the Oracle software and *HPUXmyn* for the MYNAH System on an HP platform.
3. Under each package's *XXXXxxx* directory you create a directory for the version number, e.g., 8.0.5 for the Oracle software.
4. Create a symbolic link in each package's *SUNWxxx* or *HPUXxxx* directory pointing to the version directory, e.g., for the Oracle software you would execute

```
ln -s /opt/SUNWora/8.0.5/app/oracle/product/8.0.5 oracle
```

or

```
ln -s /opt/HPUXora/8.0.5/app/oracle/product/8.0.5 oracle
```

in the Oracle directory (*/opt/SUNWora* or */opt/HPUXora*).

You are creating symbolic links so that you have to change only the link to the new version directory when you install a new version of the software. All user references should be made to the symbolic link name. This will make software upgrades easy since no library names will have to be changed if the users are referencing the symbolic links.

NOTE — For the MYNAH, Telexel, and TOPCOM (optional) products, Items 3 and 4 will be done automatically by the BAIST installation product. For Oracle and any other products you must do this yourself.

Figure 2-1 shows an example of the recommended Oracle directory structure. When configuring the MYNAH System, there are several places where you must specify the full path for the Oracle software, i.e., `/opt/XXXXora/oracle`. If you later install a new version of the Oracle software (e.g., Version 8.0) and you wish to retain the old version of the Oracle software, you will have to change only the symbolic link definition for `/opt/XXXXora/oracle`. However, if you used `/opt/XXXXora/8.0.5` to specify the path for the Oracle software, you must change each occurrence of this path declaration.

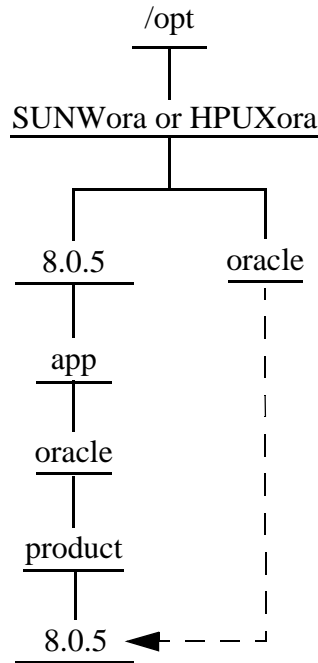


Figure 2-1. Recommended Oracle Software Directory Structure

In each subsection we will specify these environmental specifications (i.e., the `SUNWxxx` or `HPUXxxx` directory, version directory, and symbolic link names) for that package.

NOTE — For the remainder of this section we will use the notation `XXXXxxx` rather than showing both `SUNWxxx` and `HPUXxxx` in all further references.

2.2.2 Environment Settings and `xmyProfile` and `xmyLogin`

`$XMYHOME` is a shell variable that must be defined in your environment. You will be prompted to enter a value for `$XMYHOME` during the BAIST installation process. (See [Section 2.4.2.1](#).) This variable is set in the files `xmyProfile` and `xmyLogin`, which are

installed in the $\$XMYHOME/config$ directory, assuming you have used our recommended directory structure.

Users who wish to access the MYNAH System must source either *xmyProfile* or *xmyLogin* into their *.profile* or *.login*, respectively. See the MYNAH postinstallation steps in [Section 2.4.2.2](#).

NOTE — The MYNAH System is installed into two *different* directory structures.

The $\$XMYDIR$ variable designates the directory link containing the MYNAH software. (Using the recommended directory structure, this is */opt/XXXXmyn/mynah*.) The $\$XMYHOME$ variable designates the directory containing your configuration files and run logs. We require that this be a directory that all MYNAH users have access to since the run log directory needs to have open write permissions. The default will be */opt/XXXXmyn/mynah_home*.

The *xmyProfile* file contains the declarations for **ksh** environment variables for the installed software based on our recommended locations. The *xmyLogin* file contains the declarations for **cs** environment variables for the installed software based on our recommended locations. For example, the recommended location for the home link of the Oracle software, `ORACLE_HOME`, is */opt/XXXXora/oracle*.

The *xmyProfile* and *xmyLogin* files also contain “place-holders” for certain environment settings that will be unique for your installation, such as the Telexel processes port number. As you determine the values for these settings you can either enter them directly into these files or write them down and make all of your changes at one time.

Once you have completed installing the MYNAH System and all other required and optional software packages, you should source the *xmyProfile* or *xmyLogin* file into your *.profile* or *.login*, respectively, depending on which shell you are using.

NOTE — Most of the environment variables will be updated automatically during the BAIST installation process, but the *xmyProfile* and/or *xmyLogin* file need to be verified manually.

The HOME environment variable must exist.

- If you are using **ksh**, source the *xmyProfile* file into your *.profile* (and all MYNAH user's *.profile*'s) by including the following line:

```
. /opt/XXXXmyn/mynah_home/config/xmyProfile
```

See Appendix [B.1.3](#) for an example of the *xmyProfile* file.

- If you are using **cs**h, source the *xmyLogin* file into your *.login* (and all MYNAH user's *.login*'s) by including the following line:

```
source /opt/XXXXmyn/mynah_home/config/xmyLogin
```

See Appendix B.1.4 for an example of the *xmyLogin* file.

NOTE — During installation you may have to reset the environment variable `LD_LIBRARY_PATH` due to its value being lost after executing an **su** command.

NOTE — While the other shells (**sh** and **cs**h) are supported, all example shell commands shown in the installation steps assume you are using **ksh**.

2.2.3 Starting and Stopping MYNAH Software Packages on the Solaris Platform

A series of Solaris start-up scripts for the various MYNAH software packages are delivered with the MYNAH System. The instructions for using these scripts are located in Section 4.8. See Section 4.5.1 for information on the Solaris start-up mechanism.

2.2.4 Requirements

It is required that the `$XMYDIR` and `$XMYHOME` directories be accessible to all MYNAH machines and users. This can usually be accomplished by automounting these file systems on all other machines. If X-Terminals or PC's are used then the user will be logging on to the MYNAH server, so this will not be necessary for these users.

2.3 Preliminary Actions

Before you can run the MYNAH System, there are certain actions you must perform.

2.3.1 Obtaining License Keys

The MYNAH System software uses a floating license scheme involving a license-server daemon that runs on one machine in your network and takes requests for licenses from any machine on your network.

To run the MYNAH System software, you must first install a licensing key.

You must obtain a license for each of the MYNAH 5.4 products listed in [Table 2-2](#) you wish to use.

Table 2-2. Licensed Features

License	Description
XMYGUI	The GUI and Script Builder and their engines
XMYENGINE	engine used by the xmytclsh command and the CLUI (i.e. the Background Script Engine).
XMYTERMASYNC	Asynchronous Terminal Scripting Package
XMYTERM3270	Synchronous Terminal Scripting Package
XMYAPPAPP	General Purpose Application to Application Scripting Package
XMYPRT3270	Synchronous Printer Scripting Package
XMYSCRIPTEXEC	Parent/Child Scripting Scripting Package
XMYTOP	TOP Protocol Application to Application Scripting Package
XMYX25	X.25 Protocol Application to Application Scripting Package
XMYDCE	DCE scripting capability

Telcordia will generate the licenses, the keys are then sent to the MYNAH customers. The contract between the customers and Telcordia defines which of the preceding products will have keys generated, the number of floating licenses, and for what time period.

NOTE — Customers running a version of the MYNAH System lower than Release 5.3 will require new keys for Release 5.4.

Perform the following steps to obtain your license keys:

1. Decide which machine on your network will run the License Server daemon. (The daemon will run only on the machine for which it is licensed.) This is usually the machine on which the MYNAH System software is installed, although it can be any machine on your network.
2. Once you have decided which machine will run the License Server, determine its hostid by typing the following, according to your platform.

```
Solaris      /usr/sbin/sysdef -h
              or
              /usr/sbin/hostid
HP-UX       uname -i
```

WARNING — The License Server software (LicenseServ) does not properly process dashes (-) in a directory path or name.

The hostid (in hexadecimal) is used to create the license key.

3. Once you have obtained this information, call 1-732-699-2668, Option 3 or 1-(800)-795-3119, Option 3 or send e-mail to mynah-support@telcordia.com

You will receive a list containing your key(s). This list is sent to you via the medium of your choice: letter, e-mail (if available), phone, or fax.

2.3.2 Creating the mynah Group and MYNAH Administrator logid (madmin)

Perform the following steps before you begin installing the MYNAH System:

1. Decide which machine will be used as the MYNAH server.
2. Become a superuser, i.e., type

```
su root
```
3. Create a group called **mynah**.
4. Create a logid for the MYNAH Administrator in the group **mynah**, i.e., **madmin**.

NOTE — The MYNAH Administrator, **madmin**, should have a home directory and a default shell e.g., **ksh**.

5. Log onto the machine (as **madmin**) where you will be installing the system.

2.3.3 Setting /tmp_mnt (HP-UX Only)

If you're using the MYNAH System on the HP-UX platform, you must use `/tmp_mnt` for all automounting. This is because of the way HP-UX performs automounting. For example, a home directory of `/u/kjd` is really a symbolic link to `/tmp_mnt/u/kjd`. Therefore, if you submit a script, `/u/kjd/script1`, using CLUI, and have the SE configuration parameter **OutputRoot** (Section 3.3.3) set to `/tmp`, then you would get script output under `/tmp/tmp_mnt/u/kjd` instead of under `/tmp/u/kjd`.

2.3.4 BAIST Considerations

The MYNAH System is delivered using the Telcordia standard UNIX installation process, BAIST. The BAIST archive is delivered via a file archive obtained from the MYNAH **ftp** server. The BAIST package contains archives for the MYNAH upgrade, Telexel, and TOPCOM software. (The TOPCOM software is included only if you purchased the TOPCOM software from Telcordia).

This subsection describes the BAIST preinstallation procedures.

NOTE — Installation must be performed as **root** on the MYNAH machine, and **root** must have write permission to the file system where the installation will be performed.

Create a directory for the BAIST installation database, e.g., */usr/local/BCRDB*, if it has not already been created. This directory must be owned by **root**. You must also perform several actions to set the environment variable called BCRDB to the location of the BAIST installation database directory. How you set BCRDB depends on which UNIX shell (**sh**, **cs**, or **ksh**) you are using.

For all three shells, perform the following:

```
su root
mkdir /usr/local/BCRDB
```

If you're using **sh**, enter

```
BCRDB=/usr/local/BCRDB
LOCAL_BCRDB=yes
export BCRDB LOCAL_BCRDB
```

If you're using **cs**, enter

```
setenv BCRDB /usr/local/BCRDB
setenv LOCAL_BCRDB yes
```

If you're using **ksh**, enter

```
export BCRDB=/usr/local/BCRDB
export LOCAL_BCRDB=yes
```

Create the directory into which the installed software will go (e.g., */opt/XXXXmyn*) prior to performing the installation. This directory must be owned by a user other than **root**, e.g., the MYNAH System Administrator, **madmin**.

For its own scripts, BAIST uses the **ksh** in *\$BCRDB/tools*. However, some of the scripts used by the products BAIST installs, such as configuration scripts, may be using */bin/ksh*. So it will be advisable to install the newer version of **ksh**.

The commands you use to start installation from the BAIST archive depends on which medium you use and will be explained in the following subsections. Whichever method you use, after you start the BAIST archive, the BAIST **Opening Screen** (Figure 2-2) appears.

```
Telcordia Application Installation Setup Tool
- - - - -
                BAIST 2.1.1

COPYRIGHT (c) 1996 Bell Communications Research Inc.,
                All Rights Reserved.

PROPRIETARY - TELCORDIA AND AUTHORIZED CLIENTS ONLY.

                A UNIX Packaging and Installation Tool

                For further information on BAIST
                Contact: Raymond C. Gray
                        BAIST Project Manager
                        (732)699-7960
```

Figure 2-2. BAIST Opening Screen

Following the BAIST **Opening Screen**, the **BAIST Product Menu** (Figure 2-3) appears, prompting you to select the package to install.

```
BAIST FLOW CONTROL
PRODUCT MENU
Archived Products List

1) MYNAH 5.4
2) TELEXEL 7.3
3) TOPCOM 5.4.1

I) Installed Products
R) Registered Products
E) Exit

Enter your Selection:
```

Figure 2-3. BAIST Product Menu

NOTE — This is the menu for the initial release of MYNAH Release 5.4.

NOTE — Remember, the TOPCOM option will only appear if you have purchased the TOPCOM software from Telcordia.

After a software package is installed, the BAIST **Product Menu** will reappear. You can then select to install one of the other packages or exit from the archive. For example, during initial installation of MYNAH 5.4, you may wish to first install the MYNAH software, followed by the Telexel and TOPCOM software. You can then perform the necessary post installation or configuration steps needed by the MYNAH, Telexel, and TOPCOM software packages. You may instead wish to install one software package, exit from the BAIST archive, configure the installed software, and then restart the BAIST archive and install and configure another package.

Prior to installing any package, you must create a home directory for the package. You may find it convenient to create all these directories up front and not individually. (See Sections [2.4.1](#), [2.5.1](#), and [2.8.1](#).)

The following subsections detail installing one package at a time. You may also use this method if you wish to install only one of the packages, such as if you receive an updated version of one software package while retaining the currently installed versions of the other packages.

NOTE — If you wish to install the MYNAH, Telexel, and TOPCOM software packages during one BAIST session, remember to perform the necessary preinstallation steps for each package before starting the archive.

2.4 Installing the MYNAH System

This section contains the information for installing the MYNAH System.

2.4.1 MYNAH System Preinstallation Steps

Perform the following steps before installing the MYNAH System:

1. Become a superuser, i.e., type

```
su root
```
2. Change to the `/opt` directory, i.e., type

```
cd /opt
```
3. Create the `XXXXmyn` directory, i.e., type

```
mkdir XXXXmyn
```
4. Change the owner of `/opt/XXXXmyn` to **madmin**, i.e., type

```
chown madmin /opt/XXXXmyn
```
5. Change the group of `/opt/XXXXmyn` to **mynah**, i.e., type

```
chgrp mynah /opt/XXXXmyn
```

2.4.2 MYNAH Software Installation

The MYNAH System is delivered, along with the Telexel and TOPCOM software (if you purchased the TOPCOM software from Telcordia), via a file archive obtained from the MYNAH **ftp** server. You are prompted for the package you wish to install.

NOTE — When installing the MYNAH System as **root** on a remote filesystem (i.e., installing the software in a filesystem that is not local to the machine on which the installation is being performed), you may not really have **root** access to that file system. The installation could fail with permissions errors.

Install the MYNAH software using one of the following methods:

2.4.2.1 MYNAH Software File Archive Installation

If you are installing from a file, perform the following steps:

1. Obtain the BAIST archive file from the **ftp** server.
2. Enter the following command:

```
/bin/sh archive-file-name
```

where *archive-file-name* is the name of the BAIST archive file.

3. Select the **MYNAH 5.4** option from the **BAIST Product Menu** (Figure 2-3).

NOTE — See Appendix B.1.1 for an example of a BAIST installation session of the MYNAH System software.

4. Answer the following questions asked by the BAIST installation software:

Where should the MYNAH directory be created? **/opt/XXXXmyn**

Who should own the MYNAH application? **admin**

Where should XMYHOME be? **<enter your location for XMYHOME>**

The default is /opt/XXXXmyn/mynah_home

Please enter your email address

Enter your name (first & last)

Enter your phone # **NPA NNX-LINE**

Enter your company name

Do you want group execution permissions for xmyStartOA? **(No)**

NOTE — *\$XMYHOME* must be different than the MYNAH software directory (*\$XMYDIR*).

5. After the MYNAH software is installed, perform one of the following:
 - Install MYNAH 5.4 update if it exists, the Telexel (Section 2.5) or TOPCOM (Section 2.8) software packages if they have not already been installed, and then exit from the BAIST archive.
 - Exit from the BAIST archive and perform the MYNAH post installation steps in Section 2.4.2.2.

2.4.2.2 MYNAH Post Installation Steps

Once you've installed the software, verify and correct, if necessary, the value assigned to XMYDIR in your working copy (in *\$XMYHOME/config*) of the *xmyProfile* or *xmyLogin* files.

1. Become **madmin**, i.e., type

```
su madmin
```

2. Verify that a symbolic link exists pointing to the version directory (created by the BAIST installation process), i.e., type

```
cd /opt/XXXXmyn
```

```
ls -al
```

NOTE — If a previous version of the MYNAH System already exists, the link still points to the old version. The link will have to be deleted and moved to the new directory. *The MYNAH System must be down to do this.*

3. Verify and correct, if necessary, the value assigned to XMYDIR in Step 4 of [Section 2.4.2.1](#) in your working copy of the *xmyProfile* or *xmyLogin* files.

NOTE — See [Section 4.8.1](#) for information on automatically starting the MYNAH processes at boot time.

2.4.3 Changes to */etc/services* and */etc/system*

During installation, several changes must be made to the */etc/services* file. For example, when configuring the Telexel System ([Section 2.5.3](#)), you must define and export **vxIpcPort**, which is the port used by the Telexel processes.

During installation, a file called *etc.system.changes.eg* is placed in the *\$XMYDIR/examples/admin/scripts* directory. This file contains changes that should be added to the */etc/system* file. Once the changes have been made, type

```
reboot -- -rt
```

to reboot the system. The system will be reconfigured with the changes to */etc/system* incorporated in the kernel. You must do this as **root** on each system running a MYNAH component and on the ORACLE server.

NOTE — See [Appendix B.1.2](#) to see a copy of the *etc.system.changes.eg*.

2.5 Installing the Telexel System

The MYNAH 5.4 System requires the Telexel System Release 7.3 for inter-process communications and logging.

NOTE — The Telexel System must be installed on a local filesystem or on a remote filesystem that is mounted with the **suid** option. If a previous release of the Telexel System has been installed, you must re-install the Telexel system.

If you have not installed Telexel 7.3, install it.

2.5.1 Telexel System Preinstallation Steps

Perform the following steps before installing the Telexel System:

1. Become a superuser, i.e., type

```
su root
```
2. Change to the */opt* directory, i.e., type

```
cd /opt
```
3. Create the *XXXXtel* directory, i.e., type

```
mkdir XXXXtel
```
4. Change the owner of the *XXXXtel* directory to **madmin**, i.e., type

```
chown madmin XXXXtel
```
5. Change the group of the *XXXXtel* directory to **mynah**, i.e., type

```
chgrp mynah XXXXtel
```

NOTE — The Telexel installation process automatically creates a separate directory for the version of the software being installed. In addition, the software package includes a utility program, **vxInstall**, that creates a series of symbolic links from the version directory into */opt/XXXXtel*. This is handled by the BAIST Telexel post installation step.

Make sure the MYNAH System can access the Telexel System.

2.5.2 Telexel Installation

The Telexel System is delivered, along with the MYNAH and TOPCOM software (if you purchased the TOPCOM software from Telcordia), via a file archive obtained from the MYNAH **ftp** server. You are prompted for the package you wish to install.

NOTE — When installing the Telexel System as **root** on a remote filesystem (i.e., installing the software in a filesystem that is not local to the machine on which the installation is being performed), you may not really have **root** access to that file system. The installation could fail with permissions errors.

Install the Telexel software using one of the following methods:

2.5.2.1 Telexel Software File Archive Installation

If you are installing from a file, perform the following steps:

1. Obtain the BAIST archive file from the **ftp** server.
2. Enter the following command:

```
/bin/sh archive-file-name
```

where *archive-file-name* is the name of the BAIST archive file.

3. Select the **Telexel 7.3** option from the **BAIST Product Menu** (Figure 2-3).

NOTE — See Appendix B.2 for an example of a BAIST installation session of the Telexel System software.

4. Answer the following questions asked by the BAIST installation software:

```
Where should the TELEXEL directory be created? /opt/XXXXtel
```

```
Who should own the TELEXEL application? admin
```

5. After the Telexel software is installed, perform one of the following:
 - Exit from the BAIST archive and perform the Telexel post installation, configuration, and verification steps in Sections 2.5.2.2, 2.5.3, and 2.5.4, respectively.
 - Install the MYNAH (Section 2.4) or TOPCOM (Section 2.8) software packages if they have not already been installed, and then exit from the BAIST archive.

2.5.2.2 Telexel Post Installation Steps

Once you've installed the software, execute the following commands to verify that a symbolic link exists to the version directory:

```
su madmin
cd /opt/XXXXtel
ls -al
```

NOTE — If a previous version of the Telexel System already exists, the link will have to be deleted and moved to the new directory. *The MYNAH System must be down to do this.*

Additionally, if a previous Telexel System installation already exists, you must re-establish, as **root**, the link between $\${TELDIR}/lib/libC.so.3.1$ and $/usr/lib/libC.so.3.1$.

2.5.3 Configuring the Telexel System

To configure the Telexel System, you must define and export the following environment variables in your *xmyProfile* or *xmyLogin* file. You must have already installed and configured MYNAH for these files to exist before performing the steps below.

NOTE — Remember to edit your working copy of the *xmyProfile* or *xmyLogin* file, defining and exporting these variables.

1. Become **madmin**, i.e., type

```
su madmin
```

NOTE — Steps 2 through 7 can be accomplished as per [Section 2.2.2](#) or manually as described below.

2. Define and export **vxIpcPort**, which is the port used by the Telexel processes, e.g.,

```
export vxIpcPort=22100
```

This must be a valid, unused port between 1024 and 65000. To determine which ports are not being used, examine the */etc/services* file.

WARNING — If multiple versions of MYNAH 5.X are running on the same machine (e.g., you are using Release 5.3 and also testing Release 5.4), then each version must use a different value for **vxIpcPort**.

NOTE — Remember to edit the */etc/services* file, entering a line such as the following to add the port you select.

```
vxIpcPort 22100/tcp #vxIpcDir
```

3. Define and export **vxIpcDirectory**, which is the machine on which the Telexel System is installed. For example, if you installed the Telexel System on a machine named **selene**, enter the following in the *xmyProfile* or *xmyLogin* file:

```
export vxIpcDirectory=selene
```

4. Define and export the **TELDIR** variable, which is the directory where the Telexel System is installed. This will be set to the link you created to the actual directory containing the software, i.e.,

```
export TELDIR=/opt/XXXXtel/telexel
```

5. Define and export the **LD_LIBRARY_PATH** variable, e.g.,

```
export LD_LIBRARY_PATH=/usr/openwin/lib:$TELDIR/lib:\nLD_LIBRARY_PATH
```

6. Add *\$TELDIR/bin* to your *\$PATH* declaration, i.e.,

```
export PATH=$PATH:$TELDIR/bin
```

7. Add *\$TELDIR/man* to your *\$MANPATH* declaration, i.e.,

```
export MANPATH=$MANPATH:$TELDIR/man
```

8. If you are installing the MYNAH System on a Solaris machine, the following constraints must be followed in the */etc/system* file:

- A. The Message Queue parameters *msgmnb* and *msgmax* must be equal, e.g.,

```
set msgsys:msginfo_msgmnb=65535\nset msgsys:msginfo_msgmax=65535
```

NOTE — *msgmnb* defines the maximum bytes on the queue, and *msgmax* sets the maximum size of the message.

- B. *msgmax* must be less than or equal to the product of *msgseg* and the *msgssz* parameters.

NOTE — *msgseg* sets the number of message segments, and *msgssz* sets the segment size of message.

For example, if *msgseg* is set to 16384 and *msgssz* is set to 64, then *msgmax* must be less than or equal to 1,048,576.

2.5.4 Verifying the Telexel System

CAUTION — The message

"MYNAH MAY NOT BE RUNNING...CHECK WITH YOUR MYNAH ADMINISTRATOR!!"

will appear in the standard output if the Telexel processes are not running.

To manually verify your Telexel installation, **cd** to *\$TELDIR/bin* then perform the following tasks:

1. Become **madmin** and change to the *\$TELDIR/bin* directory, i.e., type

```
su - madmin
cd $TELDIR/bin
```

2. Type

```
./vxIpcDir
```

to start the IPC process.

3. Type

```
./vxIpcProcesses
```

to verify that the process started. You should get this:

```
IPC Registered Processes
ID                PID    HOST                QUEUE USER
==                ===    ====                =====
```

If you get this,

```
vxIpcProcesses: can't retrieve process list (error 1-IP-0024, errno 146).
```

then **vxIpcDir** did not start. If you have to restart this process, you may have to wait for the port to time-out.

4. Type

```
./vxIpcUp
```

to start the Telexel Gateway. This starts a process called **vxIpcRecvd**.

5. Type

```
./vxLogToFile $XMYHOME/syslog/adminLog
```

to start the Telexel Logger.

6. Type

```
./vxErrorServer $TELDIR/lib/errorText \  
$XMYDIR/lib/xtw_error_text $XMYDIR/lib/xmyErrorText
```

to start the Telexel Error Server.

NOTE — The XMYDIR environment variable must be set to the MYNAH 5.4 installation directory. Both the XMYHOME and XMYDIR variables must be exported for Steps 5 and 6. See [Section 2.4.2.2](#). This was done as part of MYNAH installation.

7. Execute

```
ps -ef | grep vx
```

You should expect output similar to the following:

```
madmin 4479 1 80 Jul 01 ? 9:59 vxIpcDir  
madmin 4511 1 80 Jul 01 ? 0:31 /opt/XXXXtel/telexel/lib/vxIpcRecvd  
madmin 22281 1 80 Jul 05 ? 0:01 vxLogToFile /opt/XXXXmyn/mynah/syslog/adminLog  
madmin 22290 1 80 Jul 05 ? 0:02 vxErrorServer /opt/XXXXtel/telexel/lib/errorText
```

These four processes should always be running on the MYNAH server.

NOTE — Other vx (Telexel) processes may also appear. The preceding processes constitute the minimum set.

2.6 Installing Oracle (Optional)

MYNAH test management makes heavy use of a database. If you are using the MYNAH System only for task automation, then installing Oracle is optional. For test management it is required.

NOTE — The Oracle System routinely creates archival files of database operations, which, over time, will decrease available disc space. It is your responsibility to perform all Oracle file maintenance and archival procedures (e.g., backup and removal) of Oracle files and archives.

Although you will follow the Oracle installation procedures, there are certain factors that you must take into consideration. For example, you must install several specific Oracle packages and create specific environment variables. This subsection discusses these factors.

NOTE — This subsection assumes you do not have an existing Oracle database and are installing one for use with the MYNAH System.

2.6.1 Oracle Preinstallation Steps

NOTE — Refer to the *Oracle Installation and Configuration Guide* for the version of the software being loaded. (Hereafter we refer to this guide as the *Oracle Manual*). For this installation we will reference those items relating to Oracle 8.0.5.

Perform the following steps before installing the Oracle System:

NOTE — The following steps assume you are using the **ksh** shell.

1. Become a superuser, i.e., type

```
su root
```
2. Create a new user named **oracle** with a group id of **dba**. (If necessary, create the group first.)
3. Change to the `/opt` directory, i.e., type

```
cd /opt
```

4. Create the *XXXXora* directory, i.e., type

```
mkdir XXXXora
```

where *XXXX* is *SUNW* for Solaris systems or *HP-UX* for HP-UX systems.
5. Change to the */opt/XXXXora* directory, i.e., type

```
cd /opt/XXXXora
```
6. Under */opt/XXXXora*, create the version directory, i.e., type

```
mkdir 8.0.5
```
7. Create the symbolic link pointing to the version directory in */opt/XXXXora*, i.e., type

```
ln -s /opt/XXXXora/8.0.5/app/oracle/product/8.0.5 oracle
```
8. The owner of the */opt/XXXXora* directory must be set to **oracle** and the group id to **dba**. Execute the following steps:

```
cd /opt  
chown -R oracle XXXXora  
chgrp -R dba XXXXora
```

where *XXXX* is *SUNW* for Solaris systems or *HP-UX* for HP-UX systems.
9. When installing Oracle, ensure that the correct packages are loaded. For Oracle Release 8.0.5, the following packages are required:

Solaris Version	HP-UX Version
SUNWbtool	B3191A
SUNWspot	B3193A
SUNWtoo	B3519AA
SUNWarc	B3693AA-APS
SUNWlibm	B3900AA-APS
SUNWlibms	B3912AA-APS
	B3920AA
	B4905BA
	HPuXEngGS800

You can use the Oracle utility **pkginfo** to ensure that a package exists. For example, to verify that the *SUNWbtool* package has been loaded (if you are using the Solaris version), type

```
pkginfo -i SUNWbtool
```

See Solaris version of the *Oracle Manual* for updated information.

To verify that the HPUXlibm package has been loaded (if you are using the HP-UX version), type

```
/usr/sbin/swlist -l prod
```

See HP-UX version of the *Oracle Manual* for updated information.

10. Define and export the following environment variables:

- For Solaris

```
export ORACLE_HOME=/opt/XXXXora/oracle
export PATH=$PATH:/$ORACLE_HOME/bin
export ORACLE_TERM=sun5
```

- For HP-UX

```
export ORACLE_HOME=/opt/XXXXora/oracle
export PATH=$PATH:/$ORACLE_HOME/bin
export ORACLE_TERM=hpterm
```

11. Depending on whether Oracle is installed on a local or remote server, update either `ORACLE_HOME` or `TWO_TASK` in your working copy of the *xmyProfile* or *xmyLogin* file as follows:

- For a local installation

```
export ORACLE_SID=mynah5
```

- For a remote installation

```
export TWO_TASK=mynah5
```

12. Create a directory under `$ORACLE_HOME` named *mynah5* with the owner set to **oracle** and the group to **dba**, i.e., type, as root,

```
cd $ORACLE_HOME
mkdir mynah5
chown oracle mynah5
chgrp dba mynah5
```

13. Under `$ORACLE_HOME/mynah5` create the *datafiles* and *logfiles* directories with the owner set to **oracle** and the group to **dba**:

```
cd mynah5
mkdir datafiles logfiles
chown oracle datafiles logfiles
chgrp dba datafiles logfiles
```

14. The following is the recommended Oracle database disk configuration:

- disk1 - Tables and system
- disk2 - Index and some logs and control files
- disk3 - Rollback segments.

If you have only one disk available for the Oracle database, proceed to Item A. If you have multiple disks, proceed to Item B.

A. Under `$ORACLE_HOME/mynah5/datafiles` create the directories `d01`, `d02`, and `d03` with the owner set to **oracle** and the group to **dba**, i.e., type

```
cd datafiles
mkdir d01 d02 d03
chown oracle d01 d02 d03
chgrp dba d01 d02 d03
```

NOTE — Directory `d01` contains items in disk1, directory `d02` contains items in disk2, and directory `d03` contains items in disk3.

Proceed to Step 15.

B. Under `$ORACLE_HOME/mynah5/datafiles` create the following symbolic links to the directories on other disks, e.g., type

```
cd datafiles
ln -s <disk1> d01
ln -s <disk2> d02
ln -s <disk3> d03
```

Proceed to Step 15.

15. Copy the following files from `$XMYDIR/examples/dbadmin` to `$ORACLE_HOME/mynah5`:

- `configmynah5.ora` (See Appendix B.3.2 for an example `configmynah5.ora`.)
- `initmynah5.ora` (See Appendix B.3.3 for an example `initmynah5.ora`.)
- `crdbmynah5.sql` (See Appendix B.3.4 for an example `crdbmynah5.sql`.)
- `crdb2mynah5.sql` (See Appendix B.3.5 for an example `crdb2mynah5.sql`.)
- `crdb3mynah5.sql` (See Appendix B.3.6 for an example `crdb3mynah5.sql`.)

16. After copying the files, change the permissions and ownership on the files by executing the following:

```
cd $ORACLE_HOME/mynah5
chmod -R 775 *
chown oracle *.ora
chgrp dba *.ora
chown oracle *.sql
chgrp dba *.sql
```

Edit the **.ora* and **.sql* files to update the correct directories paths (such as where Oracle is installed). You may also need to change the sizing information.

NOTE — Do not use environment variables to define these paths.

17. Make the changes to the */etc/system* file (See Section 2.4.3) as shown in *\$XMYDIR/examples/admin/scripts/etc.system.changes.Sun.eg* for Solaris or *\$XMYDIR/examples/admin/scripts/etc.system.changes.HP.eg* for HP-UX
18. Reboot the system with the reconfiguration before proceeding with [Section 2.6.2](#).

To reboot, execute

```
reboot -- -rt
```

2.6.2 Oracle Installation

If Oracle 8.0.5 has not already been installed, perform the following steps.

For assistance, refer to the *Oracle Manual*.

NOTE — Do not use environment variables or symbolic links until Oracle is installed.

1. Mount the CD-ROM containing the Oracle software.
 - A. For Solaris, use one of the three following methods:
 - Execute

```
mount -r -F hsfs /dev/dsk/cot6d0s1 /cdrom
```
 - Execute

```
volcheck cdrom
```
 - If you're using volume manager, the CD-ROM is automatically mounted when you insert the disk.
 - B. For HP-UX, execute the appropriate mount command for the HP-UX operating system.
2. Become the user **oracle**, i.e., type

```
su - oracle
```
3. Change to the */cdrom/oracle/orainst* directory. i.e., type

```
cd /cdrom/oracle#1/orainst
```
4. Make sure the terminal type is set to vt100, and clear the screen, i.e., type

```
export TERM=vt100  
clear
```
5. Start an xterm window, i.e., type

```
xterm &
```

This will let you use the arrow keys and display used by the Oracle installation software.
6. Start the Oracle install program. i.e., type

```
./orainst
```

Table 2-3 lists the Oracle installation items and appropriate responses you must perform to ensure that the database will work properly with the MYNAH System.

Table 2-3. Oracle Installation Items and Responses (Sheet 1 of 2)

Item	Response
Preamble.txt	OK
Installation Activity Choice	Install, Upgrade, or De-Install
Installation Option	Install New Product
Mount Point	<i>/opt/XXXXora/8.0.5</i>
Home Location	8.0.5
DB Objects - Create?	No
Logging and Status	OK
readme.first	OK
Skip readme	OK
Install source	CD-ROM
NLS	Amer.
Relink all Exec.	No
Information	OK
On-line Help Load	All Prod (Optional)
UNIX Documentation	Yes (Optional)
Product Documentation Library	All Prod (Optional)
Oracle Documentation	<i>/opt/XXXXora/8.0.5/app/oracle/doc</i>
Software Asset Manager	At this step of the installation process, the Oracle System displays a scrollable list showing the available Oracle packages. To select which packages to install <ol style="list-style-type: none"> 1. Use the arrow keys to scroll through the list of packages. 2. Press the space bar to select a package when it is highlighted. 3. When you have selected all of your desired packages, press the TAB key to go to the Install button and press the Return key. <p>Table 2-4 lists the packages that must be installed.</p>
Official Hostname	Enter your machine name (including the domain).
TCP Surf Port	8888 or any unused port number.
Password and Verify Password	any

Table 2-3. Oracle Installation Items and Responses (Sheet 2 of 2)

Item	Response
DBA Group	OK
OSOPER Group	OK
DBA doesn't exist	Yes (continue)
Enter Oracle Sid	mynah5
X Libraries	/usr/openwin/lib
Run root.sh	OK

Table 2-4. Oracle Software Asset Manager Packages

Package	Version Number
Oracle Server Manager	V2.3.2.0.0
Oracle UNIX Installer	V4.0.0.0.0
Oracle Server RDBMS	V2.3.2.3.0
PL/SQL	V2.3.2.3.0
SQL*Net (V2)	V2.3.2.1.0
SQL*Plus	V3.3.2.0.0
TCP/IP Protocol Adapter	V2.3.2.3.0

2.6.2.1 Verifying an Installation

To verify your actions during the installation, view the *install.log* file by executing either of the following in *\$ORACLE_HOME/orainst*:

```
tail -f install.log  
cat install.log | grep code
```

You should not get errors from the install process (i.e., all return codes should equal 0).

2.6.2.2 Oracle Error Messages

If you get ORACLE error messages, type

```
oerr xxx #####
```

to find out what the error is, where **xxx** is *ORA* or *DBA* and **#####** is the error number.

2.6.3 TNS Network Configuration

The Oracle environment must be configured for the TNS Listener process. These files are either stored in `/var/opt/oracle` or in `$ORACLE_HOME/network/admin`. In the latter case or if these files are in a directory other than `/var/opt/oracle`, then an environment variable, `TNS_ADMIN`, must be defined to point to this directory. See the `S96oracle.eg` file in `$XMYDIR/examples/admin/scripts` for an example.

NOTE — This environment variable must be updated in the `$XMYHOME/config/xmyProfile` and `$XMYHOME/config/xmyLogin` files.

Sample `tnsnames.ora.eg` and `lisenter.ora.eg` files are also included in `$XMYDIR/examples/admin/scripts` and in Appendix C. These example files must be edited for your environment, renamed without the `.eg` extension, and moved to the desired location.

2.6.4 Configuring the MYNAH System Oracle Database

NOTE — This subsection contains several example executions of the utilities used to configure the Oracle database. All-user supplied entries appear in bold.

Since Oracle is now installed, you are ready to configure the MYNAH Oracle database.

1. If you are still not a superuser, become one, i.e., type

```
su root
```

2. Create a directory in `/var/opt` named `oracle` with the owner set to **oracle** and the group to **dba**, i.e., type

```
cd /var/opt
mkdir oracle
chown oracle oracle
chgrp dba oracle
```

Make this directory a symbolic link from `/var/opt/oracle/oratab` to `/etc/oratab`.

3. Copy the `initmynah5.ora` and the `configmynah5.ora` files from `$ORACLE_HOME/mynah5` (See preinstallation Step 15 in Section 2.6.1) to `$ORACLE_HOME/dbs`. Make sure these files have been edited and updated.

4. Execute

```
chown oracle *.ora
chgrp dba *.ora
chown oracle *.sql
chgrp dba *.sql
```

5. Change to `$ORACLE_HOME/mynah5` and become the user **oracle**, i.e., type

```
cd $ORACLE_HOME/mynah5
su oracle
```

6. Depending on whether Oracle is installed on a local or remote server, one of the following:

- For a local installation

```
export ORACLE_SID=mynah5
```
- For a remote installation

```
export TWO_TASK=mynah5
```

7. Execute the following commands (being sure to complete the preinstallation Step 14 in [Section 2.6.1](#) first):

```
svrmgrl
SVRMGR> connect internal
SVRMGR> startup nomount pfile=/opt/XXXXora/oracle/dbs/initmynah5.ora
SVRMGR> @crdbmynah5.sql
SVRMGR> connect internal
SVRMGR> @crdb2mynah5.sql
SVRMGR> connect internal
SVRMGR> @crdb3mynah5.sql
```

NOTE — These steps may take a while to run.

8. Execute the following commands:

```
svrmgrl
SVRMGR> connect internal
SVRMGR> shutdown immediate
SVRMGR> exit
```

9. Change to `$ORACLE_HOME/orainst`, i.e., type

```
cd $ORACLE_HOME/orainst
```

10. Become a superuser, i.e., type

```
su root
```

11. Execute on of the following:

- For a local installation
`export ORACLE_SID=mynah5`
- For a remote installation
`export TWO_TASK=mynah5`

12. Run **root.sh**, which was created by the Oracle install process.

The following is a sample run.

```
./root.sh  
Running ORACLE7 root.sh script...
```

The following environment variables are set as:

```
ORACLE_OWNER= oracle  
ORACLE_HOME= /opt/XXXXora/oracle  
ORACLE_SID= mynah5
```

Are these settings correct (Y/N)? [Y]: **Y**

Enter the full pathname of the local bin directory
[/opt/bin]: **/usr/local/bin**

```
Checking for "oracle" user id...  
ORACLE_HOME does not match the home directory for oracle.  
Okay to continue? [N]: Y
```

```
Creating /var/opt/oracle/oratab file...  
Updating /var/opt/oracle/oratab file...
```

Please raise the ORACLE owner's ulimit as per the IUG.

```
Leaving common section of ORACLE7 root.sh.  
Setting orasrv file protections
```

13. Edit the `/var/opt/oracle/oratab` file, then change the *N* to *Y* on the actual data line (i.e., the last line).

WARNING — Do not use a symbolic link name in this file. This entry is case-sensitive, so you must enter a capital **Y**

See the comments in the `oratab` file for more information.

14. Become the user **oracle**, i.e., type

```
su oracle
```

15. Execute the following commands:

```
svrmgrl

svrmgrl: Release 8.0.5 - Production on Thu Apr 18 11:07:43 1996

Copyright (c) Oracle Corporation 1979, 1994. All rights reserved.

Oracle8 Server Release 8.0.5 - Production Release
With the distributed, replication and parallel query options
PL/SQL Release 2.1.6.2.0 - Production

connect internal
Connected.
startup pfile=/opt/XXXora/oracle/dbs/initmynah5.ora
ORACLE instance started.
Total System Global Area          5079016 bytes
      Fixed Size                   39696 bytes
      Variable Size                4621528 bytes
Database Buffers                  409600 bytes
Redo Buffers                       8192 bytes
Database mounted.
Database opened.
exit
```

16. To verify the database creation process, execute the following:

```
svrmgrl
SVRMGR> connect internal
SVRMGR> select * from v$controlfile;
$ORACLE_HOME/dbs/ctrl1mynah5.ctl
$ORACLE_HOME/mynah5/datafiles/d02/ctrl2mynah5.ctl
$ORACLE_HOME/mynah5/datafiles/d03/ctrl3mynah5.ctl
SVRMGR> select file_name,status from dba_data_files;
$ORACLE_HOME/mynah5/datafiles/d01/systmynah5.dbf      AVAILABLE
$ORACLE_HOME/mynah5/datafiles/d02/rbsmynah5.dbf      AVAILABLE
$ORACLE_HOME/mynah5/datafiles/d03/toolmynah5.dbf     AVAILABLE
$ORACLE_HOME/mynah5/datafiles/d03/my5mynah5.dbf     AVAILABLE
$ORACLE_HOME/mynah5/datafiles/d01/usrmynah5.dbf     AVAILABLE
$ORACLE_HOME/mynah5/datafiles/d01/tempmynah5.dbf    AVAILABLE
SVRMGR> exit
```

17. Become the MYNAH Administrator (**madmin**), i.e., type

```
su - madmin
```

18. Execute *xmyProfile* or *xmyLogin* in the MYNAH config directory (*\$XMYHOME*).

NOTE — For HP-UX only, Oracle normally recommends that you use **ORACLE_SID** instead of **TWO_TASK** when your program and the Oracle database are on the same machine. However, the MYNAH System design requires that you use **TWO_TASK** instead of **ORACLE_SID**. Make sure **TWO_TASK** equals *mynah5*.

19. Create the MYNAH database using the **xmyCreate** commands (e.g., **xmyCreateSequences** and **xmyCreateDemoObjects**).

A. **cd** to `$XMYDIR/dbadmin`.

B. Execute

```
./xmyCreateTables
```

You should see the following:

```
Database connection opened.  
Tables do not exist, creating them.
```

C. Execute

```
./xmyCreateViews
```

You should see the following:

```
View created.
```

```
View created.
```

xmyCreateViews creates a new MYNAH view in the Oracle database.

NOTE — Oracle must be up before you execute **xmyCreateViews**.

D. Execute

```
./xmyCreateSequences
```

xmyCreateSequences creates all of the sequences needed by the MYNAH database to operate.

See Appendix [B.3.7](#) for an example **xmyCreateSequences** execution.

E. Execute

```
./xmyCreateTemplates
```

See Appendix [B.3.8](#) for an example **xmyCreateTemplates** execution.

F. Execute

```
./xmyCreateDemoObjects
```

xmyCreateDemoObjects creates all of the demonstration objects.

You should see the following:

```
connecting to database  
demo objects added ...
```

20. Verify the TCP port in the `/etc/services` file by searching for an entry such as the following:

```
tnslsnr 1521/tcp #oracle listener
```

If this line is not present and NIS is not used, add this line to the `/etc/services` file.

WARNING — You must do this in every `/etc/services` file on all machines running the MYNAH System.

If NIS is used, then the following can be in the `ypservices` file to verify this `/etc/services` file entry:

```
ypcat services | grep 1521
```

If the 1521 port is being used, then choose another port on all machines that are not being used.

2.6.5 Dropping the Oracle Database

During installation, the following commands are placed in `$XMYDIR/dbadmin`:

- **xmyDropSequences.**
- **xmyDropTables**

Execute these commands if something goes wrong and the database needs to be cleaned up/deleted.

If problems are encountered with the database, consult MYNAH support before running the previous commands.

WARNING — These operations are drastic in nature, and thus they should not be run if the database has been populated with important data. They may, however, be useful when first installing a database.

Once these commands have finished executing, run the **xmyCreate** commands (Section 2.6.4, Step 19) to recreate the tables in the MYNAH database.

2.6.6 Verifying Oracle

To verify if everything is up and working in Oracle, do the following:

1. Copy the *S96oracle.eg* file from *\$XMYDIR/examples/admin/scripts* to */etc/rc2.d* (for Solaris) or */etc/rc2.d* (for HP-UX), then rename it *S96oracle* (for Solaris) or *S90oracle* (for HP-UX).
2. Set the `ORACLE_HOME` variable in *S96oracle* to the correct path, then verify all other paths.
3. Execute

```
su root
/etc/rc3.d/S96oracle start
or
/etc/rc3.d/S90oracle start
```

The following Oracle processes should be up and started:

- tnslsnr LISTENER
 - ora_reco_mynah5
 - ora_smon_mynah5
 - ora_pmon_mynah5
 - ora_lgwr_mynah5
 - ora_dbwr_mynah5
 - ora_s000_mynah5
 - ora_d000_mynah5
4. Execute a command of the following form:

```
svrmgrl system/manager@mynah5
```

You should get a good connection.

You can also try any of the following to verify your Oracle installation:

1. Verify the existence of the *\$ORACLE_HOME/rdbms/log/alert_mynah5.log* file.
2. If problems occur, try to set up a symbolic link in */etc*, for example,

```
ln -s /var/opt/oracle/oratab oratab
```


3. Verify

```
oracle      permissions 6751  
in $ORACLE_HOME/bin.
```

4. If you must relink, try the following:

```
$ORACLE_HOME/rdbms/lib/usr/ccs/bin/make -f oracle.mk install  
$ORACLE_HOME/network/lib/usr/ccs/bin/make -f network.mk install  
$ORACLE_HOME/sqlplus/lib/usr/ccs/bin/make -f sqlplus.mk install
```

5. To verify what is linked in Oracle, use the **adapters** and **drivers** command, as in the following:

```
adapters  
  Installed SQL*Net V2 Protocol Adapters are:  
  V2 BEQ Protocol Adapter  
  V2 IPC Protocol Adapter  
  V2 TCP/IP Protocol Adapter  
  V2 Raw Protocol Adapter
```

NOTE — See Section 4.8.2 for information on automatically starting the Oracle processes at boot time.

2.7 Installing I/O Concepts' X-Direct TN3270 (Optional)

If you wish to test or automate tasks over a 3270 (synchronous) interface, you must install the X-Direct TN3270 software from I/O Concepts.

2.7.1 Installing X-Direct TN3270

Perform the following steps to install the X-Direct TN3270 software:

1. Become a superuser, i.e., type

```
su root
```
2. Change to the `/opt` directory, i.e., type

```
cd /opt
```
3. Create the directory `XXXXioc`, i.e., type

```
mkdir XXXXioc
```
4. Under `/opt/XXXXioc`, create the version directory, `9.1.7`, i.e., type

```
mkdir 9.1.7
```
5. Create the symbolic link pointing to the version directory, i.e., type

```
ln -s /opt/XXXXioc/9.1.7 ioconcepts
```
6. Install the X-Direct TN3270 software in `/opt/XXXXioc/ioconcepts`. (See the X-Direct TN3270 manual.)

2.7.2 Configuring X-Direct TN3270

After you've finished installing the I/O Concepts software, you must configure several X-Direct TN3270 parameters so that the software works with the MYNAH System. To do this, perform the following tasks:

1. Edit the `ioclm.sh` shell in `/opt/XXXXioc/ioconcepts` for the following:
 - `KEYDIR=<library>/ioclm`
 - `EXDIR=<library>`
 - `LOGFILE=<library>/ioclm.log`

where `<library>` is `/opt/XXXXioc/ioconcepts`.

2. To add License keys, make sure you are logged in as **root**, and type

```
./iocadmin -c -e <library>/ioclm
```

where <library> is */opt/XXXIoc/ioconcepts*

3. Change the parameters in the TN3270 printer emulator config file, *iocluprt.cfg*, to let the PRT3270 subsystem receive print messages. The parameters and values needed for the MYNAH PRT3270 package are

```
printerCmd =      "xmyPrt3270Collector"  
printerOptions = "-P printer_1"  
fileName =       0  
luType =         1
```

2.7.3 Verifying X-Direct TN3270

NOTE — The X-Direct TN3270 software can be started automatically when the MYNAH System starts by using the *\$XMYHOME/config/xmyConfigOP* file. (See [Section 3.4](#) for information on setting up the *xmyConfigOP* file).

The process *iocLicense* must be uncommented as follows:

```
Process iocLicense  
  Mynah      = no,  
  Autostart  = yes,  
  Start      = "ioclm.sh start",  
  Stop       = "ioclm.sh stop",  
  Status     = "ps -ef | grep ioclmd";
```

To manually bring up and test the X-Direct TN3270 software, perform the following tasks in */opt/XXXIoc/ioconcepts*:

1. To set the *IOCLM_HOST* environment variable, execute

```
export IOCLM_HOST=<machine_name>
```

where <machine_name> is the name of the host on which you installed the X-Direct TN3270 software.

NOTE — Remember to update *IOCLM_HOST* in the *xmyProfile* or *xmyLogin* files.

2. To bring up the license server, type

```
./ioclm.sh start
```

3. To verify your configuration actions, examine the *ioclm.log* file, e.g., type

```
vi ioclm.log
```

4. To stop the license server, type

```
./ioclm.sh stop
```

2.8 Installing TOPCOM (Optional)

TOPCOM is needed if you want to perform TOPCOM Application-to-Application scripting.

2.8.1 TOPCOM System Preinstallation Steps

If you have purchased TOPCOM from Telcordia, Version 5.4.1 of the software is delivered with the MYNAH System as a BAIST archive as part of a file archive obtained from the MYNAH **ftp** server. You are prompted for the package you wish to install.

NOTE — Unlike the previous packages, it is not assumed that you are installing TOPCOM in */opt* nor do you need to perform any of the other Environment Consideration actions. However, *you may find it useful to follow these suggestions* (e.g., the same naming conventions and directory structure) to ensure consistency.

Perform the following steps before installing the TOPCOM software:

1. Become a superuser, i.e., type

```
su root
```
2. Change to the */opt* directory, i.e., type

```
cd /opt
```
3. Create the directory *XXXXtop*, i.e., type

```
mkdir XXXXtop
```
4. Change the owner of the *XXXXtop* directory to **madmin**, i.e., type

```
chown madmin XXXXtop
```
5. Change the group of the *XXXXtop* directory to **mynah**, i.e., type

```
chgrp mynah XXXXtop
```

2.8.2 TOPCOM Installation

Install the TOPCOM System using one of the following methods:

2.8.2.1 File Archive Installation

If you are installing from a file, perform the following steps:

1. Obtain the BAIST archive file from the **ftp** server.
2. Enter the following command:

```
/bin/sh archive-file-name
```

where *archive-file-name* is the name of the BAIST archive file.

3. Select the **TOPCOM 5.4.1** option from the **BAIST Product Menu** (Figure 2-3).
4. Answer the following questions asked by the BAIST installation software:

```
where should the TOPCOM directory be created: <enter your  
location for the TOPCOM directory>
```

```
who should own the TOPCOM application: admin
```

5. After the TOPCOM software is installed, perform one of the following:
 - Exit from the BAIST archive.
 - Install the MYNAH (Section 2.4) or Telexel (Section 2.5) software packages if they have not already been installed, and then exit from the BAIST archive.

2.8.2.2 TOPCOM Post Installation Steps

Once you've installed the software, execute the following commands to verify that a symbolic link exists to the version directory:

```
su admin  
cd /opt/XXXXtop  
ls -al
```

NOTE — If a previous version of the TOPCOM System already exists, the link may have to be deleted and moved to the new directory. *The MYNAH System must be down to do this.*

2.8.3 TOPCOM Configuration

The TOPCOM System must be configured for your environment and applications. Please reference the TOPCOM manual for this information.

2.9 Installing Third-Party GUI-Test Tools — QA/Partner (Optional)

The MYNAH System integrates Segue's QA/Partner to test X Window and MicroSoft Windows GUIs. The following subsections contain the preinstallation steps we recommend for using QA/Partner with the MYNAH System.

2.9.1 QA/Partner Preinstallation Steps

Perform the following steps before installing the QA/Partner software:

1. Become a superuser, i.e., type

```
su root
```
2. Change to the */opt* directory, i.e., type

```
cd /opt
```
3. Create the *XXXXqa* directory, i.e., type

```
mkdir XXXXqa
```
4. Under */opt/XXXXqa*, create the version directory, i.e., type

```
/opt/XXXXqa/4.0.1
```
5. Create the symbolic link pointing to the version directory in */opt/XXXXqa*, i.e., type

```
ln -s /opt/XXXXqa/4.0.1 qapartner
```

2.9.2 Installing and Configuring QA/Partner

To install and configure the QA/Partner software, see the QA/Partner manual.

2.10 Installing the License Keys

Before you use the MYNAH System, you must install the license keys for the MYNAH Packages you want to use. (See [Section 2.3.1](#) for information on obtaining license keys.)

NOTE — It is advisable that you run the license server on the same machine as the **vxIpcDir** process.

Perform the following steps to install the license keys:

1. Put each key on a line by itself in a file called *xmyLicenses*.
2. Place the *xmyLicenses* file in *\$XMYHOME/config*. If *xmyLicenses* already exists and contains other keys, add the new key after the last line in the file. Expired keys should be deleted.
3. Execute the following command to check that port 5093 is not currently being used:

```
netstat | grep 5093
```

If this port is being used, call the MYNAH hotline. (See [Section 2.3.1](#).) Otherwise, edit the */etc/services* file, adding the following line:

```
LicenseServ          5093/udp
```

4. Set and export the variable **LSHOST**, which is the license server machine. Remember to verify/update the value **LSHOST** is set to in the *\$XMYHOME/config/xmyProfile* or *\$XMYHOME/config/xmyLogin* file.
5. Select or deselect license keys by uncommenting or commenting (#) the keys.

2.11 Configuring a Minimal Working Installation

During the MYNAH install process, example configuration files were copied from the `$XMYDIR/examples/config` directory to the `$XMYHOME/config` directory. Those copied files are

- `$XMYHOME/config/xmyConfig`
- `$XMYHOME/config/xmyConfig.General`
- `$XMYHOME/config/xmyConfig.GT`
- `$XMYHOME/config/xmyConfig.TOP`
- `$XMYHOME/config/xmyConfig.OP`

The BAIST post installation that copied these files also attempted to update all information that it could, but these files do need to be verified and possibly updated.

The following steps describe how to customize those files to quickly create a minimal working installation. [Section 3](#) describes the syntax and content of these configuration files in detail and provides information on configuring optional packages, such as for the TOP extension package.

1. Execute

```
su madmin
```

2. Edit `$XMYHOME/config/xmyConfig.General`, then confirm that the `OMPport` entry lists an unused UNIX port number between 15000 and 64000. If the port number is being used by other software (i.e., an entry for it exists in the `/etc/services` file), choose another port number. Add this port number to the `/etc/services` file. For example, if you selected port 15000, enter the line

```
OMPport 15000/udp #operability manager
```

in the `/etc/services` file.

NOTE — If you use an Oracle database with the MYNAH System, edit the **Database**, **WelcomeNewUsers**, and **NonOwnerObjectModification** parameters as follows:

```
Database = yes,  
WelcomeNewUsers = yes,  
NonOwnerObjectModification = true;
```

In addition, you must uncomment out the Trigger Daemon reference in the *xmyConfigOP* file so that they appear as follows:

```
Process xmyTD
    Mynah          = Yes,
    Autostart      = Yes,
    Start          = "xmyStartTD",
    Stop           = "xmyStopTD",
    Status         = "xmyStatusTD";
```

3. If the MYNAH System was not installed in */opt/XXXXmyn/mynah*, edit *\$XMYHOME/config/xmyConfig*, then verify and change all occurrences of the path */opt/XXXXmyn/mynah* to that actual location of the installation.
4. Edit *\$XMYHOME/config/xmyConfig*, then verify the **Host** entries in the **Dispatcher** and **EngineGroup** sections to reflect the name of the machine on which you want the related processes to run. Verify that all path declarations in the file are correct, such as in the following:

```
Dispatcher SD1
    EngineGroups      = (SeGp1, SeGp3),
    DefaultEngineGroup = SeGp1,
    Host              = angels,
    ActivityLogging   = yes;
```

5. Edit *\$XMYHOME/config/xmyConfigOP*, then verify the hostname in the **OperabilityAgent** section to reflect the machine on which you want the related processes to run, such as in the following

```
OperabilityAgent selene
    Responsibilities = (vxDir, vxGateway, vxLogToFile,
                      vxErrorServer, xmyTD, xmyBD, xmySD1,
                      xmyQueues, xmyLS);
```

Once you have performed these steps, you are ready to bring up the system. This involves starting third-party software, such as Oracle, and the MYNAH platform and application processes. See Chapter 4 for complete information on starting and stopping these processes.

2.12 Verifying the MYNAH System

Perform the following to manually verify the system:

1. Execute

```
su - madmin
```

2. Execute

```
cd $XMYDIR/bin
```

3. Set your display to your host, i.e., type

```
export DISPLAY=:0.0
```

4. Start the MYNAH processes by executing

```
./xmyStartUp
```

5. Verify that all MYNAH processes are up. As a minimum, you should see the following processes:

- vxIpcRecvd
- vxErrorServer
- vxLogToFile
- vxIpcDir
- xmyBD
- xmyOA
- xmySD
- xmyEngine
- LServ.

You can verify what processes are up by using the **ps** command and grepping for the processes, as in the following examples:

- For Solaris

```
ps -face | grep madmin | egrep "vx|xmy" | grep -v egrep
```

- For HP-UX

```
ps -fade | grep madmin | egrep "vx|xmy" | grep -v egrep
```

6. Start the Tcl shell by executing

```
./xmytclsh
```

Enter the following lines in the Tcl shell:

```
> xmyLoadPkg TermAsync  
> set conn [xmyTermAsync connect]  
> $conn disconnect  
> exit
```

NOTE — If you need to debug the test for any reason, use either of the following:

```
xmytclsh -d `tty`  
xmytclsh -d filename
```

7. If you have installed an Oracle database, you can test the MYNAH GUI.

NOTE — Before starting the GUI, verify that your `DISPLAY` is set correctly.

Execute

```
./xmyRunMynah
```

The GUI should start up and you should be able to interact with it.

8. If you did not install an Oracle database, you can still use the GUI's Script Builder.

NOTE — Since a database is not used, change the **Database** and **WelcomeNewUsers** entries in the *xmyConfig.General* file to “**NO**”. See [Section 3.3.1](#) for information on the *xmyConfig.General* file and these entries.

Start the Script Builder by executing

```
./xmyRunMynah -b
```

2.13 Installing the On-line Documents

The MYNAH documents are available on-line in the Adobe Acrobat PDF format. The documents are delivered via a tar file archive, *mynah_docs.tar*, which is delivered separate from the MYNAH software BAIST archive. The *mynah_docs.tar* archive contains the PDF files and an HTML web page, which will let users access the PDF files from non-local systems, such as if the MYNAH System is installed on a server and the users must **telnet** to the MYNAH server.

Viewing the PDF file requires that users have installed the Adobe Acrobat Reader. See [Section 2.13.2](#) for information about obtaining the Acrobat Reader.

2.13.1 Installing the PDF Files

Once you receive the documentation archive, execute the following commands to unpackage the **tar** archive:

```
mkdir $XMYDIR/doc
cp mynah_docs.tar $XMYDIR/doc
cd $XMYDIR/doc
tar -xvf mynah_docs.tar
```

A directory called *mynah* is created under *\$XMYDIR/doc* containing the MYNAH documentation in PDF format.

2.13.2 Obtaining the Acrobat Reader

The Acrobat Reader is available directly over the Internet from Adobe at *www.adobe.com*. In addition, the Acrobat Reader is included as a file archive obtained from the MYNAH **ftp** server.

The following versions of the Acrobat Reader are available via the file archive:

- Solaris
- HP-UX
- Microsoft Windows 3.1 (16-bit)
- Microsoft Windows 95 and Microsoft Windows NT (32-bit)
- Macintosh[®].

2.13.2.1 Obtaining the Acrobat Reader as a File Archive

To obtain the Acrobat Reader as a file archive:

1. Obtain the appropriate Acrobat Reader and README files from the FTP site.
2. Install the Acrobat Reader as per the instructions in the README file.

2.13.2.2 Obtaining the Acrobat Reader from Adobe

You can download the Acrobat Reader directly over the Internet from Adobe at www.adobe.com.

2.13.3 Accessing the PDF Files

Once you have installed the Acrobat Reader, users can read the files

- Using the Acrobat Reader directly from *\$XMYDIR/doc/mynah* if they are running the MYNAH System on the system where it was installed.
- Using the Acrobat Reader as plug-in to a browser if the MYNAH System has not been installed on a local system.

In this case, you would move the *mynah* directory to an internal web site. You must install the *mynah* directory in such a way that will ensure that it shall not be accessed over a public, non-secured Internet connection or shared with any third party, such as through an extranet connection.

If the users access the PDF file via a browser, they may wish to download the file to their local system, which will give them direct access to the file the next time they need to read the file, rather than waiting for the browser to load it.

3. Customizing a MYNAH Configuration

Once you've installed the software and performed the tasks detailed in [Section 2.11](#), you will have a minimal working installation. This section describes the syntax and content of the MYNAH configuration files.

3.1 Introduction

The MYNAH configuration files are stored in the directory assigned to the variable `$XMYHOME`. This must be different from `$XMYDIR (/opt/XXXXmyn/mynah)`. For example, if multiple configurations are desired for different user communities of the same MYNAH installation, then multiple `$XMYHOME` locations may be created.

This section covers the two MYNAH configuration files, the `xmyConfig` and `xmyConfigOP` files, that you can edit to customize an installation.

NOTE — If you change configuration values after users have created objects in the database, users may experience problems when accessing objects associated with those changed values.

For example, in an existing release, Script objects are associated with SE group `SeGroup1`. If, in a new release, you remove or change the `SeGroup1` value, all Script objects associated with `SeGroup1` will not run.

3.1.1 The xmyConfig File

The `xmyConfig` file creates the basic configuration values for the MYNAH System, such as creating SEs and SDs and creating the configurations for the various MYNAH packages, such as the TermAsync Package.

When you access the MYNAH System, the system reads the `xmyConfig` file in `$XMYHOME/config` and loads the specified configuration information.

You could create one `xmyConfig` file containing all of the entries for your installation. The delivered `xmyConfig` file, however, follows a modular approach; there is one `xmyConfig` file and several specialized configuration files. When the system starts, it reads the `xmyConfig` file, which loads in the specialized configuration files via

```
%INCLUDE filename
```

statements, where `filename` is the name of a specialized configuration file.

NOTE — The **%INCLUDE** statement first searches for the specified *filename* in the current directory, i.e., where you started the MYNAH System, and then in *\$XMYHOME*.

As delivered, the files that comprise the *xmyConfig* file are as follows:

<i>xmyConfig</i>	This is the main configuration file, and contains the configuration information for the SDs, SEs, SE Groups, and the TermAsync and Term3270 Packages. The other configuration files are loaded into this file using the %INCLUDE statement.
<i>xmyConfig.General</i>	This file contains general configuration information, such as the default SD for this installation and whether this installation uses a database.
<i>xmyConfig.GT</i>	This file contains the configuration information for the various GUI testing tools as well as the information for GUI Test SEs.
<i>xmyConfig.TOP</i>	This file contains the configuration information for the TOP, AppApp, PRT3270, and TCP Packages.

For example, to load the *xmyConfig.TOP* file into the *xmyConfig* file, the line

```
%INCLUDE xmyConfig.TOP
```

appears at the end of the *xmyConfig* file.

This modular design was done for several reasons.

- The configuration information in the *xmyConfig.General* file must be in the *xmyConfig* and *xmyConfigOP* files. Instead of reproducing this information in both files, it is loaded by entering the line


```
%INCLUDE xmyConfig.General
```

 at the beginning of each file. This ensures that the same general configuration information appear in both files.
- If you don't need to use GUI testing tools or the TOP, AppApp, PRT3270, and TCP Packages, you can simply comment out (or remove) the **%INCLUDE** statement for the appropriate file.
- As your installation grows, the *xmyConfig* file can get very large, making it harder to find entries if you have to make changes to the configuration.

In fact, as your system grows, you may find it convenient to make your own specialized configuration files, such as placing all configuration information for the TermAsync Package in one file.

NOTE — If you do this, you must take care where you place the **%INCLUDE** statement. For example, as we will see later, one of the configuration settings you can create for an SE is control over a TermAsync configuration. If an SE has this control, the declaration for the TermAsync configuration must occur prior to the configuration for the SE.

3.1.2 The *xmyConfigOP* File

The *xmyConfigOP* file is used to create Operability Management configuration entries, which gives you a mechanism to start, stop, and get status of all of the MYNAH processes. Information on using the Operability Management can be found in Section 4.

3.2 Environmental Variables

The MYNAH System uses five environment variables: XMYDIR, XMYHOME, XMYSD, XMYSEGROUP, and XMYSUT. The two required variables, XMYDIR and XMYHOME, were defined during installation. The other three can be defined if you determine that you need them.

NOTE — XMYDIR and XMYHOME can not be the same location.

The rest of this section defines these variables and describes when they are used.

NOTE — There are several environmental variables that the MYNAH System uses, such as the location of the Oracle database and Telexel System files. The example profile delivered with the MYNAH System (Appendix B.1.3) lists these variables as well as suggested locations.

3.2.1 XMYDIR

The *\$XMYDIR* directory contains all of the MYNAH executables and libraries.

For a minimal installation, *\$XMYDIR* contains the following:

```
$XMYDIR
  /bin
  /contrib
  /conversion
    /config
  /data
    /tmpl
      /Commands
      /Procedures
  /dbadmin
  /demo
    /data
    /scripts
  /examples
    /3270
    /admin
    /appapp
    /config
    /data
    /dbadmin
    /dce
    /tcl
  /include
  /lib
    /Gui
    /dce
    /qap
    /replay
    /sql
    /tcl
    /xrunner
  /man
    /man1
  /samples
    /dce
  /wsf
    /install
    /procs
```

examples/config This directory contains a MYNAH delivered sample configuration files.

<i>examples/data</i>	This directory contains MYNAH delivered sample data files that are used with the xmyDiff command.
<i>/lib</i>	This directory contains language extension libraries, for example <i>libxmyTermAsync.so</i> .
<i>/lib/tcl</i>	This directory contains MYNAH delivered Tcl procedures.
<i>/lib/gui</i>	This directory contains all of the files necessary for running the GUI.
<i>/man</i>	This directory contains MYNAH delivered man pages.
<i>/contrib</i>	Contains tclhelp / html
<i>/wsf</i>	This directory contains information relative to the BAIST install process.

3.2.2 XMYHOME

The *\$XMYHOME* directory contains all configuration information.

For a minimal installation, *\$XMYHOME* will contain the following:

```
$XMYHOME
  /config
  /conversion
  /data
    /messages
      /PRT3270
      /TOP
      /AppApp
      /TCP
    /sedscripts
    /tagdir
  /lib
  /syslog
  /run
    /bd
    /collector
    /gtse
    /oa
    /se
    /sd
    /td
  /samples
  /scripts
  /syslog
```

NOTE — The X25 directory only exists for HP installations.

<i>config</i>	This directory contains all of the configuration files, which are discussed in the following subsection.
<i>data</i>	This directory contains subdirectories use by various subsystems. For example, the xmyDiff utility uses sed scripts. These are located here under a directory called <i>sedscripts</i> . In addition, this directory contains any required data, and is the default location for tag tables. (See Section 3.3.2.2.)
<i>syslog</i>	This directory contains the system log file that is produced by the Telexel vxLogToFile process.
<i>run</i>	This directory contains a subdirectory for each process type in the system. The processes will actually run in these directories and the directories will contain process log files.
<i>scripts</i>	This directory is optional, it may contain user scripts, and is the default location for the <i>LibraryPath</i> . (See Section 3.3.3.1.)

3.2.3 XMYSD

The XMYSD environment variable tells the CLUI what SD to submit to. In general, when the user submits a script using the CLUI, the CLUI determines the SD using the following order of precedence:

- Command-line (-d option)
- XMYSD environment variable
- database (i.e., the SD associated with the script object in the database)
- The default-SD that appears in the *xmyConfig.general* file. This is the value of the **DefaultSD** parameter in the **General Default** entry.

The CLUI also uses XMYSD in other contexts besides submittal of a script. For example, any time the CLUI interacts with an SD, such as canceling a script or getting information on all users known to the SD, it needs to know which SD to interact with. The same order of precedence applies in these other cases as above, but with the omission of the third item (i.e., database), as it is not relevant in other contexts besides submittal.

3.2.4 XMYSEGROUP

The XMYSEGROUP environment variable tells the CLUI what SE Group the script being submitted should run in. In general, the CLUI determines the SE Group using the following order of precedence:

- Command-line (-e option)
- XMYSEGROUP environment variable
- database (i.e., the SE Group associated with the script object in the database)
- <null>: i.e., if none of the above are specified, the CLUI sends in the NULL string as the name of the SE Group to run in. This tells the SD the script is submitted to use that SD's default SE Group as determined by the **DefaultEngineGroup** parameter in that SD's entry in the *xmyConfig* file.

3.2.5 XMYSUT

The XMYSUT environment variable tells the CLUI what SUT-INFO object the runtime-info object of the script being submitted should be associated with. Its value should always be numeric. In general, the CLUI determines the SUT-INFO object using the following order of precedence:

- Command-line (-S option)
- XMYSUT environment variable
- 2, that is, if the SUT-INFO id doesn't appear either on the command-line or in the environment variable, the CLUI will assume an id of 2, which is guaranteed to exist in the database.

3.3 The xmyConfig File Syntax

The *xmyConfig* file entries are used to

- Create configuration parameters that apply to the entire MYNAH System
- Set configuration parameters (e.g., name and time-out values) for each MYNAH scripting package (domain), e.g., the 3270 Terminal Emulation package
- Create and configure Script Engines (SEs), Script Engine Groups, and Script Dispatchers (SDs) processes.

NOTE — The *xmyConfig* file is the *only* place where you can create SEs, SE Groups, and SDs.

The example configuration files supplied with the MYNAH System may be used directly, but you must provide system specific values for the Term3270 **Host** and **VendorPath** entries before running scripts using the Term3270 Package.

xmyConfig file entries take the format

```
entry_name LogicalName
parameter = option,
parameter = option;
```

NOTE — A *LogicalName* must not begin with a number. For example, *Async* is a valid *LogicalName* but *3270* is not.

Each parameter listing, except for the last one for an entry, is delimited by a comma (.). Each entry is delimited by a semicolon (;).

WARNING — When specifying *option* values for the **parameters**, you must enter the entire pathname; you *cannot* use environmental variables. Where the following descriptions refer to *\$XMYHOME* or *\$XMYDIR*, remember to enter the actual values for these environmental variables when you configure your MYNAH installation.

entry_name must be one of several reserved words, each giving explicit control over a MYNAH configuration function.

NOTE — All **entry_names** are case sensitive; you must enter them exactly as shown in the following list. All **LogicalName**, **parameter**, and **option** names are case insensitive, but any further uses of these names within the *xmyConfig* file must be entered exactly as you created them.

- The **entry_name** and **LogicalName** pair, **General Default**, is used to define all MYNAH System general settings. (See [Section 3.3.1.](#))
- The following **entry_names** are used to define the configurations for the MYNAH scripting packages:

Term3270	Creates entries that define the configuration for the 3270 Terminal Emulation package. (See Section 3.3.2.2.)
TermAsync	Creates entries that define the configuration for the Asynchronous Terminal Emulation package (See Section 3.3.2.1.)
ProtocolHandler	Creates settings used to define for the TOP, AppApp, PRT3270, and TCP packages. (See Section 3.3.2.3.1.)
MsgCollector	Creates the configuration for the TOPCOM, AppApp, PRT3270, and TCP Collector process. (See Section 3.3.2.3.2.)
GuiTool_qap	Creates settings that define the configuration for a QA Partner GUI test tool. (See Section 3.3.2.4.1.)
GuiEngine	Creates the settings for GUI Test Script Engines, which are similar to the standard SEs but only handle GUI vendor tool scripts. (See Section 3.3.2.4.2.)

- The following **entry_names** are used to create and configure the SD, SE, and SE Group processes:

Engine	Defines settings for the Script Engines. (See Section 3.3.3.1.)
EngineGroup	Defines settings for the Script Engine Groups. (See Section 3.3.3.2.)
Dispatcher	Defines settings for the Script Dispatcher. (See Section 3.3.3.3.)

The **LogicalName** argument is used to assign a unique name to an **entry_name**, creating a specific configuration for an **entry_name**, such as creating a name for an SD. You can use **LogicalNames** to create multiple sets of unique configurations. For example, you can create several 3270 configurations that use different ports or have different timeout values.

In addition, there are two reserved *LogicalNames* (*Standalone* and *Embedded*) that are used by the **Engine** process. These reserved names are discussed in [Section 3.3.3.1](#)

NOTE — *LogicalNames* are limited to 21 characters.

The following sections describe the **parameters** (and the valid **option** values for each **parameter**) for each of the **entry_names**. Each section contains an example entry. An example of an entire *xmyConfig* file is shown in [Figure 3-13](#).

WARNING — If a **parameter** is an **entry_name**, the **entry_name** must have been defined prior in the file. For example, when configuring an SE, you can specify a 3270 or asynchronous configuration for that SE. These **TermAsync** and/or **Term3270** parameters entries must be defined prior to the **Engine** entry.

NOTE — All required parameters will be indicated.

3.3.1 General Configuration - `xmyConfig.General`

The `entry_name` and *LogicalName* pair **General Default** defines entries that apply to the MYNAH System. An example `xmyConfig.General` entry is shown in [Figure 3-1](#).

NOTE — The **General Default** pair is the only valid entry allowed for the General entry.

```
General Default
  DefaultSD           = SD1,           # required
  Database            = no,           # "yes" if Oracle is not used
  WelcomeNewUsers    = no,           # "yes" if Oracle is not used
  NonOwnerObjectModification = false, # true or false
  OMPort              = 15000;        # required
```

Figure 3-1. `xmyConfig` General Entry

NOTE — This information is loaded into the `xmyConfig` and `xmyConfigOP` files by saving it to the `xmyConfig.General` file and entering the line

```
%INCLUDE xmyConfig.General
```

in each file.

The valid **General** configuration parameters are

DefaultSD	Specifies the default SD for the system. This is a required parameter.
Database	Indicates whether or not this configuration of the MYNAH System makes use of a database. The database is required if users need to use the MYNAH test management abilities. yes Use a database no Do not use a database If no database is used, the ability to save scripts to the MYNAH database and full MYNAH GUI functionality to create test cases is not supported. Users will, however, be able to use their own editor or the MYNAH Script Builder to create scripts to automate tasks. The Script Builder can be brought up independently of the full MYNAH GUI by typing <code>xmyRunMynah -b</code>

Users will be able to submit scripts for automating tasks using the Script Builder or submit them to the Background Execution Environment (BEE) by using the CLUI.

Note — If you run the MYNAH System with the database, in addition entering *yes* for this parameter, you must uncomment the TD references in the *xmyConfigOP* file as follows:

```
Process xmyTD
  Mynah      = Yes,
  Autostart  = Yes,
  Start     = "xmyStartTD",
  Stop      = "xmyStopTD",
  Status    = "xmyStatusTD";
```

Note — If you choose to run the MYNAH System without the database, the following will occur:

- All SEs will automatically be SE Lites.
- The TD should not be used.

Default = no

WelcomeNewUsers

Indicates whether or not the GUI will automatically create a Person object for a new user (i.e., a user that does not exist yet in the database).

- yes** The GUI will create a Person object for the new user.
- no** The GUI will notify the user that they are not an authorized user of the MYNAH System, and the GUI will exit. However, the users will be able to use the Script builder by typing **xmyRunMynah -b**.

Default = no

NonOwnerObjectModification Specifies whether a user has the ability to modify other people's objects in the MYNAH GUI. If this parameter is set to false, only the owner of an object or an administrator will be able to edit the object.

Note — All users will still be able to open the object in read-only mode.

The ability to open Test Hierarchies for editing purposes is not affected by the setting of this configuration tag; non-owners and non-administrators can open Test Hierarchies for editing even if **NonOwnerObjectModification** is set to *false* .

Default = true

OMPort Specifies the port number used by the OA processes to communicate with the **xmyOM start**, **stop**, and **status** methods. This must be set to a valid unused port with a number greater than 5000 and less than 65000.

Refer to the */etc/services* files to determine what ports are unused.

This is a required parameter.

3.3.2 MYNAH Package Configuration

The following sub-sections provide information about the configuration of each of the MYNAH Packages. These Package entries are used by the **Engine** entries (Section 3.3.3.1) to specify information about entries for the packages. As mentioned earlier, using unique **LogicalNames**, a package can have multiple entries letting you define multiple sets of entries for a package and different **Engines** will make use of them.

For each package, the values defined in the configuration file will be assigned to the class command for each package, e.g., **xmyTerm3270**. These values can be reset within the script by using the class level attributes.

NOTE — Remember, never use a number as the first character of a **LogicalName**.

3.3.2.1 Asynchronous Configuration - TermAsync

The **TermAsync entry_name** contains entries for the Asynchronous Terminal Extension Package. An example **TermAsync** entry is shown in Figure 3-2.

```
TermAsync Async_type_1
  Terminal           = vt100,
  Timeout            = 300,
  ShowAttributes     = No,
  Shell              = /bin/sh,
  BufferSize         = 4096,
  AuxTerminfo       = .terminfo;
```

Figure 3-2. xmyConfig TermAsync Entry

The valid **TermAsync** configuration parameters are

Terminal	Specifies the default terminal being emulated. Default = vt100
Timeout	Defines the number of timeout seconds for the wait and sendWait command. Default (seconds) = 60
ShowAttributes	Determines whether to write the application's screen attribute bytes to the <i>SUTimages</i> file. yes Write to the <i>SUTimages</i> file no Don't write attributes to the <i>SUTimages</i> file. Default = no

Shell	Specifies the shell used to connect with the SUT. Default = <i>/bin/sh</i>
BufferSize	Specifies the size of the internal buffer (in kilobytes) that stores SUT responses. Default = 4096
AuxTerminfo	Specifies the name of the auxiliary <i>terminfo</i> file. An auxiliary <i>terminfo</i> file describes the capabilities/escape sequence mapping for <i>one</i> terminal. If you need more terminals, you must create a file for each terminal. The syntax for the auxiliary <i>terminfo</i> file entry lines are <CAPABILITY>=<ESCAPE SEQUENCE> where <ul style="list-style-type: none">• <CAPABILITY> is the name of the capability and follows the standard <i>terminfo</i> naming. All vt100 capabilities (see infocmp -I vt100) are currently supported.• <SEQUENCE> is the escape sequence. The sequence can be specified using TERMCAP syntax, which is simpler than <i>terminfo</i>. A line contains at most one mapping. A line beginning with a # is considered a comment. The MYNAH System first searches for the specified auxiliary <i>terminfo</i> file in the current directory and then in <i>\$XMYHOME</i> . See Appendix F for information relating to the auxiliary <i>terminfo</i> file, such as escape sequences supported terminal capabilities.

3.3.2.2 3270 Configuration - Term3270

The **Term3270 entry_name** contains entries for the Synchronous 3270 Terminal Extension Package. An example **Term3270** entry is shown in [Figure 3-3](#).

```
Term3270 Term3270_type_1
Host                = <"host_1, host_2">,          # required
Model               = 2,                            # optional
Port                = 23,                            # optional
CompareInvisibleFields = No,                        # optional
ScreenIdentificationFile = "",                      # optional
TN3270E             = No,                            # optional
ShowAttributes      = No,                            # optional
TagDir              = /opt/XXXXmyn/mynah_home/data/tagdir, # optional
Timeout             = 300,                            # optional
InitialWait         = No,                            # optional
InitialWaitExpect   = "",                            # optional
CollectKeyCount     = No,                            # optional
VendorPath          = "/opt/XXXXioc/ioconcepts",    # required
UnderlineUnprotectedFields = Yes;                  # optional
TranslatePath       = ""                             # optional
```

Figure 3-3. xmyConfig Term3270 Entry

The valid **Term3270** configuration parameters are

Host	<p>Specifies the name of the 3270 host machine that the connections are to connect to. Since connections will be driven over TCP/IP, this may often be IP addresses, unless the user environment makes use of alias hostnames to IP addresses.</p> <p>You can specify more than one host, where the host names are enclosed within double quotes and separated by spaces, for example</p> <pre>host_1 = "pyibm1.telcordia.com pyibm2 \ 128.96.250.102"</pre> <p>This is a required parameter.</p>
Model	<p>Specifies the default type of terminal model.</p> <p>Valid types are 2, 3, 4 or 5.</p> <p>Default = 2.</p>
Port	<p>Specifies the default IP port for sessions to connect through. Typically, the port is set as a standard of telnet or port 23. Some installations may set aside special port numbers for MYNAH access to their host machines and will need to be able to specify this port number.</p> <p>Default = 23.</p>

CompareInvisibleFields	<p>Indicates whether invisible fields should be processed by statements done to a particular connection.</p> <p>yes Process</p> <p>no Don't process.</p> <p>Default = yes.</p>
TN3270E	<p>Determines whether sessions should begin in TN3270E mode, meaning they will be connecting to a TN3270E server and will be establishing handshakes through the TN3270E protocol.</p> <p>yes Expect TN3270E</p> <p>no Don't expect TN3270E.</p> <p>Default = no.</p>
ShowAttributes	<p>Determines whether to write the application's screen attribute bytes to the <i>SUTimages</i> file.</p> <p>yes Write to the <i>SUTimages</i> files</p> <p>no Don't write attributes to the <i>SUTimages</i> file.</p> <p>Default = no.</p>
TagDir	<p>Specifies the directory containing the tag name files, which are used to translate tagname name locators to row/column positions.</p> <p>Default = <i>\$XMYHOME/data/tagdir</i>.</p>
TranslatePath	<p>Specifies the location of the EBCDIC-ASCII translation file, which is used to define the presentation space while making a 3270 connection. See Section 3.6.4 for information on how to override default ASCII-EBCDIC translations.</p> <p>Note — You can override this specification by using the Term3270 connect method's -XLT attribute. See the <i>MYNAH System Scripting Guide</i> for information on the Term3270 connect method.</p> <p>Default = " "</p>

ScreenIdentificationFile	<p>Specifies the location and name of the screen identification file, which identifies each format/screen in the applications(s) to be tested.</p> <p>This must be a valid pathname.</p> <p>You may enter a null string ("").</p>
Timeout	<p>Determines the amount of time, in seconds, a waiting call should wait for a response from the SUT before quitting or executing a timeout handler.</p> <p>Default (seconds) = 300.</p>
InitialWait	<p>Determines whether a session should initially expect data from the SUT at connection time.</p> <p>yes Expect transmission</p> <p>no Don't expect transmission.</p> <p>Default = yes.</p>
InitialWaitExpect	<p>Specifies the string expression that the InitialWait should wait for from the host.</p> <p>Note — This parameter is important for synchronizing with the first screen after the connection is made. For example, if you do not specify a value, the script does not wait until all of the first screen is received; the subsequent commands get executed before an initial screen is fully displayed.</p> <p>The InitialWaitExpect value should be a string that appears towards the bottom of the screen so that you can be sure that the screen is fully displayed before typing begins. For example, presume you want to access a region and the first field where you can enter a region code appears in row 21. Therefore, you want the script to wait until the screen is displayed so that typing can begin in row 21. In this case, any string on row 20 should be good candidate. So, while the default is an empty string, the administrator should assign a meaningful string that appears towards the bottom of the screen.</p> <p>Default = "" (empty string).</p>
VendorPath	<p>Indicates the path to where the 3270 emulator being used by the MYNAH System is located.</p> <p>I/O Concepts™ example: <i>/opt/XXXXioc/iocinst</i></p>

CollectKeyCount	Indicates whether to collect a count for all Function Keys pressed for all 3270 connections under an SE. yes Collect the count no Don't collect the count. Default = no.
UnderlineUnprotectedFields	Specifies if the unprotected fields on the screen are to be underlined. yes Underline unprotected fields no Do not underline unprotected fields. Default = yes.

NOTE — If you specify multiple hosts, the MYNAH System uses the first name on the list when trying to establish a connection. If the connection is successful, the remaining specified host names are ignored. If the connection fails for any reason, the MYNAH System attempts to use each successive host name until a successful connection is made or the end of the list is reached. Upon successful connection, the return will be a handle to the **xmyTerm3270** instance. For each failed connection, an error message, with the hostname, is generated. If no connection can be established with any host from the list, then the error message *“Unable to establish connection to any host”* is generated.

3.3.2.3 TOP/AppApp/PRT3270/TCP Configuration

TOP, AppApp, and TCP configurations require at least one **ProtocolHandler** for each installation and at least one **MsgCollector** entry.

PRT3270 configurations may have only one **ProtocolHandler** per machine (due to its usage of shared memory) and more than one **MsgCollector** entry.

NOTE — In the delivered example configuration files, the entries explained in this section are saved to the *xmyConfig.TOP* file and loaded into the *xmyConfig* file by including the following line in the *xmyConfig* file:

```
%INCLUDE xmyConfig.TOP
```

3.3.2.3.1 Protocol Handler Configuration

The **ProtocolHandler** contains the entries used to define the TOP, AppApp, PRT3270, or TCP installations. Example **ProtocolHandlers** for TOP, AppApp, PRT3270, and TCP installations are shown in [Figure 3-4](#).

```

ProtocolHandler topcom_1
    Protocol           = TOP,
    TopQueue           = "a:/etc/motd",
    TopSendSession     = 11,
    TopRecvSession     = 11,
    Timeout            = 60,
    ConversionMode     = "",
    ListenMode         = "MSG_LISTEN_NOW",
    MatchProcedure     = "",
    MaxMsgs            = 10,
    MaxSegmentLen      = 65535,
    TopDefaultDTN      = "",
    TopDefaultPSN      = "";

ProtocolHandler printer_1
    Protocol           = PRT3270,
    TopQueue           = "",
    TopSendSession     = 0,
    TopRecvSession     = 0,
    Timeout            = 300,
    ConversionMode     = "",
    ListenMode         = "MSG_LISTEN_NOW",
    MatchProcedure     = "",
    MaxMsgs            = 5;

ProtocolHandler app_1
    Protocol           = AppApp,
    Host               = cricket.base.telcordia.com,
    TcpPort            = 6542,
    Timeout            = 31,
    ListenMode         = "MSG_LISTEN_SEND",
    MatchProcedure     = "",
    MaxMsgs            = 10;

ProtocolHandler tcp_1
    Protocol           = TCP,
    Timeout            = 31,
    ListenMode         = "MSG_LISTEN_SEND",
    MatchProcedure     = "",
    MaxMsgs            = 10;

```

Figure 3-4. Example xmyConfig ProtocolHandler Entries

NOTE — A MsgCollector can have a maximum of 25 TOPCOM ProtocolHandlers.

The valid **ProtocolHandler** configuration parameters are

Protocol	<p>Specifies what protocol (i.e., TOP, AppApp, PRT3270, or TCP) this configuration entry is for. Valid values are TOP, AppApp, PRT3270, or TCP.</p> <p>Note — The name of the ProtocolHandler for the PRT3270 protocol should be the same as the one configured in TN3270 Printer emulator config file. e.g., <i>printer_1</i> in this example. See Section 2.7.2 for details.</p> <p>This is a required parameter.</p>
TopQueue	<p>Specifies the TOP handler IPC message queue key.</p> <p>This is a required parameter for TOP.</p>
TopRecvSession	<p>Specifies the TOP receive session number.</p> <p>This is a required parameter for TOP.</p>
TopSendSession	<p>Specifies the TOP send session number.</p> <p>This is a required parameter for TOP.</p>
Host	<p>The host on which the Interface Collector is executing. The value may be a DNS name or an IP address.</p> <p>Note — If the Message Collector is handling a Printer protocol handler, then the host on which the Message Collector runs should be the same as the one on which the TN3270 Printer emulator process runs. e.g., the iocluprt process for I/O Concepts TN3270.</p> <p>This is a required parameter for AppApp.</p>
TcpPort	<p>The specific port on the Host on which the Interface Collector listens.</p> <p>This is a required parameter for TCP.</p>
Timeout	<p>Specifies the seconds to wait for receiving a SUT message or send back.</p> <p>This is a required parameter for TOP, AppApp, PRT3270, or TCP.</p>
ConversionMode	<p>Valid values are "" (empty string, which means that no conversion is performed), MSG_TCIS, MSG_TCIS2, MSG_EWNL.</p> <p>Default = "".</p>

ListenMode	<p>Valid values for TOP , AppApp, and TCP are MSG_LISTEN_NOW, MSG_LISTEN_SEND, MSG_LISTEN_NEXT, or digits.</p> <p>Valid values for PRT3270 are all TOP values excluding MSG_LISTEN_SEND.</p> <p>Default = MSG_LISTEN_NOW</p>
MatchProcedure	<p>Tcl match procedure for all received messages. Messages that do not satisfy the match procedure will not be returned. By default a match procedure is not defined, and all messages can be returned by the receive method.</p> <p>For information on how to define a match procedure, see the -match entries in the following <i>MYNAH System Scripting Guide</i> Tcl Language Extensions sections:</p> <ul style="list-style-type: none"> • General Application-to-Application (Section 12.2.2.9) • TOP (Section 13.2.2.8) • PRT3270 (Section 14.2.2.7) • TCP Application-to-Application (Section 17.2.2.10) <p>Procedures must be in the proc repository as members of a TLIB library file.</p> <p>Default = "" (empty string)</p>
MaxMsgs	<p>Maximum number of appended messages used by xmyMsgMatchUntil().</p> <p>Default = 10</p>
MaxSegmentLen	<p>Segment size (in bytes) used only in TCIS and TCIS2 conversions.</p> <p>Default = 65535</p> <p>Note — This value should be changed by the use to match the message segment size appropriate for the user's application.</p>
TopDefaultDTN	<p>TOP Destination Transaction Name.</p> <p>Default = ""</p>
TopDefaultPSN	<p>TOPCOM Presentation Services Name.</p> <p>Default = ""</p>

3.3.2.3.2 Message Collector Configuration

The **MsgCollector** entry_name defines the configuration for a process called a **xmyCollector**, which stores all incoming messages into a special area (the **Message Response Directory**) on disk. The Collector is the interface between the MYNAH System and TOPCOM, sending and receiving messages from TOPCOM or any other protocol handler. All SEs performing a receive operation access the disk in order to get the message they are interested in. Whether or not you are concerned with checking received messages, the collector must be running.

NOTE — **MsgCollector** names must begin with an alphabetic character; names beginning with numbers are not supported.

```
MsgCollector cricket_A
  MessageDirectory      = "",
  TableSize            = 500,
  Port                 = 2010,
  Host                 = <hostname>,
  Handler              = (topcom_1, topcom_2, printer_1, tcp_1);
```

Figure 3-5. Example xmyConfig MsgCollector Entry

All of the following **MsgCollector** configuration parameters are required:

MessageDirectory	Specifies the full path of the directory where the received messages are written. If you enter an empty string (""), the default path is used. Default = <i>\$XMYHOME/data/messages</i>
-------------------------	--

TableSize	<p>Specifies the number of received messages being tracked by the collector. If you enter TableSize = 0, the default value is used.</p> <p>NOTE — The TableSize parameter specifies the size of an internal table that is used by the collector to keep track of received messages. Messages that are not in the internal table are not known by the collector, and therefore can not be received by the application. If the number of received messages is larger than TableSize, only the most current messages are tracked by the collector.</p> <p>To determine the size of this parameter, you need to know how and when the application processes the received messages. If the application processes and removes the received messages as they are received by the collector, then the default size can be used. If the application normally does not process and remove the received messages as they are received by the collector (i.e., it lets received message accumulate in the message directory), then a larger TableSize value is needed. The best solution is to remove all processed messages from the message directory.</p> <p>Default = 500</p>
Port	<p>Specifies the Internet port of main collector. This must be a valid unused port between 1024 and 65000. Check the <i>/etc/services</i> files to find what ports are being used.</p> <p>If it is out of valid range, it defaults to 2010.</p>
Host	<p>Specifies the hostname of the machine where the collector is running.</p> <p>Note — If the Message Collector is handling a printer protocol handler, then the host on which the Message Collector runs must be the same as the one on which the TN3270 printer emulator process runs e.g., the process iocluprt for I/O concepts TN3270.</p>
Handler	<p>Specifies the list of ProtocolHandlers served by this collector.</p> <p>NOTE — You can configure only one PRT3270 ProtocolHandler per machine. If multiple printer/collectors are required, then each must be on a separate host machine.</p>

The MYNAH Message Collector must be started after the TOPCOM process has started. If TOPCOM is restarted, the Message Collector must be restarted. If this procedure is not followed, an error will occur during script execution, outputting the message

```
xmyTop send: Handler () - send session (1) not in TSOPEN state.
```


It is preferable, for permissions/writing purposes, that TOPCOM and the Message Collector be started by the same user id. If this is not the case, then the **umaskn** for the Message Collector starting ID must grant permission for TOPCOM to write in the Message Collector directories or the user starting Message Collector must execute

```
chmod 777
```

on all of the Message Collector sub-directories. If this is not done, TOPCOM will not be able to write the received data, causing errors and shutting down the session.

3.3.2.4 GUI Test Configuration

Each GUI test tool requires a separate configuration entry as well as an engine, much like a standard SE. However, this engine only handles GUI vendor tool scripts.

To execute GUI vendor scripts, use the CLUI's **xmyCmd submit** command and send the script to a GUI Test Script Engine (GTSE).

NOTE — In the delivered example configuration files, the entries explained in this section are saved to the *xmyConfig.GT* file and loaded into the *xmyConfig* file by entering the following line to the *xmyConfig* file.

```
%INCLUDE xmyConfig.GT
```

3.3.2.4.1 QA Partner GUI Test Tool Configuration

The **GuiTool_qap entry_name** contains entries that define the configuration for a QA Partner GUI test tool. An example **GuiTool_qap** entry is shown in [Figure 3-6](#).

```
GuiTool_qap QAP1
  Command      = partner,
  Home         = /opt/XXXXqa/qapartner,
  Bin          = partner/bin,
  Lib          = partner/lib,
  LicenseFile  = partner/FLEXlm/license.dat,

#   OPT_TRACE = 1,
#   # other QA Partner configuration options
#   # see QA Partner documentation

  Debug       = TRUE;
```

Figure 3-6. Example GuiTool_qap Entry

The valid QA Partner **GuiTool_qap** configuration parameters are

Command	Defines the command to run. The default value is <i>partner</i> . This is a required parameter.
Home	Defines the home of QA Partner installation. (This is the SEGUE_APPS environment variable.) This is a required parameter.
Bin	Defines the location of the directory containing the QA Partner executables. This location is added to the PATH environment variable. This is a required parameter.
Lib	Defines the location of QA Partner shared libraries. (This location is added to the LD_LIBRARY_PATH environment variable). This is a required parameter.
LicenseFile	Defines the file prepended to the LM_LICENSE_FILE environment variable.
Debug	Determines if debugging information is saved. TRUE Turns on QAP-MYNAH debugging output. FALSE Partner-specific GTSE process files are deleted at the end of execution. This is the default.
IniFile	Specifies a file containing configuration options for seeding the QAP configuration. Any QA Partner configuration parameters specified in this MYNAH configuration file overrides the corresponding parameter specified in the users <i>IniFile</i> .
AgentPort	Specifies the TCP/IP port for partner-agent communication. Default = 7654.
MynahPort	Specifies the TCP/IP port for MYNAH-QAP communication. Default = 5555.

In addition to the parameters listed above, many QA Partner configuration options may be added to the **GuiTool_qap** entry. An example QA Partner configuration parameter is OPT_TRACE = 1. Specifying this parameter in the MYNAH *xmyConfig* file overrides the corresponding configuration in the QA Partner *IniFile*.

A list of the available options can be found in Appendix D. See the QA Partner documentation for the use of these options.

The tag for each option is simply the name of the option, e.g., `OPT_WINDOW_TIMEOUT`.

The user's window manager must be configured for "click-to-type" behavior and the X resources in Figure 3-7 must be added to the user's `.Xdefaults` file.

```
Mwm*autoKeyFocus:      True
Mwm*deiconifyFocus:    True
Mwm*interactivePlacement: False
Mwm*keyboardFocusPolicy: explicit
Mwm*raiseKeyFocus:     True
Mwm*startupKeyFocus:   True
```

Figure 3-7. QA Partner X Resources

3.3.2.4.2 GUI Test Script Engine (GTSE) Configuration

While the GTSEs are similar to the standard SEs (see Section 3.3.3.1), they only handle GUI vendor tool scripts. The configuration for GTSEs is performed via `GuiEngine` parameters, and you must create a separate **GuiEngine** GTSE entry for each windowing system or tool you use. Figure 3-8 shows an example of a GTSE for QA Partner testing a Windows 3.1 GUI on a PC, for example.

```
# GUI Test Script Engine (GTSE) for QA Partner connecting to Win 3.1 agent
# for testing Win 3.1 GUIs
GuiEngine qapWin31
    Type           = qap,
    Package        = QAP1,
    Sut            = sut_1,
    Displays       = (<hostname>:0.0),
    Agents         = (klehrpc.mac259:5500),
    UseVirtualDisplays = FALSE,
    Debug         = TRUE;
```

Figure 3-8. Example `GuiEngine` GTSE Configurations

The valid **GuiEngine** entries are

Type	Defines the type of vendor package. The only valid value is qap .
Package	Specifies the name of package (i.e., configuration specifications) for a GUI test tool. Enter the <i>LogicalName</i> you created for the GuiTool_qap entry_name .
Displays	Specifies a list of X11 displays to use for running the GUI each instance of the GTSE is given one display to use.

Agents	<p>Specifies a list of agents to perform testing on.</p> <p>Each entry is of the form <i>agentHost:agentPort:mynahPort</i>, where</p> <ul style="list-style-type: none">• <i>agentHost</i> is the hostname of the agent,• <i>agentPort</i> is the TCP/IP port of the agent-partner communication• <i>mynahPort</i> is the TCP/IP port of the MYNAH-QAP communication <p>Any entry may be empty; if <i>agentHost</i> is LOCAL, it uses the current X11 display.</p>
UseVirtualDisplay	<p>Specifies whether to create X11 displays.</p> <p>TRUE Create virtual X11 displays</p> <p>FALSE Do not create X11 displays (default).</p>
Debug	<p>Specifies whether to create debugging output</p> <p>TRUE Turn on certain debugging output.</p> <p>FALSE GTSE process files are deleted at the end of execution (default).</p>

Figure 3-9 shows a complete example *xmyConfig.GT* file.

```
# This is an example configuration file for the Gt domain.

# Configuration for QA Partner

GuiTool_gap QAP1
  Command      = partner,
  Home         = /opt/XXXXqa/qapartner,
  Bin          = partner/bin,
  Lib          = partner/lib,
  LicenseFile  = partner/FLEXlm/license.dat,

  OPT_USE_FILES = $HOME/pub/textedit.inc,
  # other QA Partner configuration options
  # see QA Partner documentation

  Debug       = TRUE;

# Similar configuration for QA Partner with different OPT_USE_FILES setting

GuiTool_gap QAP2
  Command      = partner,
  Home         = /opt/XXXXqa/qapartner,
  Bin          = partner/bin,
  Lib          = partner/lib,
  LicenseFile  = partner/FLEXlm/license.dat,

  OPT_USE_FILES= $HOME/exercises/textedit.inc,

  Debug       = TRUE;

# GUI Test Script Engine (GTSE) for QA Partner connecting to Win 3.1 agent
# for testing Win 3.1 GUIs
GuiEngine qapWin31
  Type          = qap,
  Package       = QAP1,
  Sut           = sut_1,
  Displays      = (<hostname>:0.0),
  Agents        = (klehrpc.mac259:5500),
  UseVirtualDisplays = FALSE,
  Debug        = TRUE;

# GTSE for QA Partner connecting to X11 agent
# for testing X11 GUIs
GuiEngine qapX11
  Type          = qap,
  Package       = QAP2,
  Sut           = sut_1,
  Displays      = (<hostname>:0.0),
  UseVirtualDisplays = FALSE,
  Debug        = TRUE;
```

Figure 3-9. Example *xmyConfig.GT* File (Sheet 1 of 2)

```

# sample EngineGroup for each GTSE above
# Note that these EngineGroups use GuiEngine instead of Engine

EngineGroup GuiGroup1
    GuiEngine    = qapWin31,
    Host         = <hostname>,
    Max          = 1;

EngineGroup GuiGroup2
    GuiEngine    = qapX11,
    Host         = <hostname>,
    Max          = 1;

# Script Dispatcher configuration is the same, regardless of whether
# or not it contains GTSEs

Dispatcher GuiDispatcher
    EngineGroups    = ( GuiGroup1, GuiGroup2),
    DefaultEngineGroup = GuiGroup1,
    Host            = <hostname>,
    Resources       = no,
    ActivityLogging = yes;

```

Figure 3-9. Example xmyConfig.GT File (Sheet 2 of 2)

3.3.2.4.3 MYNAH System Include Files and Accessing the Symbol Table

The MYNAH System is shipped with scripts that let the GUI test tools access the MYNAH Symbol Table, which is important for data flow between SEs.

To use these scripts, include the line

```
use "xmy.inc"
```

in your GUI test script.

You can use the general MYNAH Tcl extensions (**xmySymTblExists**, **xmySymTblGet**, and **xmySymTblPut**) to manipulate the Symbol Table. In addition, you can use **xmyMYNAHIsRunning** to determine if the GUI tool is running under MYNAH control

NOTE — For information on the Symbol Table and the general MYNAH Tcl extensions, see Section 8 of the *MYNAH System Scripting Guide*.

The following text lists the extensions (and their syntax) you must enter in your GUI test script to manipulate the Symbol Table.

```

BOOLEAN exists = xmySymTblExists("SymbolName")
STRING value = xmySymTblGet("SymbolName")
xmySymTblPut("SymbolName", "value")
BOOLEAN mynah = xmyMYNAHIsRunning()

```

3.3.3 Defining Script Engines (SEs), Script Engine Groups, and Script Dispatchers (SDs)

These **entry_names** are used to define and configure SEs, SE Groups, and SDs.

NOTE — Remember, the *xmyConfig* file is the only place where you can define SEs, SE Groups, or SDs.

3.3.3.1 Script Engine Configuration - Engine

The **Engine entry_name** contains entries for the SEs. As mentioned earlier, there are two *LogicalNames* that are reserved for the **Engine entry_name**. The reserved **Engine LogicalNames** are

- Standalone** Defines the configuration for SEs that are started with the **xmytclsh** tool.
- Embedded** Defines the configuration for SEs that are started from within the MYNAH GUI.

User created *LogicalNames* define the configuration for SEs that will be started as part of an SE Group. Groups are used for efficient background execution of scripts. See Section 1.2.8 for a discussion on MYNAH Background Processes.

NOTE — The MYNAH System does not impose a limit on the number of SEs that will run on a host. However, system resources (such as swap space) will impose an upper limit on the number of SEs that can run on each machine. Therefore, if your installation requires a large number of SEs, you should spread the SEs across several machines.

This may require defining additional SE Groups (Section 3.3.3.2) since all SEs in a particular SE Group run on a single host.

Example **Engine** entries are shown in [Figure 3-10](#).

```

Engine Embedded
  StartupScript = "", # optional
  LibraryPath = /opt/XXXXmyn/mynah/lib, # optional
  ProcRepository= "/opt/XXXXmyn/mynah/lib/tcl", # optional
  OutputLevel = ( error, user, script, compare,
                 summary, sutimage), # optional
  OutputBackup = 1, # optional
  OutputPath = /opt/XXXXmyn/mynah/embedded, # optional
  Term3270 = Term3270_type_1, # optional
  TermAsync = Async_type_1;

Engine script_engine_1
  Mode = Stateless,
  StartUpScript = "",
  ExecScript = /opt/XXXXmyn/mynah/demo/scripts/wrapper.tcl,
  LibraryPath = "/opt/XXXXmyn/mynah/lib",
  ProcRepository = "/opt/XXXXmyn/mynah/lib/tcl",
  OutputLevel = ( error, childscr, compare,
                 script, summary, sutimage,
                 testobj, user ),
  OutputBackups = 2,
  TermAsync = Async_type_1;

```

Figure 3-10. xmyConfig Engine Entries

NOTE — Only one occurrence of each MYNAH Package, e.g., a TermAsync configuration, may be defined for an SE.

The valid **Engine** configuration parameters are

Mode Determines the runtime state of the SE. Valid values are the SE execution modes (**Stateless**, **FullState**, and **ConnOnly**) listed in [Section 1.5.2](#).

Default = **Stateless**

NOTE — SEs for scripts run from **xmytclsh** and the Script Builder are always in **FullState** mode, regardless of the *xmyConfig* file setting, since they run individual script lines, not entire scripts.

StartupScript Specifies the path and name of a Tcl script that will be executed each time a new Tcl interpreter is started in a Script Engine. (In **ConnOnly** and **StateLess** mode, this is each time a script is submitted for execution). The start-up script is guaranteed to be the first script executed. An example would be a wrapper script that is used to run third party scripts.

NOTE — This entry is not used by the Standalone engine. It is ignored if you use it with the Standalone engine.

The start-up script can contain any Tcl command. Typically, it will load extension packages and can append to the **auto_path** variable; it should not override the variable.

NOTE — The **auto_path** variable lists the directories containing libraries of Tcl procedures. The **auto_path** variable is initially set to the value of **ProcRepository** if that entry appears in the MYNAH configuration file. If not, it defaults to `$XMYHOME/lib/tcl:$XMYDIR/lib/tcl`

For example, if you've defined an exit handler that you want all SEs in a given group to execute, you would put this line in a start-up script

```
set xmyVar(ExitHandler) myExitHandler
```

If the start-up script fails to execute, the SE will come down.

NOTE — Since the symbol table is overwritten on an incoming execution request, the start-up script should not be used to set symbol table variables. (See **xmySymTblPut**, Section 8.2.23 of the *MYNAH System Scripting Guide*.)

ExecScript Specifies the name of a MYNAH wrapper script (i.e., a script that is passed to third party software) used to run non-MYNAH scripts. This options gives you the ability to integrate other testing tools into the MYNAH System. See [Section 6](#) for more information.

LibraryPath Specifies the location of scripts used by the **xmySE send** and **xmySE sendWait** commands.

You can specify several directories, delimiting them with colons, as in the following example:

```
LibraryPath = "/opt/mynah/SUNWmyn/mynah/lib:\n/u/mgt02/LIB:/mynah/SunOS5.4/5.2/scripts:\n/mynah/SunOS5.4/5.2/scripts/stt"
```

Default = `$XMYHOME/scripts`

Key	<p>Contains a scrambled or unscrambled key to be used to decrypt user database files. (See the xmyCmd scramble subcommand entry in the <i>MYNAH System Users Guide</i> and the <i>MYNAH System Scripting Guide</i>.)</p> <p>This key sets the default value for the -key arguments to the xmyUdb command. (See Section 3.1.2.1 of the <i>MYNAH System Scripting Guide</i> for information on the xmyUdb command.)</p> <p>Note — xmyCmd scramble scrambles the decryption key used by the des system software, which is part of the Sun Encryption Kit. (See Section 1.4.8 for information on the required version of the Sun Encryption Kit.)</p>
ProcRepository	<p>Specifies a series of directories that contain Tcl procedure libraries. The value of ProcRepository is used to set the Tcl auto_path variable in a running SE.</p> <p>Tcl procedure libraries are located in two directories. Delivered MYNAH Tcl procedures are stored in <i>\$XMYDIR/lib/tcl</i>. Users can build up their own libraries of procedures in <i>\$XMYHOME/lib/tcl</i>. Procedures must be placed in one or more files in this directory. You must then create an index file for each file that contains procedures. See Section 3.6.2 for information on creating this index.</p> <p>Note — All files inside the directories specified in the ProcRepository “PATH” must have <i>.tlib</i> appended to their names. Only the <i>*.tlib</i> files from these directories are read into memory.</p> <p>You can specify several directories, delimiting them with colons, as in the following example:</p> <pre>ProcRepository = "/opt/SUNWmyn/mynah/lib/tcl:\ /u/user/lib/tcl"</pre> <p>Default = <i>\$XMYDIR/lib/tcl:\$XMYHOME/lib/tcl</i></p> <p>Note — This processing reads all procedures into memory. If a particular procedure exists in more than one location, whether it is in the same directory path or a different one, the latest procedure read in will take precedence.</p>
OutputBackups	<p>Specifies the number of script output directories that should be retained. Valid values are in the range <i>0</i> to <i>maxint</i> and the special value <i>-1</i>, which means that backups should never be removed. Zero backups indicates that script output for the current run only is maintained.</p>
Term3270	<p>Specifies the default 3270 emulation configuration section of the <i>xmyConfig</i> file for this SE.</p>

OutputLevel Specifies the type of information that should be recorded in the script output directory.

The syntax of **OutputLevel** is a Tcl list with zero or more elements representing output file prefixes. Setting **OutputLevel** to {""} causes the *output* file to be empty. An asterisk ({*}) means include all types of output.

Error output should always be enabled or errors will not be seen in the output file.

Valid values are:

- childscr
- compare
- error
- lang
- script
- summary
- sutimage
- suttiming
- testobj
- user

Default = {**error, user**}

The following OutputLevels are recommended:

- For Standalone Engines
OutputLevel = (error, childscr, compare, script,
summary, sutimage, testobj, user)
- For Embedded Engines
OutputLevel = (error, childscr, compare, script,
testobj, user)
- For Background Engines
OutputLevel = (error, childscr, compare, script,
summary, sutimage, testobj, user)

See Section [10.4.6](#) for more information on setting the content level of the script output directory.

TermAsync Specifies the default asynchronous emulation configuration section of the *xmyConfig* file for this SE.

Only one of the following two entries should be contained in the *xmyConfig* file. If neither one is used, the script output goes to the script directory, which is the directory where you executed the script.

OutputRoot Specifies the path for the root file system under which script output directories should be created. Script output directories are created using the same directory hierarchy as the original scripts.

For example, if the value for **OutputRoot** is */tmp* and the script is in */u/sam/script.tcl*, the output would be in */tmp/u/sam/script.tcl.out.tttttt.ttttt*, where *tttttt.ttttt* is the timestamp for when the script was executed.

OutputPath Specifies the script output directories. The hierarchy of the original scripts is not maintained; this is a flat directory.

3.3.3.2 Engine Group Configuration - EngineGroup

The **EngineGroup** entry **entry_name** contains entries that define a specific group of SEs. An example **Engine** entry is shown in [Figure 3-11](#).

```
EngineGroup SeGpl
Engine          = script_engine_1,
# LiteEngine    = <engine_name>,          # optional
Host            = <hostname>,
NumEngines     = 2;
```

Figure 3-11. xmyConfig Engine Group Entry

The valid **EngineGroup** configuration parameters are

Engine	Specifies the SE configuration entry for this group. Engine and LiteEngine are mutually exclusive.
LiteEngine	Specifies that the SE is configured as an SE Lite. LiteEngine and Engine are mutually exclusive.
Host	Specifies the name of the host machine that the group runs on. Connections are driven over TCP/IP. Usually, this will be an alias hostname, as in the example, but can also be an IP address.
NumEngines	Specifies the number of SEs that are in the group on start-up. An administrator can dynamically change the SE Group to have any number of SEs while the SD is running using xmyOM 's seincr (see Section 5.7.1 and Appendix A.1.2.7) and sedecr (see Section A.1.2.6) subcommands. This may be zero (0). If you do so, you can define many SEs without actually using an system resources since you will not be starting an SEs when the system is started. You later increase the number of SEs using xmyOM seincr .

3.3.3.3 Script Dispatcher Configuration - Dispatcher

The **Dispatcher entry_name** contains entries that define an SD. An example **Dispatcher** entry is shown in Figure 3-12.

```
Dispatcher SD1
  EngineGroups      = (SeGp1, SeGp3),
  DefaultEngineGroup = SeGp1,
  Host              = <hostname>,
  ActivityLogging   = yes;
```

Figure 3-12. xmyConfig Dispatcher Entry

The valid **Dispatcher** configuration parameters are

EngineGroups	<p>Specifies the names of all SE Groups the SD manages. They must appear in a comma-separated list enclosed in parentheses in the format</p> <pre>EngineGroups = (<SE_Group1>, <SE_Group2>, \ ..., <SE_GroupN>)</pre> <p>Note — Each SD can manage up to 24 SE Groups.</p> <p>This is a required parameter.</p>
DefaultEngineGroup	<p>Specifies the default SE Group for this SD entry.</p> <p>This is a required parameter.</p>
Host	<p>Specifies what host the SD is supposed to run on. The SD starts up all the SEs in the SE Groups it manages.</p> <p>The host must be the same host as the Operability Agent (OA) in the <i>xmyConfigOP</i> file that has this SD in its responsibility list. See Section 3.4 for information on the <i>xmyConfigOP</i> file and Section 4 for information on OAs and Operability Management.</p> <p>This is a required parameter.</p>
ActivityLogging	<p>Specifies whether the SD should do activity logging. See Section 10.3 for information on activity logging.</p> <p>yes Do activity logging</p> <p>no Don't perform activity logging</p> <p>Default = yes</p>

UserConcurrencies	<p>Tells the SD whether or not to pay attention to user concurrencies.</p> <p>yes User concurrency limits are enforced (e.g., if a user's actual-concurrency value is 7, then that user will be able to run at most seven scripts at a time, not including child scripts).</p> <p>Concurrency levels are set using the xmyCmd CLUI subcommands (Appendix A.1.2).</p> <p>The xmyCmd subcommands that control concurrency levels are</p> <ul style="list-style-type: none">• dftusrconc• sysconc• usrmaxconc• usractconc. <p>Note — The usractconc subcommand lets users set their own concurrency limits. It is not used by the MYNAH Administrator.</p> <p>no User concurrency limits are not enforced, and users can have any number of scripts running at a time, limited only by the overall system (i.e., SD) concurrency, and system resources such as available SEs.</p> <p>Default =yes</p>
--------------------------	---

3.3.4 Complete xmyConfig Example

Figure 3-13 contains a complete example *xmyConfig* file as delivered with the MYNAH System. This includes the use of the **%INCLUDE** statement to load the *xmyConfig.General*, *xmyConfig.TOP*, and *xmyConfig.GT* files.

As mentioned earlier, the order of the entries is only important when a **parameter** refers to an **entry_name**. In this case the **entry_name** must have been defined earlier. For example, the **Engine Standalone** entry uses the parameter **TermAsync = Async_type_1**, which had to be defined prior to the **Engine Standalone** definition.

```
%INCLUDE xmyConfig.General # required
%INCLUDE xmyConfig.TOP # uncomment for AppApp testing
%INCLUDE xmyConfig.GT # uncomment for GUI testing

#Term3270 Term3270_type_1
#   Host = <hostname>,
#   Model = 2, # optional
#   Port = 23, # optional
#   CompareInvisibleFields = Yes, # optional
#   ScreenIdentificationFile = "", # optional
#   TN3270E = No, # optional
#   ShowAttributes = No, # optional
#   TagDir = /opt/XXXXmyn/mynah_home/tag, # optional
#   Timeout = 300, # optional
#   InitialWait = No, # optional
#   InitialWaitExpect = "", # optional
#   CollectKeyCount = No, # optional
#   UnderlineUnprotectedFields = Yes,
#   VendorPath = "/opt/XXXXioc/ioconcepts";

TermAsync Async_type_1
    Terminal = vt100,
    Timeout = 60,
    ShowAttributes = No,
    Shell = /bin/sh,
    BufferSize = 2048,
    AuxTerminfo = .terminfo;

#   Engines

Engine Standalone
#   StartUpScript = "", # optional
#   LibraryPath = /opt/XXXXmyn/mynah/lib, # optional
#   ProcRepository= /opt/XXXXmyn/mynah/lib/tcl, # optional
#   OutputLevel = ( error, childscr, compare,
#                   script, summary, sutimage,
#                   testobj, user ), # optional
#   OutputBackup = 0, # optional
#   OutputPath = /opt/XXXXmyn/mynah/standalone, # optional
#   Term3270 = Term3270_type_1, # optional
#   TermAsync = Async_type_1;
```

Figure 3-13. Example xmyConfig File (Sheet 1 of 3)


```
Engine Embedded
#   StartupScript = "", # optional
#   LibraryPath = /opt/XXXXmyn/mynah/lib, # optional
#   ProcRepository= "/opt/XXXXmyn/mynah/lib/tcl", # optional
#   OutputLevel = ( error, user, script, compare,
#                   summary, sutimage), # optional
#   OutputBackup = 1, # optional
#   OutputPat = /opt/XXXXmyn/mynah/embedded, # optional
#   Term3270 = Term3270_type_1, # optional
#   TermAsync = Async_type_1;

Engine script_engine_1
#   Mode = Stateless,
#   StartUpScript = "/opt/XXXXmyn/mynah/lib/tcl/io.tcl", # optional
#   LibraryPath = "/opt/XXXXmyn/mynah/lib",
#   ProcRepository = "/opt/XXXXmyn/mynah/lib/tcl",
#   OutputLevel = ( error, childscr, compare,
#                   script, summary, sutimage,
#                   testobj, user ),
#   OutputBackups = 2,
#   TermAsync = Async_type_1;
#   Term3270 = Term3270_type_1, # optional

#Engine script_engine_2
#   Mode = ConnOnly,
#   StartUpScript = "/opt/XXXXmyn/mynah/scripts/startup",
#   LibraryPath = "/opt/XXXXmyn/mynah/lib",
#   ProcRepository = "/opt/XXXXmyn/mynah/lib/tcl",
#   OutputLevel = ( error, childscr, compare,
#                   script, summary, sutimage,
#                   testobj, user ),
#   OutputBackups = 0,
#   TermAsync = Async_type_1,
#   Term3270 = Term3270_type_1;

Engine script_engine_3
#   Mode = Stateless,
#   StartUpScript = "",
#   ExecScript = /opt/XXXXmyn/mynah/demo/scripts/wrapper.tcl,
#   LibraryPath = "/opt/XXXXmyn/mynah/lib",
#   ProcRepository = "/opt/XXXXmyn/mynah/lib/tcl",
#   OutputLevel = ( error, childscr, compare,
#                   script, summary, sutimage,
#                   testobj, user ),
#   OutputBackups = 2;
```

Figure 3-13. Example xmyConfig File (Sheet 2 of 3)

```

Engine  script_engine_litel
  Mode      = Stateless,
#  StartUpScript = "/opt/XXXXmyn/mynah/lib/tcl/io.tcl", # optional
  LibraryPath = "/opt/XXXXmyn/mynah/lib",
  ProcRepository = "/opt/XXXXmyn/mynah/lib/tcl",
  OutputLevel = ( error, childscr, compare,
                  script, summary, sutimage,
                  user  ),
  OutputBackups = 2,
  TermAsync    = Async_type_1;

#Engine Groups

EngineGroup SeGp1
  Engine      = script_engine_1,
  Host        = <hostname>,
  NumEngines  = 2;

#EngineGroup SeGp2
#  Engine      = script_engine_2,
#  Host        = <hostname>,
#  NumEngines  = 2;

EngineGroup SeGp3
  Engine      = script_engine_3,
  Host        = <hostname>,
  NumEngines  = 0;

EngineGroup SeGp4
  LiteEngine  = script_engine_litel,
  Host        = <hostname>,
  NumEngines  = 1;

#Dispatchers

Dispatcher SD1
  EngineGroups      = (SeGp1, SeGp3),
  DefaultEngineGroup = SeGp1,
  UserConcurrencies = yes,
  Host              = <hostname>,
  ActivityLogging   = yes;

#Dispatcher SD2
#  EngineGroups      = (SeGp1, SeGp2 SeGp4),
#  DefaultEngineGroup = SeGp1,
#  UserConcurrencies = yes,
#  Host              = <hostname>,
#  ActivityLogging   = yes;

```

Figure 3-13. Example xmyConfig File (Sheet 3 of 3)

3.4 The xmyConfigOP File Syntax

Configuration information for the Operability Management (OM and OA) processes is contained in a file called *xmyConfigOP*. This file is located in *\$XMYHOME/config*.

NOTE — Operability Management gives you, a MYNAH administrator, a single mechanism with which you can start, stop and get status of all of the MYNAH processes from any host, including those processes not developed by the MYNAH System but that are an integral part of the MYNAH operation (e.g. all of the Telexel processes).

This section describes the entries of the *xmyConfigOP* file, which creates configuration information for the MYNAH Operability Management structure. Information on the Operability Processes (e.g., the OA, OM, **Autostart**) can be found in [Section 4](#).

xmyConfigOP file entries use the same format as the *xmyConfig* file entries:

```
entry_name LogicalName  
  parameter      = option,  
  parameter      = option;
```

NOTE — As with the *xmyConfig* file, all **entry_names** are case sensitive; you must enter them exactly as shown in the following list. All **LogicalName**, **parameter**, and **option** names are case insensitive, but any further uses of these names within the *xmyConfigOP* file must be entered exactly as you created them.

xmyConfigOP **entry_names** can be one of the three following reserved names, each of which is used to create specific operability configurations:

- | | |
|-------------------------|--|
| General | Used to create configuration parameters that apply to the entire MYNAH System. |
| Process | Used to define processes to be managed. |
| OperabilityAgent | Used to define the OA for the host. |

The **LogicalName** argument is used to assign a unique name to an **entry_name**.

The following sections detail the **parameters** and their values for each **entry_name**.

3.4.1 General Entry

The *xmyConfigOP* file must contain the same **General** entry as the *xmyConfig* file (Section 3.3.1). This entry is therefore saved to a separate file, e.g., *xmyConfig.General*, and included into the *xmyConfigOP* and *xmyConfig* files by entering the line

```
%INCLUDE xmyConfig.General
```

at the beginning of each file.

3.4.2 Process Entries

There are processes that must be running before you start the MYNAH System, and each process must have a **Process** entry in the *xmyConfigOP* file. These processes can be Telexel processes, BDs, or SDs. The syntax of a **Process** entry is shown in Figure 3-14.

```
Process LogicalName
Mynah      = {yes|no},
AutoStart  = {yes|no}, # Yes tells the OA to start
                    # process at boot time
Start      ="start command",
Stop       ="stop command",
Status     ="status command";
```

Figure 3-14. xmyConfig Process Entry Structure

The **Process** configuration parameters are

Mynah	Indicates whether the process is a MYNAH process. For example, there are Telexel processes that are required by MYNAH, but they themselves are not MYNAH processes. In this case, this parameter would be set to no . In addition, you can use the Operability feature to bring up your own processes. yes This is a MYNAH process no This is not a MYNAH process.
AutoStart	Indicates whether the process is automatically started when the OA is booted. yes Start this process when the OA is booted. no Do not start this process when the OA is booted.
Start	Specifies the command that starts the process.

Stop	Specifies the command that stops the process.
Status	Specifies the command that returns the status of the process.

The names you enter for the *LogicalName* are used as elements in the **Responsibility** list parameter for the **OperabilityAgent** entry. (See [Section 3.4.3.](#)) This list tells the OA which processes it is responsible for.

The **Start**, **Stop**, and **Status** commands for the background processes are standard Telexel commands.

[Figure 3-15](#) contains an example of an **Process** Entry.

```
Process xmySD1
  Mynah      = yes,
  AutoStart  = Yes,
  Start      = "xmyStartSD -n SD1",
  Stop       = "xmyStopSD -n SD1",
  Status     = "xmyStatusSD -n SD1";
```

Figure 3-15. Example xmyConfigOP Process Entry

3.4.2.1 .xmyStartup

One of the Operability Processes we define is **vxIpcDir**, which starts the Telexel directory service. For general Telexel installations, this process is started using the **vxIpcDir** command as its **Start** command. However, we recommend you instead use the MYNAH command **.xmyStartup**.

Unlike **vxIpcDir**, which simply tries to bring up the Telexel directory service,

1. **.xmyStartup** checks to see if **vxIpcDir** is already running. If so **.xmyStartup** quits.
2. If it is not already running, **.xmyStartup** checks to see if the port defined by the **vxIpcPort** variable is idle. If not, then **.xmyStartup** waits until it becomes idle before starting **vxIpcDir**. This is important because if you attempt to bring up **vxIpcDir** while the port is busy, **vxIpcDir** will fail to come up. The **.xmyStartup** command waits about 5 and a half minutes for the port to become idle. If it is still busy after that time, **.xmyStartup** gives up and exits with an error code.

The **.xmyStartup** command assumes that the OA is running on all client machines.

The **.xmyStartup** command is only intended to be run on the server machine in a MYNAH installation, and only at OA bring-up time on the server. In fact, it is only used via the *xmyConfigOP* file. If the **vxIpcDir** process has

```
start=".xmyStartup"
```

as its **Start** command and **vxIpcDir** is an **Autostart** process (which it should be), then it will be run at OA bringup time.

3.4.2.2 MYNAH Process Start, Stop, and Status Commands

The following lists the commands used to start, stop, and obtain status information for the BD, TD, and SDs.

All of the following commands accept the following options:

- h** Returns a brief help message for the command.
- H** Returns a detailed help message for the command.
- R** Returns the current release number of the MYNAH System.

3.4.2.2.1 BD Commands

The BD **Start**, **Stop**, and **Status** commands are

Start = **xmyStartBD**
Stop = **xmyStopBD**
Status = **xmyStatusBD**

3.4.2.2.2 TD Commands

The TD **Start**, **Stop**, and **Status** commands are

Start = **xmyStartTD**
Stop = **xmyStopTD**
Status = **xmyStatusTD**

3.4.2.2.3 SD Commands

The SD **Start**, **Stop**, and **Status** commands are

Start = **xmyStartSD**
Stop = **xmyStopSD**
Status = **xmyStatusSD**

The SD commands have the syntax

```
SD_commandname -n SD_LogicalName ?-r?
```

where *SD_commandname* is **xmyStartSD**, **xmyStopSD**, or **xmyStatusSD**.

In addition to the *?-hHR?* options listed in [Section 3.4.2.2](#), the SD commands also accept the following options:

-n *SD_LogicalName* This option specifies the name of the SD to start. This is a required argument since the SD must know its logical name. The *SD_LogicalName* must be one of those defined in the *xmyConfig* file.

-r This is an optional argument for **xmyStartSD**. It is not defined for **xmyStopSD** and **xmyStatusSD**.

This option is used to recover the “state” of the SD as that state existed when the SD last came down, whether the SD crashed or not (i.e., whether the SD came down gracefully, or on an abnormal termination condition). In particular, it tells the SD to recover the “configuration” of the SD when it last came down. The configuration is simply the set of SE Groups and the number of SEs in each group when the SD last came down. This is only different from the settings in the *xmyConfig* file if you used the CLUI **seincr** or **sedecr** subcommands.

There are two caveats to bear in mind when using this option

- An error will occur if the SE was running with an SE Group that has been removed from the *xmyConfig* file. The SD will refuse to come up since the SD does not know what host the removed SE Group should run on, and the SEs in that group will not know what mode to come up in.
- If any configuration information for that group has been changed in the *xmyConfig* file, the new configuration information will be used (other than the number of SEs in the group). This includes the host the group runs on as well as the mode of the SEs in that group. For example
 1. SE Group G is configured to have 5 SEs and to run on host h1 in FullState mode.
 2. The SD is started and the administrator changes SE Group G to have 7 SEs.
 3. The SD is brought down or crashes.
 4. The MYNAH Config file is changed so that SE Group G has 10 SEs and runs on host h2 in conn-only mode
 5. The SD is brought up with the **-r** option.

In this case, SE Group G will run on host h2, in conn-only mode and have 7 SEs. So it is only the number of SEs in the SE Group that is saved and restored by the SD. All other configuration information is read in from the *xmyConfig* file, even when the **-r** option is used.

Note — Always stop all MYNAH components before you change the *xmyConfig* file or you may have problems stopping processes.

3.4.2.2.4 CL (Collector) Commands

The CL **Start**, **Stop**, and **Status** commands are

Start = **xmyStartCL**

Stop = **xmyStopCL**

Status = **xmyStatusCL**

The CL commands accept the following options:

-n CL_LogicalName This option specifies the name of the CL to start. This is a required argument for the **xmyStartCL** and **xmyStopCL** commands since the CL must know its logical name. The *CL_LogicalName* must be the name of a **MsgCollector** entry in the *xmyConfig* file.

-s N This option lets you specify the sleep time in $N * .1$ seconds, where *N* is a number from 1 to 10. For example, if *N* = 1, the collector will start with a sleep time of 0.1 seconds.

If *N* is outside the range of 1 to 10, then the sleep defaults to 1 second. In addition, if this option is not specified, the sleep defaults to 1 second.

Note — This option is available for **xmyStartCL** only.

NOTE — The CL commands do not support the *?-hHR?* options listed in [Section 3.4.2.2](#).

3.4.2.2.5 TN3270 (Prt3270) Printer Emulator Commands

The Prt3270 domain requires the **iocluprt** printer client from IO concepts. Once this process is started, it listens for print requests from the 3270 host/gateway. This process goes down when there is an auto logoff from the 3270 host/gateway. The auto logoff occurs when there is no activity on the given printer for a certain amount of time. The amount of

time depends on the site policy. For example, the gateway to which the MYNAH System test environment connects to has an auto logoff time of 120 minutes (2 hours). This means you must check from time to time if to see **iocluprt** is running. A UNIX shell command can be written to take care of this situation.

iocluprt takes the following form:

```
$IOC/iocluprt -E -h ip_address -c cfg.file -s server_name \  
-C lu_name -d mode> printer.log &
```

where

\$IOC	Contains the name of the I/O Concepts home directory
-E	Indicates that the TN3270E protocol is to be used instead of the TN3270 protocol.
-h ip_address	Specifies the host on which the TN3270E server is running
-c cfg.file	Specifies the name of the iocluprt configuration file
-s server_name	Indicates the name of the server. Typically this is telnet .
-C lu_name	Specifies the LU name that is defined as a printer
-d mode	Indicates the debug mode

NOTE — See the *X-Direct User's Guide* for more information on **iocluprt**.

The TN3270 **Start**, **Stop**, and **Status** commands are:

```
Start = $IOC/iocluprt -E -h 128.96.102.234 -c $IOC/iocluprt.cfg -s telnet \  
-C MYNAHPRI -d 2> printer.log &
```

```
Status = ps -ef | grep iocluprt
```

```
Stop = kill -USR1 <process id obtained from Status command>
```

The **Stop** command cannot be automated without writing a complex set of commands. It is best that you do this manually.

NOTE — The printer client **iocluprt** will not work with a TN3270 server. It needs a TN3270E server to work. The TN3270E server could be running directly on an IBM mainframe or on a gateway connected to an IBM mainframe.

3.4.3 OperabilityAgent

There is one OA per host and every OA is defined in the *xmyConfigOP* file. The syntax of an **OperabilityAgent** entry is shown in [Figure 3-16](#).

```
OperabilityAgent OA_Hostname
    Responsibility=(list of names);# managed processes
```

Figure 3-16. xmyConfigOP OperabilityAgent Entry Structure

NOTE — An *OA_Hostname* can contain hyphens (-), for example *sc-rooge*.

The **OperabilityAgent** configuration parameter is

Responsibility Specifies the names of all of the processes for which the OA has responsibility. They must appear in a comma-separated list enclosed in parentheses in the format

Responsibility = (<process1>, <process2>, ..., <processN>)

NOTE — The **OperabilityAgent LogicalName** must be the name of the host on which the OA runs.

[Figure 3-17](#) contains an example of an **OperabilityAgent** entry.

```
OperabilityAgent selene
    Responsibilities = (vxDir, vxGateway, vxLogToFile,
        vxErrorServer, xmyTD, xmyBD, xmySD1, xmyQueues, xmyLS);
```

Figure 3-17. Example xmyConfigOP OperabilityAgent Entry

While there is only one OA per host, you must define the OAs for all hosts in the *xmyConfigOP* file. In addition, each process will be able to appear in multiple OAs' responsibility lists. This way you don't have to redefine the processes for each host.

When the OA starts, it determines its name by looking at what host it was started on. The OA then looks for its entry in the *xmyConfigOP* file to see what processes it is responsible for and immediately starts the ones with **Autostart = yes**.

Each **LogicalName** that appears in the **Responsibility** list for an OA must correspond to a defined **Process** entry.

The OA starts these processes *in the order listed* in the **Responsibility** list.

NOTE — This is very important because some processes *must* be up and running before other processes can start. These are

1. vxIpcDir (started via the **.xmyStartup** command)
2. The rest of the Telexel processes
3. BDs
4. TD
5. SDs.

3.4.4 Example xmyConfigOP File

Figure 3-18 contains a sample *xmyConfigOP* file for a MYNAH configuration. In this example, the General entry in Figure 3-1 has been saved to the file *xmyConfig.General* and included into the *xmyConfigOP* file via the

```
%INCLUDE xmyConfig.General
```

statement.

The following Operability processes must appear in an *xmyConfigOp* file:

vxIpcDir	This defines the Telexel directory service. One occurrence of this process must be running. It provides the directory name service for all other processes.
vxGatewayhost	This defines the Telexel gateway daemon. One occurrence of this process must be running on each host in the MYNAH System configuration.
vxErrorServer	This defines the Telexel error server.
vxLogToFile	This defines the Telexel log to file process.
xmyBD	This defines the Operability configuration for the Boot Daemon (BD). Note — The xmyBD process uses the nice command to start the BD process so that the Script Engines are started with a lower priority. For information on the nice command, please refer to the nice manual page. nice is available on both Solaris and HP-UX systems.
xmyLS	This defines the Operability configuration for the License Server.
xmyQueues	This defines the Operability configuration for the queues.

The following Operability processes are optional, depending on the packages you are using and how you are using the MYNAH System (e.g., whether you are using test management):

xmySDn	<p>This defines the Operability configuration for a Script Dispatcher (SD). At least one SD must be defined per <i>xmyConfigOp</i> file if scripts are to be executed using a background engine.</p> <p>The example <i>xmyConfigOp</i> file in Figure 3-18 defines two SDs, SD1 and SD2.</p>
xmyTD	<p>This defines the Operability configuration for the Trigger Daemon.</p> <p>Note — Use the following guidelines when defining xmyTD:</p> <ul style="list-style-type: none"> • xmyTD must <i>not</i> be defined if you are not using test management abilities. • xmyTD can be defined only once per each <i>xmyConfigOp</i> file.
xmyCollector	<p>This defines the Operability configuration for the Collector Process (CL) and is required if you are using the Application to Application package.</p>
iocLicense	<p>This defines the Operability configuration for the I/O Concepts X-Direct TN3270 software. This lets you start the software, used by MYNAH Users to test or automate tasks over a 3270 (synchronous) interface, when the system starts.</p> <p>NOTE — The entries for this process are commented in Figure 3-18.</p>
iocluprt	<p>This defines the Operability configuration for the I/O Concepts X-Direct TN3270 Printer emulator. This lets you start the Printer emulator, used by the MYNAH PRT3270 subsystem.</p> <p>NOTE — The entries for this process are commented in Figure 3-18.</p>

```
%INCLUDE xmyConfig.General

Process vxIpcDir
  Mynah      = No,
  Autostart  = Yes,
  Start      = ".xmyStartup",
  Stop       = "vxIpcDown -d",
  Status     = "vxIpcProcesses";

Process vxGateway
  Mynah      = No,
  Autostart  = Yes,
  Start      = ".xmyIpcUp",
  Stop       = "vxIpcDown",
  Status     = "vxIpcUp";

Process vxErrorServer
  Mynah      = No,
  Autostart  = Yes,
  Start      = "vxErrorServer $TELDIR/lib/errorText\  
                $XMYDIR/lib/xtw_error_text\  
                $XMYDIR/lib/xmyErrorText",
  Stop       = "vxIpcTerm vxErrorServer",
  Status     = "vxIpcProcesses vxErrorServer";

Process vxLogToFile
  Mynah      = No,
  Autostart  = Yes,
  Start      = "vxLogToFile -C 3 -M 2m $XMYDIR/syslog/adminLog",
  Stop       = "vxIpcTerm vxLogDestFile",
  Status     = "vxIpcProcesses vxLogDestFile";

Process xmyBD
  Mynah      = Yes,
  Autostart  = Yes,
  Start      = "nice -5 xmyStartBD",
  Stop       = "xmyStopBD",
  Status     = "xmyStatusBD";

Process xmySD1
  Mynah      = Yes,
  Autostart  = Yes,
  Start      = "xmyStartSD -n SD1",
  Stop       = "xmyStopSD -n SD1",
  Status     = "xmyStatusSD -n SD1";

Process xmySD2
  Mynah      = Yes,
  Autostart  = Yes,
  Start      = "xmyStartSD -n SD2",
  Stop       = "xmyStopSD -n SD2",
  Status     = "xmyStatusSD -n SD2";
```

Figure 3-18. Example MYNAH xmyConfigOP File (Sheet 1 of 3)

```

#Process xmyTD
#   Mynah           = Yes,
#   Autostart       = Yes,
#   Start           = "xmyStartTD",
#   Stop            = "xmyStopTD",
#   Status          = "xmyStatusTD";

Process xmyLS
    Mynah           = Yes,
    Autostart       = Yes,
    Start           = "xmyStartLS",
    Stop            = "xmyStopLS",
    Status          = "xmyStatusLS";

Process xmyQueues
    Mynah           = Yes,
    Autostart       = No,
    Start           = "",
    Stop            = "",
    Status          = "xmyCmd checkQueues";

#Process xmyCollector
#   Mynah           = yes,
#   Autostart       = Yes,
#   Start           = "xmyStartCL -n <msg collector name> -s 5",
#   Stop            = "xmyStopCL -n <msg collector name>",
#   Status          = "xmyStatusCL -n <msg collector name>";

#Process iocLicense
#   Mynah           = no,
#   Autostart       = yes,
#   Start           = "ioclm.sh start",
#   Stop            = "ioclm.sh stop",
#   Status          = "ps -ef | grep ioclmd";

#Process iocluprt
#   Mynah           = no,
#   AutoStart       = yes,
#   Start           = "$IOC/iocluprt -E -h 128.96.102.234 -c $IOC/iocluprt.cfg\
                    -s telnet -C MYNAHPR1 -d 2> printer.log &"
#
#Where
#
#   $IOC contains the name of the I/O Concepts home directory.
#   -h option specifies the host on which the TN3270E server is running.
#   -c option specifies the name of the config file.
#   -C option specifies the LU name which is defined as a Printer.
#
#   Status          = "ps -ef | grep iocluprt"
#   Stop            = "kill -USR1 <process id obtained from Status command>"
#
#   The Stop command cannot be automated without writing a complex set of
#commands. It is better for you to do this manually.

```

Figure 3-18. Example MYNAH xmyConfigOP File (Sheet 2 of 3)

```
#OPERABILITY ENTRIES FOR AGENTS

OperabilityAgent <hostname>
    Responsibilities = (vxIpmdir, vxGateway, vxLogToFile,
#                   iocLicense,
#                   xmyCollector,
#                   xmyTD,
                    xmyLS, vxErrorServer, xmyBD, xmySD1,
                    xmyQueues);

#OperabilityAgent <hostname_2>
#    Responsibilities = (vxGateway, xmyBD, xmySD2, xmyQueues);
```

Figure 3-18. Example MYNAH xmyConfigOP File (Sheet 3 of 3)

3.5 The `.xmyMYNAHrc` File

The `xmyRunMynah` process creates and maintains a `.xmyMYNAHrc` file in the user's home directory. This file is used to store desktop and preference information. It is read by the process upon start-up.

3.6 Specialized Configuration Concerns

3.6.1 Batch Package Configuration — The `.netrc` File

A `.netrc` file must be created with permissions set to `400` such that it is read-able only to the MYNAH administrator's login and placed in the home directory of the administrator if users are to be permitted to submit batch jobs via MYNAH processes started by the administrator. The contents of the file should be the names of the machines that will be accessed to process batch jobs and the login and password information permitted for that access.

The `ftp` command requires that a `.netrc` file be created in the home directory of the owner of the process executing the batch commands described in this document. The batch commands are executed as Tcl procedures. The `.netrc` file must contain an entry for each host machine that a user might select to submit a batch job to. Each entry in this file will contain the host machine name, the login name on that machine, and the associated password. The `.netrc` file must be protected from access by any other user than its owner. To establish this protection, the file's permissions must be set to read-only for the owner and no access by group or other. The mode of the file should be set to `400`. An example of a `.netrc` file is shown as follows:

```
machine pyibm1 logon siuol password x123
machine pyibm2 logon ortep password xpdq
```

If batch jobs are submitted to a script engine that was started by another user, the user's `.netrc` file who started the script engine will be used by the batch job processing tools described in this document.

Although the usage of the `.netrc` file is suggested by the `ftp` product, it is considered a security hole and another method of providing the logon and password information will be investigated in post MYNAH 5.2.

3.6.2 Maintaining a Tcl Procedures Library

This section describes how to add user written Tcl procedures to your MYNAH installation so that they are generally accessible. These procedures must be contained in a **ProcRepository**. A set of generally accessible Tcl procedures is shipped with the MYNAH System. They are contained in the *ProcRepository* `$XMYDIR/lib/tcl`. Locally written procedures should be stored in the *ProcRepository* `$XMYHOME/lib/tcl`.

To add a set of procedures to a *ProcRepository* you:

- Put them in a file with the suffix **.tlib** in *ProcRepository*
- Add a line with this syntax to the top of the file:

```
#@package: <proc lib name> <proc name list>
```

where:

- <*proc lib name*> Is the user-supplied name of the procedure library.
- <*proc name list*> Is a space separated list of the procedures in this library that should be exported.

- Create an index file from within **xmyTclsh** for the library running *buildpackageindex* <*pro file list*> from within **xmytclsh**.

NOTE — The index files can be created automatically as long as the user running the script has write permissions in the *ProcRepository*.

For more information on *buildpackageindex* and other Tcl library management command, see the Scriptics Tcl man pages at <http://www.scriptics.com/man/tcl8.0/contents.htm>.

3.6.3 Synchronizing Clocks for the MYNAH System

System clocks on networked systems tend to drift. This condition exists because the internal clocks on different machines are not synchronized to second accuracy. In fact, some machines can drift minutes apart.

This is always a problem, especially for AppApp emulation where the AppApp and script processes are distributed between different machines.

For example, in the AppApp architecture, a set of MYNAH 5.3 “Collector” processes are started on a machine (call it machineA), and the Tcl script (SE) can be executed from a different machine (call it machineB). The purpose of the “Collector” process is to receive messages sent by the SUT, timestamp each messages, and write them to disk. Note that the timestamp is relative to machineA’s internal clock. On machineB, our customers execute the AppApp Tcl scripts which connects to a “collector” process on machineA and asks the

“collector” to send and receive SUT messages. Note that the timestamp of the sent messages are relative to machineB’s internal clock.

Let’s assume the current time on machineA is 1 PM and on machineB is 1:10 PM. At 1:10 PM the script on machineB instructs the “Collector” process on machineA to send a message and return any message which was received after the send time (1:10 PM). The response message is received 1 minute later and time stamped by machineA at 1:01 PM. But this message is NOT returned to the script on machineB, because the script is expecting to receive messages with a timestamp after 1:10 PM. So, the script will never receive the response message.

There are two solutions: setup the SE Group to run on the same machine as the collector or have the clocks synchronized. Since drifting clocks can cause problems/confusion in other parts of the MYNAH System beside AppApp, we recommend synchronizing your clocks.

Two possible methods for synchronizing your clocks are **rdate** and Network Time Protocol (NTP).

rdate is a UNIX command that lets you set the time for your machine to that of another machine. For example, to synchronize your machine’s clock with that of the machine **selene**, you would type

```
rdate selene
```

You must be **root** to use **rdate**.

rdate, however, is only as accurate as the server you’re using as a reference. NTP uses a multi-layered architecture, with the top level attached to such accurate hardware devices as radio receivers.

NTP runs on most versions of UNIX and is freely available from the Internet via ftp at

louie.udel.edu/pub/ntp/xntp3.5c.tar.Z

For more information on synchronizing clocks on networked systems, see the Article Time Bombs by Hal Stern in the April, 1996 issue of *SunWorld Online*. The complete text of the article is available on the Web. The URL is

<http://www.sun.com/sunworldonline/swol-04-1996/swol-04-sysadmin.html>

3.6.4 Overriding Default ASCII-EBCDIC Translations

The default ASCII-EBCDIC character translation provided by I/O Concepts may not be appropriate for all MYNAH installations. Specifically, I/O Concepts translates the ASCII "[" (left square bracket) to the EBCDIC cent sign and the ASCII "]" (right square bracket) to the EBCDIC solid vertical bar. If your SUT uses square brackets, then the default character translation has to be overridden by following these steps:

1. **cd** to *\$IOCHOME/Hllapi*.
2. Create a file called *xlate.instr* that contains descriptions of the new translations. For example, the following lines create new mappings between the ASCII square brackets and EBCDIC characters:

```
ASCII '[' (0x5B) IS EBCDIC 0xAD  
ASCII ']' (0x5D) IS EBCDIC 0xBD  
/* eof */
```

3. Run **ioctrans** to create the new translation table, i.e., execute

```
$IOCHOME/ioctrans -r xlate.instr -o new.xlt
```

The newly created translation table, in this example *new.xlt*, can be used in the *xmyConfig* file and/or by the Term3270 **connect** method (see the *MYNAH System Scripting Guide*).

This newly created translation table can also be specified in the IO/Concept configuration file. To do so

1. Edit the file *\$IOCHOME/Hllapi/HIocte* and find the line

```
!hiocte.xltFile: /usr/iocinst/azerty.xlt
```
2. Uncomment the line by removing the "!" and replace the listed filename with the full path to *new.xlt*.
3. Open a 3270 connection in the MYNAH System to confirm that the new translations have taken effect.

See the I/O Concepts User Guide for more information on ASCII-EBCDIC character translations.

4. Operability Management — Starting and Stopping MYNAH Processes

Operability Management gives you, a MYNAH administrator, a single mechanism for starting, stopping, and getting status of the MYNAH processes from any host, including those processes not developed by MYNAH but that are an integral part of the MYNAH operation (e.g., all of the Telexel processes).

This mechanism is composed of an Operability Manager (OM), an Operability Agent (OA), and configuration information. The OM is invoked by typing the **xmyOM** command.

NOTE — See Appendix [A.1.3](#) for a discussion of the **xmyOM** command.

4.1 Basic Steps

This section briefly lists the basic steps used for configuring the Operability Management files and starting MYNAH processes. The following sections provide detailed descriptions.

1. Set up the *xmyConfigOP* file. This file contains all of the configuration information for the **platform** processes required by the OM and the OA, as well as some information about all SDs.
2. Start the OA on each MYNAH system host. Generally this is done by including the **xmyStartUp** command in a start-up file. The file *S99mynah.eg* is included in the *\$XMYDIR/examples/admin/scripts* directory. It can be updated for the correct path and machine names and then placed in the */etc/rc3.d* directory (for Solaris) or */sbin/rc3.d* (for HP-UX) directory. (See [Section 4.8](#).)

The OA reads the *xmyConfigOP* file to determine which processes it is responsible for and then starts those processes on that machine that have **Autostart = Yes**, such as Telexel processes and the Boot Daemon (BD).

3. Start each SD and the associated SEs using the **xmyOM** command. This is *only* necessary if the SD entry in the *xmyConfigOP* file *does not* have the **Autostart** option set to **Yes**.

NOTE — The SDs and SEs are defined in the *xmyConfig* file, but an SD's Autostart information is specified in the *xmyConfigOP* file.

4.2 Overview

MYNAH processes fall into two phase categories: the **platform** and the **application** phases. Platform processes are the processes that are required by the MYNAH Packages, such as the Telexel IPC processes and the MYNAH Boot Daemon (BD).

Application processes are the actual MYNAH processes, such as an SD, involved in executing scripts.

For start-up, the platform phase takes place first, then the application phase. For shut-down, it is the reverse: the application phase shutdown happens first, then the platform phase.

NOTE — The application phase can be shut-down, and then brought back up again, without shutting down the platform phase.

For each host in the MYNAH configuration, the platform processes are composed of

- A MYNAH OA
- The required Telexel processes, e.g., the Telexel IPC and logger processes
- The MYNAH BD and any other process required for the domains (e.g., the AppApp Package Collector process, TOPCOM).

The BD is a platform process required by an SD. Because SE groups can run on a different host than the SD, a BD must be running on all hosts that run the MYNAH System. The BD must be up and running for the SE Groups to come up; for each MYNAH installation, there is one BD per host. This is needed only if an SE Group is to run on that host.

The BD “manages” the SEs running on its machine. It starts the SEs, and it knows when an SE dies. If the SE dies abnormally, the SE’s exit code tells the BD why the termination occurred. If the SE died due to certain types of errors, the BD restarts it.

- The Trigger Daemon (TD)

The TD is a procedure that is stored in the database and implicitly executed when a table is modified, broadcasting that a change has been made to the database.

The TD process is responsible for sending dynamic updates about script execution status to GUI processes.

If dynamic updates are not happening at all, or seem to be happening very slowly, you should check the following:

- There may be more than one TD process running for the same database.

NOTE — There should be only one TD process running. If there is more than one both TD processes should be brought down and only one restarted.

- Telexel may think there are processes up that are actually defunct. This will cause the TD to be very slow. See (Section 5.12) for instructions on how to determine, and correct this situation.

NOTE — The TD is required only if your MYNAH installation uses a database. The TD runs only on the machine storing the MYNAH database.

NOTE — All platform processes should have the **Autostart** option set to **Yes**.

The MYNAH application processes are composed of

- The SD
- The associated SEs (which may be spread across the various hosts in the MYNAH configuration).

NOTE — The SD tells the BD to start the subordinate SE processes so they are not managed by the Operability feature.

4.3 Operability Design

As mentioned earlier, the design is composed of an OM, an OA, and configuration information.

There is only one OA per MYNAH host machine, per MYNAH configuration. The OA “manages” all MYNAH required processes on a host. The OM, via the **xmyOM** subcommands, provides a user interface to the OAs. The basic design is depicted in Figure 4-1.

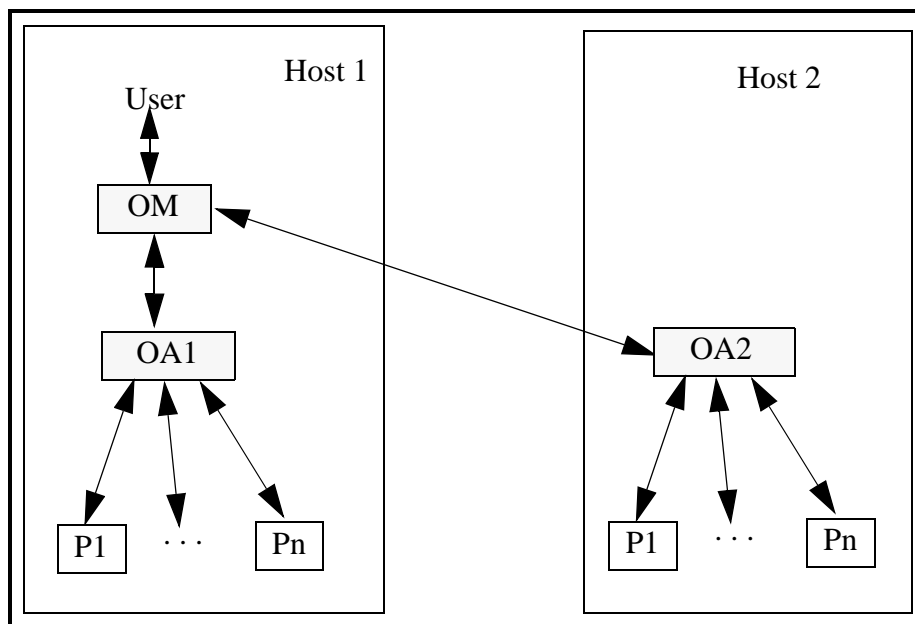


Figure 4-1. Operability Architecture

4.3.1 Operability Manager (OM)

The OM provides you with a set of commands to manage all MYNAH processes. The management is very high level. You can start or stop a process or determine if a process is running properly.

4.3.2 Operability Agent (OA)

The OAs are responsible for communicating the start, stop, and status requests to individual processes, and then communicating the reply back to the OM.

OAs are started at host boot time or by the user using the CLUI commands.

There is only one OA per MYNAH system host.

NOTE — This *must* be done on each MYNAH system host and not just on the MYNAH server system.

The OA reads the *xmyConfigOP* file to determine what platform and application processes it is responsible for. The *xmyConfigOP* tells the OA:

- The port number to use to listen on for messages from the OM. (This port number must be greater than 5000 and less than 65000.)
- The name of the individual processes it is to manage.
- All information it needs to know about each individual process that it is to manage.

The OA relies on the **Start**, **Stop**, and **Status** parameters that are defined for each process that the OA is responsible for.

NOTE — Each process entry contains an **Autostart** parameter. If a process's **Autostart** parameter is set to **Yes**, the OA automatically start that process when the OA starts. By default, the delivery configuration entries for all Telexel processes, the BD, and TD are set to **Yes**.

4.3.3 Operability xmyOM Subcommands

The CLUI's **xmyOM** commands takes a series of subcommands that let you manage the MYNAH processes. Table 4-1 lists the **xmyOM** subcommands and their functions. See Appendix A.1.3 for complete descriptions of each subcommand.

Table 4-1. xmyOM Sub-commands

Command	Function
autostart	Causes the OA on the specified host to start up all Autostart processes defined for that OA.
autostop	Causes the OA on the specified host to shut down all the Autostart processes defined for that OA.
query	Provides information on all managed processes.
readconfig	Lets you request that all OA processes re-read the <i>xmyConfigOP</i> file.
recycle	Shuts down and restarts all Autostart processes on the specified host.
shutdown	Terminates an OA.
stop	Stops a process.
start	Starts a process.
status	Sends a status request to a process.

When a subcommand is executed, the OM takes the appropriate action:

- If the request is a query, the OM simply produces the output information.
- If the request is to **start**, **stop**, or obtain the current **status** of a process, the OM forwards the request to the appropriate OA(s) (the OA for the host machine that the targeted process runs on).
- If the request is to reread the *xmyConfigOP* file (i.e., **readconfig** subcommand), the OM sends a request to all OAs defined in the *xmyConfigOP* file, telling each OA to reread the *xmyConfigOP* file.

This request provides a convenient method for you to get *xmyConfigOP* changes to take effect. Without this request in the OM, you would have to bring down all OAs and then restart all OAs to effect a configuration change.

- If the request is to **shutdown**, **autostart**, **autostop**, or **recycle**, the OM forwards the request to the OA indicated on the command-line.

4.4 Licensing

NOTE — Remember, you are advised to run the license server on the same machine as the **vxIpcDir** process.

4.4.1 MYNAH Licensing Commands

The MYNAH System provides the **xmyStartLS**, **xmyStopLS**, and **xmyStatusLS** commands to let you stop and start the License server and obtain information about the status of the License server.

xmyStartLS	Starts the License server. xmyStartLS will look for the file <i>\$XMYHOME/config/xmyLicenses</i> for the licensing keys.
xmyStopLS	Stops the License server.
xmyStatusLS	Displays the key information in <i>xmyLicenses</i> in a readable format.

4.4.2 Starting the License Server

After the licensing code is installed, start the license server by typing

```
xmyStartLS
```

while logged in to the machine that will run the server. **xmyStartLS** assumes that the license server resides in *\$XMYDIR/bin* and that the licensing key resides in the directory *\$XMYHOME/config* in a file called *xmyLicenses*.

WARNING — Remember, the License Server software does not properly process dashes (-) in a directory path or name.

The license server runs in the background and does not require regular monitoring. If a problem does develop, use **xmyStatusLS** (see Section 4.4.4.1) to confirm that the server is responding properly. If not, try stopping and restarting the server, after bringing down all licensed MYNAH System products.

NOTE — The License Server can also be started or stopped using the *xmyConfigOP* file.

4.4.3 Stopping the License Server

To stop the license server, type

```
xmyStopLS
```

If problems persist after restarting the license server, contact the MYNAH System support staff for assistance.

4.4.4 License Utilities

4.4.4.1 Monitoring

The **xmyStatusLS**¹ program provides information about the installed licenses. It has the following syntax:

```
xmyStatusLS ?machine_name?
```

where *machine_name* is the name of the machine on which the server is installed. *machine_name* is optional if:

- The environment variable LSHOST is set.
- A file called *LicenseServ* in the current directory contains the name of the machine running the server.
- The server runs on the local machine.

xmyStatusLS writes the information to the standard output.

xmyStatusLS provides information about all software and users that are currently licensed through *xmyLicenses* and all licenses in use, as in the following:

```
xmyStatusLS
lsmonitor for LicenseServ 3.0 Copyright (c) Viman Software

Feature Name: SCRIPTEXEC(v5.0) (floating license) No expiration date.
Concurrent licenses: 65535
Available unreserved   : 65535           In Use: 0
Available reserved     : 0             In Use: 0
Number of subnets     : 0
Site License info      : *.*.*.*
Hostid based locking

Feature Name: XMYTERMASYN(v5.0) (floating license) No expiration date.
Concurrent licenses: 65535
Available unreserved   : 65535           In Use: 0
Available reserved     : 0             In Use: 0
```

1. The xmyStatusLS program uses the **lsmonitor** utility from Viman Software.

```
Number of subnets      : 0
Site License info      : *.*.*.*
Hostid based locking

Users:
wunn                    (DefaultGrp)
Host name              : ariel
X display name         : ariel:0
Shared ID name         : viman default shared id
Number of keys         : 1
Q wait time            : 10
Hold time              : 0 minute(s)
```

4.4.4.2 Decoding

A vendor utility called **lsdecode** is shipped with the MYNAH System in the directory `$XMYDIR/bin`. It decrypts part of the information in the file containing licensing codes (usually `xmyLicenses`). If you have trouble using MYNAH System products after installing the licensing code, use **lsdecode** to confirm that the installed license code is correct.

The **lsdecode** utility has the following syntax:

```
lsdecode ?-s license_key_file?
```

where **-s license_key_file** is the name of the file containing the license, as in the following example:

```
lsdecode -s $XMYHOME/config/xmyLicenses

          License Decoding Utility
    Copyright (c) Viman Software 1992, 1993

Reading license codes from file: "/opt/SUNWmyn/mynah/config/xmyLicenses"

License code: "YEXSUIHFTTLFKWTTBACXT9K6TIM8PX49BKUATTMS"
Feature Name: "SCRIPTEXEC", Feature Version Number: "5.0"
Exclusive license (will override additive licenses).
Floating license with
  Server host ID: "8E" (2 least significant hex characters)
Maximum concurrent users : 65535
Vendor Info              : ""
Lifetime of keys         : 2 minute(s)
Held licenses disabled.
Shared licenses disabled.
License has no expiration.
```

4.5 Starting Processes

As stated earlier, the OAs are started at host boot time. The OA has its `setuid` and `setgid` bits set. (This was done at installation time.) **xmyOA** should be owned by a MYNAH administrator. This means that when an OA starts, it changes its `uid` and `groupid` to be the same as the `uid` and `groupid` of the `logid` that owns **xmyOA**. All processes that an OA starts will run with the same `uid` and `groupid` as the OA.

NOTE — The OA will NOT run as **root**, so if the owner of **xmyOA** is **root**, the OA will not come up. Normally, all processes will be owned by the MYNAH Administrator, **admin**.

NOTE — Normally, it is recommended that all processes start and stop via the `xmyConfigOP` file and either the `/etc/rc3.d/S99mynah` (Solaris) or `/sbin/rc3.d/S990mynah` file (HP-UX) file (see Section 4.8).

4.5.1 Solaris Start-up Mechanism

At start-up, Solaris looks in the `/etc/init.d` directory for the scripts used to start the system. Solaris then executes, in order, the start and kill process files in the `rc?.d` (where ? can be one of S or 0-6, e.g., `rcS.d`, `rc0.d`, or `rc4.d`) directories. Each `/etc/rc[S0 - 6].d` directory is designed for a specific function or state. For example, user-defined processes are stored in `/etc/rc3.d`.

The file names for start processes begin with S, such as `/etc/rc3.d/S99mynah`. All start files are executed in numeric and alphabetic order, i.e., S01, then S02, etc.

The file names for termination processes begin with K, such as `/etc/rc0.d/K01mynah`. All start files are executed in numeric and alphabetic order, i.e., K01, then K02, etc.

4.5.2 HP-UX Start-up Mechanism

At start-up, HP-UX looks in the `/sbin/init.d` directory for the scripts used to start the system. HP-UX then executes, in order, the start and kill process files in the `rc?.d` (where ? can be one of S or 0-4, e.g., `rcS.d`, `rc0.d`, or `rc4.d`) directories. Each `/sbin/rc[S0 - 4].d` directory is designed for a specific function or state. For example, user-defined processes are stored in `/sbin/rc3.d`.

The file names for start processes begin with S, such as `/sbin/rc3.d/S990mynah`. All start files are executed in numeric and alphabetic order, i.e., S01, then S02, etc.

The file names for termination processes begin with K, such as `/sbin/rc0.d/K001mynah`. All start files are executed in numeric and alphabetic order, i.e., K01, then K02, etc.

4.5.3 Starting at Boot Time

A MYNAH start-up file, `S99mynah` (Section 4.8.1 and Appendix B.1.5) is delivered with the system. You are not required to use this file, but it is strongly recommended since it contains environmental settings (such as library paths, hostports, and the hostname) that you will have to manually set if you do not use this file.

NOTE — Remember to edit this file so that settings reflect site-specific values.

If you do not use the `S99mynah` file but still want to start MYNAH processes at boot time, you must create your own start-up process file in the `/etc/rc3.d` directory (Solaris) or `/sbin/rc3.d` directory (HP-UX) on each OA host (each host that will run platform or application processes), and each file must contain two MYNAH commands.

4.5.3.1 .xmyRemovePips

The first command is **.xmyRemovePips**.

The **.xmyRemovePips** command removes “local” pip files (pip files of processes that run on the local machine) from the run directories in the configuration. This includes the following directories:

- `$XMYHOME/run/oa`
- `$XMYHOME/run/bd`
- `$XMYHOME/run/td`
- `$XMYHOME/run/sd`

The **.xmyRemovePips** command takes the following options:

- | | |
|------------|--|
| -h | Provides a usage statement |
| -v | Provides verbose output (shows what's about to be removed) |
| oa | Remove pip file in the OA's run directory for the OA on this host |
| bd | Remove pip file in the BD's run directory for the BD on this host |
| td | Remove pip file in the TD's run directory if the TD runs on this host |
| sd | Remove pip file(s) in the SD's run directory for SD(s) that run on this host |
| all | Remove pip files for all processes that run on this host |

The **.xmyRemovePips** command is run from the start-up script and should be performed as the MYNAH Administrator, **madmin**. Therefore, you should add a line such as the following to your start-up file:

```
/bin/su - madmin -c '/opt/SUNWmyn/mynah/bin/.xmyRemovePips all'
```

For example, if the machine is called **xyz**, then **.xmyRemovePips** removes the following files (assuming the TD runs on **xyz**):

- `$XMYHOME/run/oa/pip.oa.xyz`
- `$XMYHOME/run/bd/pip.bd.xyz`
- `$XMYHOME/run/td/pip.td.xyz`

The **.xmyRemovePips** command also removes all files of the form

```
$XMYHOME/run/sd/pip.sd.<sd_name>
```

where the SD called `<sd_name>` runs on machine **xyz**.

4.5.3.2 xmyStartUp

To start the OAs at boot time, the start-up file must contain the following command after the **.xmyRemovePips** command:

```
xmyStartUp
```

The OA starts those processes it is responsible for that have **Autostart = yes**. The OA simply executes the indicated **Start** command to start a process.

The OA then sits idle, listening on the OMPort for any incoming **start**, **stop**, **status**, **readconfig**, **shutdown**, **autostop**, **autostart**, or **recycle** requests.

As with **.xmyRemovePips**, **xmyStartUp** is run from the start-up script and should be performed as the MYNAH Administrator, **madmin**. Therefore, you should add a line such as the following to your start-up file:

```
/bin/su - madmin -c '${XMYDIR}/bin/xmyStartUp'
```

If all other processes are started via the *xmyConfigOP* file, then nothing else needs to be done. Otherwise see Section 4.5.4.

4.5.4 Starting a Specific Process

The **start** subcommand (Appendix A.1.3.7) lets you start specific processes. The basic syntax for **xmyOM start** is

```
xmyOM start ?-o oa_name? logical_process_name
```

where

- **-o oa_name** specifies the name of an OA in the *xmyConfigOP* file.
- **logical_process_name** is the name of the process as it appears in the **Responsibility** list of the OA in the *xmyConfigOP* file

If you do not supply an OA name, **xmyOM start** verifies that the **logical_process_name** appears in exactly one OA's responsibility list.

For example, the **logical_process_name xmySD1** in the example *xmyConfigOP* file in Figures 3-15 and 3-18 specifies the start command for the SD **SD1**. Therefore, to start **SD1**, execute the following:

```
xmyOM start xmySD1
```

The OM process reads the *xmyConfigOP* file, sending a message to an OA, which also knows the contents of the *xmyConfigOP* file. The OA executes the command and sends output back to the OM. The OM prints the output to the user and quits.

4.5.5 Starting Autostart Processes

The **autostart** subcommand (Appendix A.1.3.1) lets you have the OA on a specific host start all processes for that OA that have the **Autostart = yes**. The basic syntax for **xmyOM autostart** is

```
xmyOM autostart oa_name
```

where **oa_name** is the name of the OA whose **Autostart** processes you want to start.

For example, to start all Autostart processes for the OA **selene** (as defined in the example *xmyConfigOP* file in Figure 3-17), you would execute

```
xmyOM autostart selene
```

4.6 Stopping Processes

4.6.1 Stopping a Specific Process

The **stop** subcommand (Appendix A.1.3.7) lets you stop specific processes. The basic syntax for **xmyOM stop** is

```
xmyOM stop ?-o oa_name? logical_process_name
```

where

- **-o oa_name** specifies the name of an OA in the *xmyConfigOP* file.
- **logical_process_name** is the name of the process as it appears in the **Responsibility** list of the OA in the *xmyConfigOP* file.

If you do not supply an OA name, **xmyOM start** verifies that the **logical_process_name** appears in exactly one OA's responsibility list.

For example, the **logical_process_name** **xmySD1** in the example *xmyConfigOP* file in Figure 3-17 specifies the stop command for the SD **SD1**. Therefore, to stop **SD1**, you would execute the following:

```
xmyOM stop xmySD1
```

4.6.2 Stopping Autostart Processes

The **autostop** subcommand (Appendix A.1.3.2) lets you have the OA on a specific host stop all processes for that OA that have the **Autostart** = **yes**. The basic syntax for **xmyOM autostop** is

```
xmyOM autostop host_name
```

where **host_name** is the name of the host whose **Autostart** processes you want to stop.

For example, to stop all **Autostart** processes for the OA **selene** (as defined in the example *xmyConfigOP* file in Figure 3-17), execute

```
xmyOM autostop selene
```

xmyOM autostop shuts down the processes in the reverse order that they were started up. The OA continues to run after stopping the processes.

4.6.3 Stopping an OA and all Autostart Processes on the Local Host

The **xmyShutDown** command lets you gracefully terminate (stop and shut down) an OA on the local host. The syntax for **xmyShutDown** is

```
xmyShutDown
```

xmyShutDown first stops all **Autostart** processes the local OA is responsible for and then shuts down the local OA itself. This is in contrast to the **autostop** subcommand, which leaves the OA up and running.

4.6.4 Stopping an OA and all Autostart Processes on a Specific Host

The **shutdown** subcommand (Appendix A.1.3.6) lets you gracefully terminate (stop and shut down) an OA. The basic syntax for **xmyOM shutdown** is

```
xmyOM shutdown oa_name
```

where **oa_name** is the name of an OA in the *xmyConfigOP* file.

As with **xmyShutDown**, **xmyOM shutdown** stops an OA and all related processes, but does so for a specified host. For example, to stop the OA **selene**, you would execute

```
xmyOM shutdown selene
```

xmyOM shutdown first stops all **Autostart** processes the OA **selene** is responsible for and then shuts down the OA **selene** itself. This is in contrast to the **autostop** subcommand, which leaves the OA up and running.

4.6.5 Stopping an OA

In addition to **xmyOM shutdown**, which stops an OA and all of the **Autostart** processes the OA is responsible for, there is also the **xmyStopOA** command, which only stops the OA. All **Autostart** processes continue running.

The basic syntax for **xmyStopOA** is

```
xmyStopOA
```

xmyStopOA only takes the *?-hHR?* options listed in Section 3.4.2.2. In order to stop the OA on a particular machine, you must be on that machine.

4.6.6 Stopping and Restarting a Host

The **recycle** subcommand (Appendix A.1.3.5) lets you stop and restart the **Autostart** processes on a specific host. The basic syntax for **xmyOM recycle** is

```
xmyOM recycle host_name
```

where **host_name** is the name of the host whose **Autostart** processes you want to stop and restart.

This subcommand is useful to reinitialize log files that have grown very large.

For example, to stop and restart the **Autostart** processes for the OA **selene**, execute

```
xmyOM recycle selene
```

4.7 Obtaining Information About Processes

The MYNAH CLUI provides several subcommands to the **xmyOM** command for obtaining information on the MYNAH processes and configuration.

4.7.1 Obtaining the Status of Processes

The **status** subcommand (Appendix A.1.3.7) lets you get the status of specific processes. The basic syntax for **xmyOM status** is

```
xmyOM status ?-o oa_name? logical_process_name
```

where

- **-o oa_name** specifies the name of an OA in the *xmyConfigOP* file.
- **logical_process_name** specifies the name of the process as it appears in the **Responsibility** list of the OA in the *xmyConfigOP* file.

If you do not supply an OA name, **xmyOM start** verifies that the **logical_process_name** appears in exactly one OA's responsibility list.

For example, the **logical_process_name** **xmySD1** in the example *xmyConfigOP* file in Figure 3-17 specifies the status command for the SD **SD1**. Therefore, to obtain the status information for **SD1**, execute

```
xmyOM status xmySD1
```

The result takes the form:

```
xmyStatusSD: SD(SD1) is running: started by madmin, its pid is 6816
```

Another example would be to obtain the status information for the process **vxGateway** (which appears in Figure 3-17 in the responsibility list for the OA **luna**) by executing

```
xmyOM status -o luna vxGateway
```

The results take the form:

```
vxIpcMgr: vxErSrv00          on luna      .. selected. no action.  
vxIpcMgr: vxErrMsgClient000000 on luna  .. selected. no action.  
vxIpcMgr: vxErrMsgClient000001 on luna  .. selected. no action.  
vxIpcMgr: vxErrMsgClient000002 on luna  .. selected. no action.  
vxIpcMgr: vxErrMsgClient000003 on luna  .. selected. no action.  
vxIpcMgr: vxErrMsgClient000004 on luna  .. selected. no action.  
vxIpcMgr: vxErrMsgClient000005 on luna  .. selected. no action.  
vxIpcMgr: vxErrMsgClient000006 on luna  .. selected. no action.  
vxIpcMgr: vxErrMsgClient000007 on luna  .. selected. no action.  
vxIpcMgr: vxErrMsgClient000008 on luna  .. selected. no action.  
vxIpcMgr: vxErrMsgClient000009 on luna  .. selected. no action.  
vxIpcMgr: vxErrMsgClient000010 on luna  .. selected. no action.
```

4.7.2 Displaying Operability Configuration Settings

The **query** subcommand (Appendix A.1.3.3) displays the configuration settings from the *xmyConfigOP* file for OAs and the managed processes. The basic syntax for **xmyOM query** is

```
xmyOM query ?-o oa_name | -p logical_process_name | -s?
```

where

- **-o oa_name** specifies the name of an OA in the *xmyConfigOP* file.
- **-p logical_process_name** specifies the name of a process as it appears in the **Responsibility** list of the OA in the *xmyConfigOP* file.
- **-s** displays a list of all of the OAs defined in the *xmyConfigOP* file.

You can use only one of these options at a time, e.g., you can not use the **-s** option with the **-o oa_name** option.

For example, to see what OAs are defined in your installation, execute

```
xmyOM query -s
```

which generates output of the form:

```
selene  
luna
```

Then, if you wanted to see what processes the OA **selene** is responsible for, execute

```
xmyOM query -o selene
```

which generates output of the form:

```
The OA on selene is responsible for the following processes:  
vxIpcDir  
vxGateway  
vxIpcClean  
vxLogToFile  
vxErrorServer  
xmyTD  
xmyBD  
xmySD1
```

If you do not specify any options, **xmyOM query** lists the defined OAs in your configuration and the processes that appear in the **Responsibility** lists for those OAs in the *xmyConfigOP* file. This is equivalent to executing **xmyOM query -s** followed by **xmyOM query -o oa_name**, except that instead of seeing the defined processes for one OA only, you would see the defined processes for all OAs in the system.

For example, if you type

```
xmyOM query
```

and you have defined the OAs **selene** and **luna**, you would see output of the form:

```
-----  
The OA on selene is responsible for the following processes:  
vxGateway  
xmyBD  
-----  
The OA on luna is responsible for the following processes:  
vxIpcDir  
vxGateway  
vxIpcClean  
vxLogToFile  
xmyCollector  
vxErrorServer  
xmyTD  
xmyBD  
xmySD1  
xmySDwfal  
-----
```

To see settings for the process **vxIpcDir**, execute

```
xmyOM query -p vxIpcDir
```

which generates output of the form:

```
Process      : vxIpcDir  
StartCommand : /opt/SUNWmyn/mynah/bin/.xmyStartup  
StopCommand  : vxIpcDown -d  
StatusCommand: vxIpcProcesses  
Autostart    : Yes
```

4.8 Starting and Stopping MYNAH Software Packages

After installation, the `$XMYDIR/examples/admin/scripts` directory contains a start-up script for the various MYNAH software packages, e.g., `S99mynah.eg`.

To start these packages when the system is booted, copy this file from the `$XMYDIR/examples/admin/scripts` directory to the `/etc/rc3.d` directory (Solaris) or `/sbin/rc3.d` directory (HP-UX).

If you are using the Solaris platform and you want the software packages to terminate correctly when the system is rebooted, you can create logical links from the files in the `/etc/rc3.d` directory to the `/etc/rc0.d` directory, preappending the link with the kill prefix, `K`.

If you are using the HP-UX platform and you want the software packages to terminate correctly when the system is rebooted, you can create logical links from the files in the `/sbin/rc3.d` directory to the `/sbin/rc0.d` directory, preappending the link with the kill prefix, `K`.

NOTE — The owner of the start files (and the terminating links) must be **root**, and you must set the permissions on these files (and links) to 755.

4.8.1 Automatically Starting the MYNAH System

To automatically start-up the MYNAH and Telexel processes at boot time, perform the following tasks for the appropriate operating system.

4.8.1.1 Starting on the Solaris Platform

1. Copy the example start-up file `S99mynah.eg` from the `$XMYDIR/examples/admin/scripts` directory to the `/etc/rc3.d` directory and rename the copy `S99mynah`.

NOTE — See Appendix [B.1.5](#) for a copy of `S99mynah.eg`.

2. Edit `S99mynah` for library paths, hostport, hostname, etc.
3. Create a logical link to `S99mynah` in `/etc/rc0.d` to terminate the MYNAH processes correctly at reboot time, e.g.,

```
cd /etc/rc0.d
ln -s /etc/rc3.d/S99mynah K01mynah
```


4.8.1.2 Starting on the HP-UX Platform

1. Copy the example start-up file *S99mynah.eg* from the *\$XMYDIR/examples/admin/scripts* directory to the */sbin/rc3.d* directory and rename the copy *S990mynah*.

NOTE — See Appendix B.1.5 for a copy of *S99mynah.eg*.

2. Edit *S990mynah* for library paths, hostport, hostname, etc.
3. Create a logical link to *S990mynah* in */sbin/rc0.d* to terminate the MYNAH processes correctly at reboot time, e.g.,

```
cd /sbin/rc0.d
ln -s /sbin/rc3.d/S990mynah K001mynah
```

4.8.2 Automatically Starting Oracle

To automatically start-up the Oracle processes on a client machine at boot time, perform the following tasks for the appropriate operating system.

4.8.2.1 Starting on the Solaris Platform

1. Copy the example start-up file *S96oracle.eg* from the *\$XMYDIR/examples/admin/scripts* directory to the */etc/rc3.d* directory and rename the copy *S96oracle*.

NOTE — See Appendix B.3.1 for a copy of *S96oracle.eg*.

2. Edit the *S96oracle* file to correct directories and paths, etc.
3. Create a logical link to *S96oracle* in */etc/rc0.d* to terminate the Oracle processes correctly at reboot time, e.g.,

```
cd /etc/rc0.d
ln -s /etc/rc3.d/S96oracle K02oracle
```

4. It is recommended that you change the **shutdown** command in */opt/SUNWora/oracle/bin/dbshut* from **shutdown** to **shutdown immediate**.

4.8.2.2 Starting on the HP-UX Platform

1. Copy the example start-up file *S96oracle.eg* from the *\$XMYDIR/examples/admin/scripts* directory to the */sbin/rc3.d* directory and rename the copy *S960oracle*.

NOTE — See Appendix B.3.1 for a copy of *S96oracle.eg*.

2. Edit the *S960oracle* file to correct directories and paths, etc.
3. Create a logical link to *S960oracle* in */sbin/rc0.d* to terminate the Oracle processes correctly at reboot time, e.g.,

```
cd /sbin/rc0.d  
ln -s /sbin/rc3.d/S960oracle K002oracle
```

4. It is recommended that you change the **shutdown** command in */opt/HPUXora/oracle/bin/dbshut* from **shutdown** to **shutdown immediate**.

4.9 User Defined Processes

If users want to add their own processes to the *xmyConfigOP* file, their start, stop, and status commands must follow these rules:

- They must exit with a 0 return code upon success.
- They must exit with a non-zero on failure. Any output to standard out or standard error will be included in the reply to the OM from the OA that ran this command.
- These commands must exit after performing their task. That is, when starting up a process, they must exit when that process is “up” and running by whatever definition makes sense for that process. They can NOT stay around or the OA will hang, waiting for a response from the command in terms of its exit.

4.10 Operability Summary

An administrative user needs to be able to start, stop, or determine the status of any MYNAH application process, and any platform process that a MYNAH application process relies on. The operability design allows a user, from a single user interface and a single host machine, to perform this function.

The flexibility of the configuration portion of the operability design allows the MYNAH Administrator to add new types of managed processes as the need arises.

Only one process for each host needs to be explicitly started at boot time, the OA process. The OA process will start everything else that should be started at boot time.

The Operability feature is summarized in Figure 4-2.

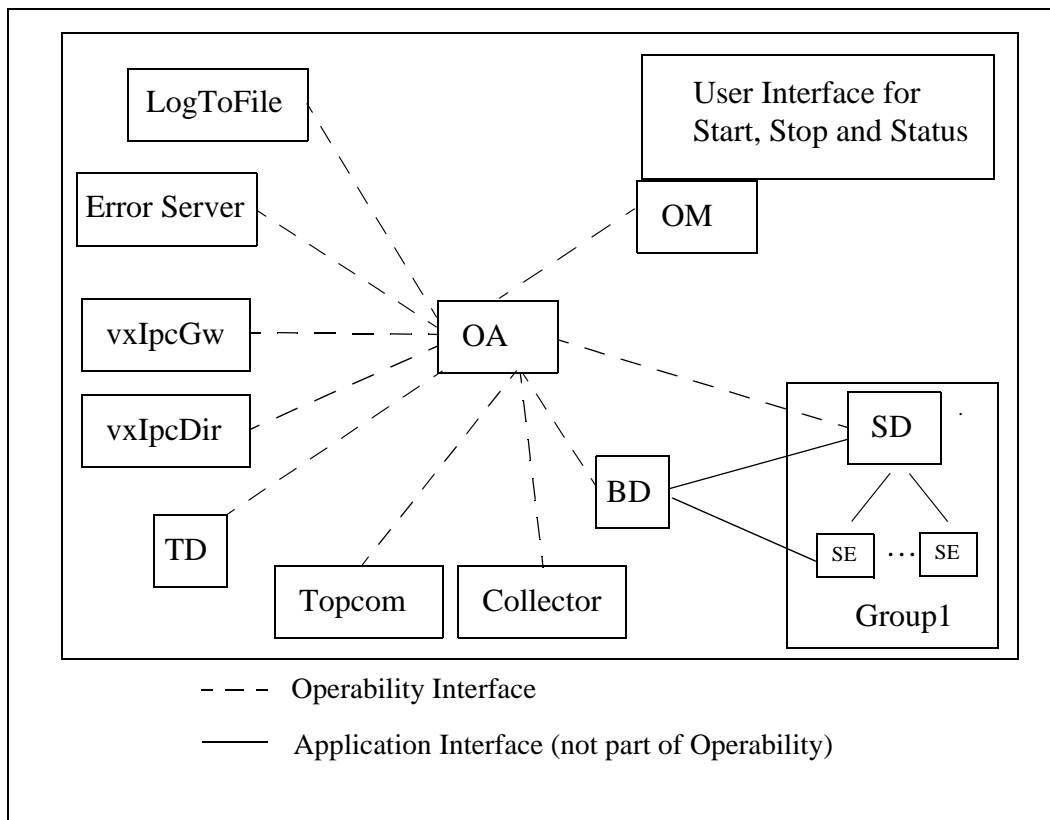


Figure 4-2. Operability Feature Summary

5. MYNAH Administrative Tasks

This section contains information needed for administering the MYNAH System. This includes dynamically increasing and decreasing the number of SEs in an SE Group, becoming familiar with Telexel™ processes, and monitoring the system.

NOTE — It is assumed that you are familiar with UNIX and Oracle.

5.1 The `xmyCmd` Command

The MYNAH CLUI's `xmyCmd` command accepts a series of subcommands that help you administer the MYNAH System. The following subsections describe how to use these administrative subcommands, while Appendix [A.1.2](#) contains complete descriptions of each subcommand.

NOTE — The following subsections contain examples of how to use the administrative `xmyCmd` subcommands. Some of these subcommands generate output to **stdout** while others do not. When appropriate, each example will contain example output.

In addition, all lines you would type will begin with **saline:** to signify the UNIX prompt.

All administrative `xmyCmd` subcommands accept the following options:

- v** Generates verbose responses.
- h** Displays a brief help message.

All administrative `xmyCmd` subcommands that interact with an SD accept the following options:

- d *sd_name*** Specifies the SD name that is the subject of the command. If this option is not used, the value of the XMYSD variable is used. If this variable is not set, then the default SD name defined in the `xmyConfig` file is used.
- t *time-out*** Specifies the time-out interval. The CLUI waits for this interval to get the responses from SD. If a value of 0 is used, the CLUI waits indefinitely until all the responses are received. By default, the time-out is 30 seconds.

5.2 User Enum Values

The MYNAH System can be tuned to meet specific user organization needs. This tuning is accomplished by changing the set of legal values, called **User Enum Values**, for specific fields.

These value lists populate drop-down menus in the GUI. For example, Figure 5-1 shows the User Enum Values for the **Type** drop down menu on a SUT Object Properties Window.

NOTE — While this drop-down menu is labeled **Type** on a SUT Object, in the MYNAH database the field name of this menu is labeled **Level**. When adding User Enum values for this menu, as described in Section 5.2.1, you must use the field name **Level**, and not **Type**.

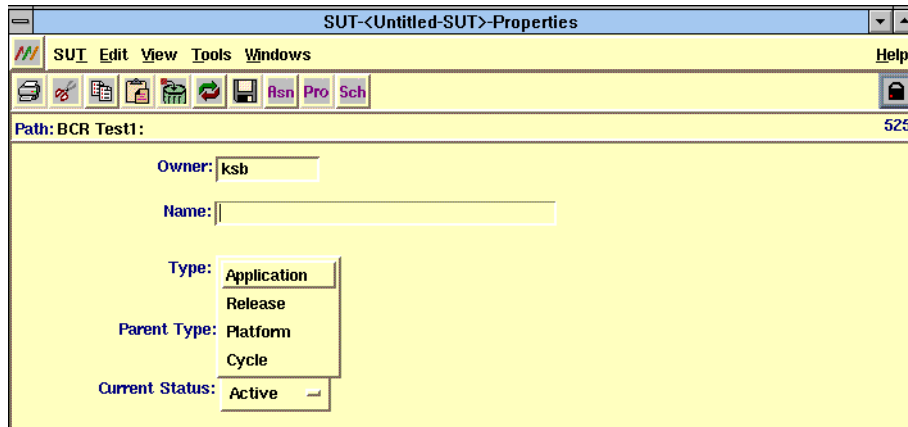


Figure 5-1. SUT Object's Type Drop-down Menu User Enum Values

Table 5-1 lists the User Enum values delivered with the MYNAH System.

Table 5-1. Delivered User Enum Values

Object	Field	Short Values	Delivered Long Values
Data	Type	GEN	General
Doc	Type	GEN	General
Doc	Type	SPEC	Specification
Doc	Type	DES	Design
Doc	Type	TP	Test Plan
Issue	Status	OP	Open
Issue	Status	CL	Closed
Issue	Type	GEN	General

Table 5-1. Delivered User Enum Values

Object	Field	Short Values	Delivered Long Values
Issue	Type	ENV	Environment
Issue	Type	SOFT	Software
Keyword	Type	GEN	General
Procedure	Type	GEN	General
Result	Reason	AS	Ana Success
Result	Reason	AU	Ana Unsuccess
Result	Reason	AI	Ana Inconclusive
Result	Reason	SF	Script Failure
Result	Reason	SC	Script Cancel
Result	Reason	CG	Compare Good
Result	Reason	CF	Compare Failure
Result	Reason	CW	Compare Warn
Result	Reason	KF	Child Failed
Result	Reason	KW	Child Warn
Result	Reason	LI	Limbo
Result	Reason	BL	Blocked
Result	Reason	DF	Deferred Failure
Requirement	Type	GEN	General
Step	Type	GEN	General
Step	Type	INI	Initialization
Step	Type	VAL	Validation
Step	Type	CL	Cleanup
SutInfo	Level	APP	Application
SutInfo	Level	REL	Release
SutInfo	Level	PLA	Platform
SutInfo	Level	CYC	Cycle
SutInfo	Status	ACT	Active
SutInfo	Status	IDL	Idle
TestVersion	Type	AREA	Area
TestVersion	Type	FEAT	Feature
TestVersion	Type	TEST	Test
TestVersion	Type	TC	Test Case

NOTE — **Data**, **Doc**, and **Procedure** Objects are not supported in MYNAH 5.0 and do not appear on any GUI drop down menu. However, these objects do appear in the User Enum listing created by **xmyCmd addEnum Value -q** (Section 5.2.1).

5.2.1 Adding User Enum Values

Your organization may wish to provide additional User Enum values. For example, you may want users to be able to assign a Type of **Application** to an **Issue**. Another possibility is that you want **Phase** as a possible Type for **SutInfo** objects.

The MYNAH CLUI command **xmyCmd** has a subcommand, **addEnumValue**, that lets you make these changes on behalf of the user community.

The basic **addEnumValue** subcommand syntax is

```
xmyCmd addEnumValue -t type -s string -l string \
                    -d true/false ?-q?
```

where:

- **-t type** is an Object-Field pair shown in Table 5-1.
- **-s string** is a short description of the newly introduced value.
- **-l string** is a long description of the newly introduced value.
- **-d true/false** specifies if this is the default value for this Enum Value Object/Field pair.
- **-q** creates a list of the existing User Enum values.

For example, to add the User Enum value **Application** for the Object **Issue** and the Field **Type**, execute

```
saline: xmyCmd addEnumValue -t issueType -s APP -l Application -d true
```

In this example you add the value **Phase** for the Object **SutInfo** and the field **Type**.

```
saline: xmyCmd addEnumValue -t sutInfoLevel -s PHS -l Phase -d false
```

NOTE — Remember, in the GUI this is the **Type** field, but in the database the field name is **Level**.

For a complete explanation of the **addEnumValue** subcommand, see Appendix A.1.2.1.

5.3 Obtaining Current System and User Statistics

The following subsections explain the steps for changing several MYNAH parameter values for an SD or a specific user, such as specifying the number of scripts that can be executed at any given time. Before you change a parameter, you must first know the current parameter values. The MYNAH CLUI contains several subcommands to the **xmyCmd** command, **sysstats** and **usrstats**, that let you see the current values for these parameters.

NOTE — These subcommands are available to all MYNAH users. Therefore, complete descriptions of the **sysstats** and **usrstats** subcommands are located in the *MYNAH System Users Guide*. You can also obtain help messages for these subcommands, for example by typing

```
xmyCmd sysstats -h
```

5.3.1 Obtaining an SDs Global Parameter Values

The **sysstats** subcommand lets you see the current global parameter values for an SD. The **xmyCmd sysstats** syntax is

```
xmyCmd sysstats ?-h? ?-d sd_name? ?-t time-out? ?-v?
```

For example, to see the current global parameter values for the default SD defined in the *xmyConfig* file, you would execute

```
saline: xmyCmd sysstats
SD(SD1): SD Program statistics on 09/07/96 at 11:47:32:
SD(SD1)'s parameters are as follows:
  Name: SD1
  Start Time: 09/05/96 at 07:35:56
  Overall SD Concurrency: 30
  Default User Conc.: 5
  Default User Priority: 2
  SD(SD1) is running with the MYNAH database
```

5.3.2 Obtaining a User's Parameter Value Statistics

The **usrstats** subcommand lets you see the current parameter values for MYNAH users.

The **xmyCmd usrstats** syntax is

```
xmyCmd usrstats ?-h? ?-d sd_name? ?-t time-out? ?-v? login_id
```

where **login_id** is the login name of the user whose parameter values you want to see. If you omit this option, **xmyCmd usrstats** returns the parameter values for all users.

For example, to see the current parameter values for the user **kjd**, assuming **kjd** is using the default SD defined in the *xmyConfig* file, you would execute

```
saline: xmyCmd usrstats kjd
SD(SD1):
User          Pri-   Max   Actual   Tests   Tests   Tests
              ority  Conc   Conc    Queued  Active  Paused
ksb           2     5     5         0       0       0
```

To see the current parameter values for all users using the default SD defined in the *xmyConfig* file, you would execute

```
saline: xmyCmd usrstats
User          Pri-   Max   Actual   Tests   Tests   Tests
              ority  Conc   Conc    Queued  Active  Paused
soll          2     5     5         0       0       0
root          2     5     5         0       0       0
rapheal       2     5     5         0       0       0   (admin)
pt03          2     5     5         0       0       0
madmin        2     5     5         0       0       0   (admin)
pt04          2     5     5         0       0       0
pt01          2     5     5         0       0       0
pt05          2     5     5         0       0       0
ralphs        2     5     5         0       0       0
pt06          2     5     5         0       0       0
wynnd         2     5     5         0       0       0
kjd           2     5     5         0       0       0
```

The difference between MaxConc and ActualConc is that MaxConc is what the administrator sets, and ActualConc is manipulable by the user. Users can set their ActualConc anywhere from 0 up to the MaxConc setting.

NOTE — The administrator cannot change a user's ActualConc. However, if the administrator sets a user's MaxConc to a value below his/her ActualConc, the SD automatically reduces the user's ActualConc to be equal to the new MaxConc value.

For more information, see [Section 16.2.10.6](#) of the *MYNAH System Users Guide*.

5.4 Concurrency Levels

The MYNAH CLUI contains several subcommands to the **xmyCmd** command that let you set **concurrency** levels for an SD, which are the number of scripts that can be executed at any given time. When you change the parameter, that change will last across runs of the SD (i.e., If the SD is brought down and then back up, the change will remain in effect no matter how many times it is brought down and up.

5.4.1 Setting System Concurrency Limits

The **sysconc** subcommand (Appendix [A.1.2.10](#)) lets you set the overall concurrency limit for a system. The **xmyCmd sysconc** syntax is

```
xmyCmd sysconc ?-d sd_name? ?-t time-out? ?-v? value
```

where **value** is the concurrency limit you want to set for the system.

NOTE — The new overall concurrency limit remains in effect for all future runs of the SD until such time that you use **xmyCmd sysconc** to set a new system concurrency limit.

For example, to allow the default SD (as defined in the *xmyConfig* file) to execute 50 scripts at a time, you would execute

```
saline: xmyCmd sysconc 50
```

NOTE — Use **xmyCmd sysstats** to see the system's current concurrency limit.

5.4.2 Setting Concurrency Limits for a Specific User

The **usrmaxconc** subcommand (Appendix A.1.2.12) lets you set the concurrency limit for a specific user. The **xmyCmd usrmaxconc** syntax is

```
xmyCmd usrmaxconc -u login_id ?-d sd_name? ?-t time-out? \  
?-v? value
```

where:

- **login_id** is the id of the user whose concurrency limit you want to set
- **value** is the concurrency limit you want to set for the user.

NOTE — The new concurrency limit remains in effect for the user for all future runs of the SD until such time that you use **xmyCmd usrmaxconc** to set a new user concurrency limit.

For example, to allow the user *kjd* to run nine scripts at one time, you would execute

```
saline: xmyCmd -u kjd usrmaxconc 9
```

NOTE — Use **xmyCmd usrstats** to see the user's current maximum concurrency limit.

5.4.3 Setting Concurrency Limits for New Users

The **dfltusrconc** subcommand (Appendix A.1.2.4) lets you set the default concurrency limit for new users. The **xmyCmd dfltusrconc** syntax is

```
xmyCmd dfltusrconc ?-d sd_name? ?-t time-out? ?-v? value
```

where **value** is the new default concurrency limit.

NOTE — The new default concurrency limit remains in effect for all news users for all future runs of the SD until such time that you use **xmyCmd dfltusrconc** to set a new default concurrency limit.

For example, to allow new users to run eight scripts at one time (using the default SD defined in the *xmyConfig* file), execute

```
saline: xmyCmd dfltusrconc 8
```

5.5 Working with Script Queues

When users submit scripts to the background, the submitted scripts are placed in a queue, waiting for execution. The MYNAH CLUI provides several subcommands to the **xmyCmd** command for administering the queues.

5.5.1 Checking the Queues on the Local Host

The **checkQueues** subcommand (Appendix A.1.2.3) checks if the queues on the current host (defined in the *xmyConfig* file) are full or nearly full. The **xmyCmd checkQueues** syntax is

```
xmyCmd checkQueues ?-h? ?-thresh percent? ?-v?
```

where *percent* is the percentage of the total queue space in use on the system; above this threshold a warning is produced.

By default, **checkQueues** uses a warning threshold of 90%, which means that it checks to see if 90% of the available queues in the system are in use, and if so it produces a warning message. For example, to use the default threshold level, execute

```
saline: xmyCmd checkQueues  
queues ok
```

where *queues ok* means that the usage of the queues was below the default 90% threshold level.

Add the **-v** option for more information, as in

```
saline: xmyCmd checkQueues -v  
total number of queues in use: 71  
total number of queues in system: 200  
total number of bytes in use: 255  
total number of bytes in system: 131072  
percentage of queues used: 35.5%  
percentage of bytes used: 0.2%  
number of queues waiting to receive: 17  
queues ok
```

Use the **-thresh** option to specify a different threshold level. For example, to see if 33% or more of the available queues are being used, execute

```
saline: xmyCmd checkQueues -thresh 33  
WARNING: # of queues used is 42% of max: 84/200
```

5.5.2 Checking the Queues on all Hosts

The **checkAllQueues** subcommand (Appendix [A.1.2.2](#)) checks if the queues on all systems in a MYNAH configuration are full. The **xmyCmd checkAllQueues** syntax is

```
xmyCmd checkAllQueues ?-h?
```

Therefore, to check all queues, execute

```
saline: xmyCmd checkAllQueues
```

```
checking queues on selene, please wait...  
queues ok
```

```
checking queues on mimir, please wait...  
queues ok
```

5.6 Setting Queuing Priority Levels

When users start the MYNAH System, they are assigned a default **queuing priority** value, which determines the order in which scripts are executed when they are submitted to the background. Scripts from users of the same queuing priority level are executed in the order they are submitted to the SD.

Queuing priority adheres to the following standards:

- The lower the number the higher the priority.
- 0 is the lowest queuing priority value and the highest priority.
- The recommended upper limit of a queuing priority value is 999.

The MYNAH CLUI provides several subcommands to the **xmyCmd** command for setting queuing priority levels

5.6.1 Setting Queuing Priorities for New Users

The **dfltusrpri** subcommand (Appendix A.1.2.5) lets you set the default queuing priority for new users. The **xmyCmd dfltusrpri** syntax is

```
xmyCmd dfltusrpri ?-d sd_name? ?-t time-out? ?-v? value
```

where **value** is the new default queuing priority value.

You can set the default user queuing priority only for a specific SD. If you do not specify an SD, **xmyCmd dfltusrpri** defaults to the default SD specified in the general default entry.

NOTE — The new default queuing priority remains in effect for all news users for all future runs of the SD until such time that you use **xmyCmd dfltusrpri** to set a new default queuing priority.

For example, to set the queuing priority for new users using the default SD to level 2, you would execute

```
saline: xmyCmd dfltusrpri 2
```

NOTE — Use **xmyCmd sysstats** to see the default queuing priority for each SD.

5.6.2 Setting Queuing Priorities for Specific Users

The **usrpriority** subcommand (Appendix A.1.2.13) lets you change the queuing priority for a specific user. The **xmyCmd usrpriority** syntax is

```
xmyCmd usrpriority -u login_id ?-d sd_name? \  
?-t time-out? ?-v? value
```

where:

- **login_id** is the if of the user whose queuing priority value you want to set
- **value** is the queuing priority value you want to set for the user.

NOTE — The new default queuing priority remains in effect for all the user for all future runs of the SD until such time that you use **xmyCmd dfltusrpri** to set a new user queuing priority.

For example, to set the queuing priority for the user *kjd* to level 1, you would execute

```
saline: xmyCmd usrpriority -u kjd 1
```

NOTE — Use **xmyCmd usrstats** to see the user's current queuing priority.

5.7 Changing the Number of SEs in an SE Group

The number of engines in an SE Group are initially set in the *xmyConfig* file. However, there may be times when you want to change the number of available SEs. The user community may want to execute more scripts than what can be executed by the number of SEs in the *xmyConfig* file. Rather than bringing down the system, editing the *xmyConfig* file, and restarting the system, the MYNAH CLUI provides several subcommands to the **xmyCmd** command for increasing and decreasing the number of available SEs without stopping the system.

NOTE — For these subcommands you don't enter the number of SEs you need but the number of extra or fewer SEs you need from the current configuration.

5.7.1 Increasing the Number of SEs

The **seincr** subcommand (Appendix A.1.2.7) lets you increase the number of SEs in a particular SE Group. The **xmyCmd seincr** syntax is

```
xmyCmd seincr ?-d sd_name? ?-t time-out? ?-v? \  
-e se_group_name delta_value
```

where:

- *se_group_name* is the SE Group whose number of SEs you want to increase.
- *delta_value* is the number of *extra* SEs you need.

For example, if the SE Group **SeGp1** was configured with six engines but you need ten, execute

```
saline: xmyCmd seincr -d Sd1 -e SeGp1 4
```

5.7.2 Decreasing the Number of SEs

The **sedecr** subcommand (Appendix A.1.2.6) lets you decrease the number of SEs in a particular SE Group. The **xmyCmd sedecr** syntax is

```
xmyCmd sedecr ?-d sd_name? ?-t time-out? ?-v? \  
-e se_group_name delta_value
```

where

- *se_group_name* the SE Group whose number of SEs you want to decrease.
- *delta_value* is the number of *fewer* SEs you need.

For example, let's assume you no longer need the ten SEs for SE Group **SeGp1** you set in Section 5.7.1, but instead of the six configured in the *xmyConfig* file, you need eight, execute

```
saline: xmyCmd sedecr -d Sd1 -e SeGp1 2
```

5.8 Setting Administrative Privileges

When the MYNAH System is installed, administrative privileges are granted to the user **admin**. However, you may need to grant these privileges to another user, such as when you're on vacation.

If your MYNAH installation uses a database (and therefore the MYNAH GUI), you can grant and remove administrative privileges using the Person Object (Section 6). If your installation does not use a database (and you do not have access to the MYNAH GUI), you can still use the following subcommands to the **xmyCmd** CLUI command.

NOTE — These subcommands do not make any permanent changes to the MYNAH database. Therefore, use them only if your installation does not use a database.

5.8.1 Granting Administrative Privileges

The **setadm** subcommand (Appendix A.1.2.9) grants administrative privileges to a specified user. You can grant this user administrative privileges only for a specific SD. If you do not specify an SD, **xmyCmd setadm** will default to the SD specified in the *xmyConfig* file. The **xmyCmd setadm** syntax is

```
xmyCmd setadm ?-d sd_name? ?-t time-out? ?-v? login_id
```

where *login_id* is the user to whom you wish to grant administrative privileges.

For example, to grant the user *kjd* administrative privileges for the SD **SD2** (assuming the default SD is **SD1**) you would execute

```
saline: xmyCmd setadm -d SD2 kjd
```

5.8.2 Removing Administrative Privileges

The **unsetadm** subcommand (Appendix A.1.2.11) removes administrative privileges from a specified user. If you do not specify an SD, **xmyCmd setadm** defaults to the SD specified in the *xmyConfig* file. The **xmyCmd unsetadm** syntax is

```
xmyCmd unsetadm ?-d sd_name? ?-t time-out? ?-v? login_id
```

where *login_id* is the user whose administrative privileges you wish to remove.

For example, to remove the administrative privileges you granted to the user *kjd* in Section 5.8.1, execute

```
saline: xmyCmd unsetadm -d SD2 kjd
```

5.9 Returning the Status of SE Groups

The **sestat** subcommand (Appendix A.1.2.8) returns the status of the SEs in a particular group. The **xmyCmd sestat** syntax is

```
xmyCmd sestat ?-d sd_name? ?-t time-out? ?-v? ?se_group?
```

where *se_group* is the name of a specific SE Group whose status you wish to display.

You may request the status of the SEs controlled by one SD only. If you do not specify an SD, **xmyCmd sestat** defaults to the SD specified on the *xmyConfig* file.

The **xmyCmd sestat** subcommand returns one line listing

- The SE Group name
- What host the group resides on
- The number of SEs in the group
- The number of unstarted SEs in the group
- The number of busy SEs in the group
- How many kills (using the **xmyCmd cancel** subcommand) are pending
- The number of scripts to be executed
- The run mode of the SE.

For example, to see the status of all SE Groups for the default SD, you would execute

```
saline: xmyCmd sestat
SD(SD1):
SeGroup      Resides      Total Unstarted Busy Pending Executed Run
  Name        On           SEs      SEs  SEs  Kills  Scripts Mode
SeGp2         luna         1         0   0    0      0 connstate
SeGp1(df1t)   luna         3         0   0    0      0 stateless
```

You can add the **-v** option get to more information, as in

```
saline: xmyCmd sestat -v
SD(SD1):
SeGroup      Resides      Total Unstarted Busy Pending Executed Run
  Name        On           SEs      SEs  SEs  Kills  Scripts Mode
SeGp2         luna         1         0   0    0      0 connstate
  SE xmySE0000SD1: state=available, status-on-startup='ready'
SeGp1(df1t)   luna         3         0   0    0      0 stateless
  SE xmySE0003SD1: state=available, status-on-startup='ready'
  SE xmySE0002SD1: state=available, status-on-startup='ready'
  SE xmySE0001SD1: state=available, status-on-startup='ready'
--->BD Status as it is known to SD(SD1):
BD-host      State  Comments
-----
luna         up    07/26/96 07:46:25 Ping succeeded
```

5.10 Creating New MYNAH Configurations

During installation, one directory is `$XMYHOME`, this will be the directory containing the MYNAH configuration files. As your installation grows, however, you may wish to have more than one `$XMYHOME` directory (i.e., more than one MYNAH configuration) working off the same software (i.e., working off the same `$XMYDIR` directory).

To help you create new configuration directories, the MYNAH System includes the program **xmyMkHome**. **xmyMkHome** prompts you for the directory to be the new `$XMYHOME`. **xmyMkHome** automatically enters the current value of `$XMYHOME` as the default location of the new `$XMYHOME`, as shown in the following:

```
saline: xmyMkHome
Where should XMYHOME be installed? [/opt/SUNWmyn/mynah]
```

Simply type the new location for the configuration directory. **xmyMkHome** then creates this directory and all necessary subdirectories. In addition **xmyMkHome** copies the example configuration files from the `$XMYDIR/examples/config` directory into the new configuration directory's `config` directory, stripping off the `.eg` suffix during the copy.

NOTE — **xmyMkHome** does not change the definition of `$XMYHOME`. This must be done in the `xmyProfile` or `xmyLogin` file. **xmyMkHome** creates a new directory that *can* be used as a new `$XMYHOME`.

5.11 Removing Locks on Database Objects

When you open a MYNAH GUI object, e.g., a Person Object, the system creates a lock file in the database, signifying that the object is being used. The GUI removes locks when you bring it up, but if the person who owns the lock is gone and you want to remove a lock on a particular row of an object table, you can use the **xmySdbRemoveLock** program to remove the lock.

The syntax of the **xmySdbRemoveLock** command is

```
xmySdbRemoveLock ?-d table_name object_id? ?-l user??
```

where:

- d *table_name object_id*** Causes the lock to be removed from the given object id of the given table.
- l *user*** Causes a list to be generated and written to **stdout** showing all locks owned by the specified *user*.

If *user* is not specified, this causes a list to be generated and written to **stdout** showing all locks owned by the user invoking **xmySdbRemoveLock**.

5.12 Monitoring Processes

Table 5-2 contains brief descriptions of the Telexel tools that can be used to monitor all MYNAH and MYNAH related processes. For complete descriptions, see the Telexel manual pages.

Table 5-2. Required Telexel Processes

Process	Description
vxIpcDir	Directory name server for all processes. There is one server per configuration. The vxIpcDir daemon is needed for the Telexel interprocess communication subsystem to function. It maintains a map of the names of registered processes to their message queue id, process id, host id, and owner.
vxIpcClean	Cleans up the channel that is left when there are process core dumps.
vxIpcMgr	Used to select processes to take specified actions
vxIpcProcesses	Displays a sorted list of registered Telexel processes.

NOTE — All of these tools have an **-h** (help) option.

If you notice that **vxIpcProcesses** displays information about processes that cannot possibly exist (for example processes that are on a host that you know is down), then you need to use **vxIpcMgr** to force the cleaning up of these defunct entries. e.g.,

```
vxIpcMgr -m 'dione' -s a -a f
```

will remove all channel names for processes that apparently are running on the host 'dione'.

5.13 Monitoring the MYNAH System

After the MYNAH System is installed, one of your main tasks is monitoring the MYNAH and Telexel processes. The MYNAH processes begin with `xmy`, and Telexel processes begin with `vx`.

5.13.1 MYNAH Processes

The valid MYNAH process types are:

SD	Script Dispatcher
SE	Script Engine
BD	Boot Daemon
TD	Trigger Daemon
GU	Graphical User Interface
SH	Command Line Tcl Shell
AD	Command Line Administration
DO	Command Line Do
OA	Operability Agent
OM	Operability Manager

5.13.2 Monitoring the System Status

You can use `xmyOM`'s `status` subcommand or `vxIpcProcesses` to monitor the system.

5.13.2.1 Using `xmyOM status`

`xmyOM status` returns the status of MYNAH OAs and logical processes, and takes the form:

```
xmyOM status ?-o oa_name? logical_process_name
```

where:

-o oa_name	Specifies the name of an OA configured in the <code>xmyConfigOP</code> file.
logical_process_name	Specifies the name of a process defined in the <code>xmyConfigOP</code> file.

xmyOM status runs the command specified by the process's **Status** parameter.

For example, you know the *xmyConfigOP* file defines an OA called **luna**, and you know this OA has responsibility for the Telexel Error Server process called **vxErrorServer**. To see the status of this process, you could type

```
saline: xmyOM status -o luna vxErrorServer
```

You would get output of the form:

```
IPC Registered Processes
ID                               PID      HOST      QUEUE    USER
==                               ===      ====      =====  =====
vxErrorServer                   98       luna      3602     madmin
```

You can also use **xmyOM query** (Section 4.7.2 and Appendix A.1.3.3) to see what OAs and processes are defined in your system, and then use **xmyOM status** to see the status for a process. For example, you could type

```
saline: xmyOM query
```

which would generate output of the form:

```
-----
The OA on selene is responsible for the following processes:
vxGateway
xmyBD
-----
The OA on luna is responsible for the following processes:
vxIpDir
vxGateway
vxIpClean
vxLogToFile
xmyCollector
vxErrorServer
xmyTD
xmyBD
xmySD1
-----
```

To see the status of the **vxGateway** process on the OA **selene**, you could type

```
saline: xmyOM status -o selene xmyBD
```

which would generate output of the form

```
saline: xmyStatusBD: BD(selene) is running: started by madmin, its pid is 1906
```

Then, to see the status of the **vxGateway** process on the OA **selene**, you could type

```
saline: xmyOM status -o selene vxGateway
```

which would generate output of the form

```
vxIpcMgr: vxErSrv00          on luna      .. selected. no action.  
vxIpcMgr: vxErrMsgClient000000 on selene   .. selected. no action.  
vxIpcMgr: vxErrMsgClient000001 on selene   .. selected. no action.  
vxIpcMgr: vxErrMsgClient000002 on selene   .. selected. no action.  
vxIpcMgr: vxErrMsgClient000003 on selene   .. selected. no action.  
vxIpcMgr: vxErrMsgClient000004 on selene   .. selected. no action.  
vxIpcMgr: vxErrMsgClient000005 on selene   .. selected. no action.  
vxIpcMgr: vxErrorServer      on selene   .. selected. no action.  
vxIpcMgr: vxLogDestFile      on luna     .. selected. no action.  
vxIpcMgr: xmyBDselene        on selene   .. selected. no action.  
vxIpcMgr: xmyGU0000          on luna     .. selected. no action.  
vxIpcMgr: xmyGU0001          on selene   .. selected. no action.  
vxIpcMgr: xmySDSD1           on luna     .. selected. no action.  
vxIpcMgr: xmyTDLuna          on luna     .. selected. no action.
```

See Appendix [A.1.3.7](#) for further discussion on **xmyOM status**.

5.13.2.2 Using vxIpcProcesses

vxIpcProcesses returns a list of all IPC processes that are currently running. This list contains the process name, process id, host name, message queue id, and the user name of the person who registered the process.

You can enter a MYNAH process as the input to **vxIpcProcesses** to see the information for that process, such as:

```
vxIpcProcesses GU  
IPC Registered Processes  
ID          PID      HOST      QUEUE  USER  
==          ==      ==      =====  ==  
xmyGU0000   7045   ahimsa    1900    stw  
xmyGU0001   27135  selene    7811    hsb  
xmyGU0002   9050   selene    7230    lpetro  
xmyGU0003   23825  selene    6450    ralpm  
xmyGU0004   9122   selene    4477    ksb
```

NOTE — This returns the information for all MYNAH GUI processes.

You can use **grep** to filter the list. For example, to see processes only for a particular user, type

```
saline: vxIpcProcesses|grep kjd  
vxErrMsgClient0000028    6249  selene  19    kjd  
xmyGU0004                9122  selene  4477  kjd
```

5.14 Monitoring MYNAH Log Files

All MYNAH processes perform administrative and error logging through the Telexel Logger. This includes

- Start up information
- Shut down information
- Errors.

The Telexel Logger facility is available to all MYNAH processes. It logs all MYNAH errors messages in one location, called the SystemLog, which is *\$XMYHOME/syslog/adminLog*.

The Telexel Logger facility allows actions to be specified for particular messages (e.g., e-mail notification to the administrator).

The Telexel Logger provides a tool called **vxFilterLogFile** for filtering information in the log. The basic syntax for **vxFilterLogFile** is

```
vxFilterLogFile log_filename
```

NOTE — The MYNAH Telexel log file is in *\$XMYHOME/syslog/adminLog*. If you set the **VXLOGFILES** environment variable to this file, the *log_filename* parameter is automatically entered for you.

This basic version of **vxFilterLogFile** prints every message in the log file to standard output. However, several field specifiers let you select which messages to print. Table 5-3 list the field specifier IDs and their related values.

Table 5-3. vxFilterLogFile Field Specifier IDs

Specifier ID	Value
e	errorCode
t	Time according to process originating the message (local time zone)
T	Time according to originating process in the time zone of originating process
c	Time the record was actually logged by vxLogToFile (local time zone)
C	Time record was actually logged in the time zone of originating process
z	Time zone this report is being generated in
Z	The time zone of the process originating the message
s	Log message severity
h	Host name of the machine running the process

Table 5-3. vxFilterLogFile Field Specifier IDs

Specifier ID	Value
d	Domain name of the machine running the process
p	Process ID of the process
P	Parent process ID
u	Name of the user who owns the process
U	Name of the effective user ID
g	Name of the group that owns the process
G	Name of the effective group ID
N	Name of the process's executable
M	Text of the error message

Logical relation operators are used to specify regular expression string matches.

NOTE — See Sections 4.2.2.2 and 4.2.2.3 of the *MYNAH System Scripting Guide*, for explanations of the logical relation operators. See Appendix A.45 of the *MYNAH System Scripting Guide*, for an explanation of regular expressions.

For example, to see the messages for the user **wyndd**, type

```
saline: export VXLOGFILES=$XMYHOME/syslog/adminLog
saline: vxFilterLogFile "u=wynnd"
1-XMY-SE-0000:1 08/12/96 15:36:17 wynnd@ariel pid=16855
    Started
1-XMY-SE-0000:1 08/12/96 15:43:04 wynnd@ariel pid=16873
    Started
1-XMY-SE-0002:1 08/12/96 17:06:40 wynnd@ariel pid=16855
    Stopped
```

The beginning string, e.g., 1-XMY-SE-0000, specifies what process created the log entry. In this case, the XMY means this log was produced by the MYNAH System, and the SE indicates the process type, which in this case means the entry was created by an SE.

See the **vxFilterLogFile** manual page for a explanation of how to use the filter. i.e., type

```
saline: man vxFilterLogFile
```

5.15 Deleting Message Files

The Tcl administrative method **xmyMsgDel** (Appendix A.2.1) lets you delete messages in the Application to Application Message Directories.

NOTE — Remember, **xmyMsgDel** must be executed using a Tcl interpreter, such as **xmytclsh** or the Script Builder. In addition, you can include **xmyMsgDel** in scripts that can be executed using the **xmyCmd submit** command

The **xmyMsgDel** method syntax takes one of the following forms:

```
xmyMsgDel -topcom topcom_name ?-all? \  
          ?-timeBef timestring? ?-timeAft timestring?  
  
xmyMsgDel -printcom printcom_name ?-all? \  
          ?-timeBef timestring? ?-timeAft timestring?  
  
xmyMsgDel -handler handler_name ?-subDir subDir? ?-all? \  
          ?-timeBef timestring? ?-timeAft timestring?
```

where:

- all** Deletes all message files from the Message Directories.
- timeBef *timestring*** Specifies the time before which the message files should be deleted.
- timeAft *timestring*** Specifies the time after which the message files should be deleted.
- subDir *subDir*** Indicates the subdirectory containing the messages to be deleted. If this attribute is not present, message files from all subdirectories are deleted.

NOTE — This attribute is valid with the **-handler** form only.

In this example, all messages for the handler **app_1** after January 1, 1997 and before January 1, 2000 will be deleted for all subdirectories.

```
saline: xmytclsh  
> xmyMsgDel -handler app_1 -timeAft "1 Jan 97" \  
           -timeBef "1 Jan 2000"
```

In the following example, all messages for the Topcom handler **top_1** before September 22, 1997, 11:00 AM will be deleted.

```
saline: xmytclsh  
> xmyMsgDel -topcom top_1 -timeBef "22 Sep 1997 11:00:00"
```

This time, all messages for the Printcom handler *print_1* will be deleted.

```
saline: xmytclsh  
> xmyMsgDel -printcom print_1 -all
```

5.16 Cleaning Queues

If you see the error code

```
errorCode 1-1P-0004
```

execute the command

```
vxIpcMgr -af
```

to clean up the queues.

This error code may be generated when the MYNAH System attempts to create a channel to a SD but it fails, which usually occurs after UNIX crashes. Files in */usr/tmp/telexel* remain and **xmyIpcDir** can't clean them up because the process that was associated with the process id may now be used by another UNIX process, not a Telexel process.

NOTE — You can also manually also remove the files from the */usr/tmp/telexel* directory.

6. Integrating Other Testing Tools

Integration of the MYNAH System with a specific set of third-party GUI testing tools is covered in [Section 3.3.2.4](#). This section describes how you can integrate any additional third-party or home-grown tools.

This mechanism is useful in those cases where it would be advantageous to manage, run, and track results for non-MYNAH test scripts from the MYNAH System.

6.1 Wrapper Script Basics

In normal operation, the SE receives an execution request containing the name of the MYNAH script to run. When the script begins executing, its name is available to it in the global Tcl variable `xmyVar(ScriptName)`.

NOTE — See Section 7.15 of the *MYNAH System Scripting Guide*, for information on the `xmyVar(ScriptName)` variable.

If the script exists in the MYNAH database, a record of its run is stored in a Runtime object and, optionally, Result object(s).

When using the mechanism described in this section, however, an SE is configured to run a special wrapper script when it receives an incoming execution request rather than the script whose name was passed in the execution request. The SE **ExecScript** configuration parameter names the special wrapper script. Engines configured this way never directly run any script other than the one specified by the **ExecScript** parameter for the **Engine** entry in the `xmyConfig` file ([Section 3.3.3.1](#)).

When an engine of this type receives an incoming execution request, it begins executing the wrapper script, which has available to it all the same information that a regular script would have, including the name of the script passed in the execution request. There are no restrictions on the content of the wrapper script since it is an arbitrary Tcl script. Your MYNAH installation contains an example of a wrapper script in the file `$XMYDIR/demo/scripts/wrapper.tcl`. This example wrapper script “tests” UNIX shell scripts by running a mock-up of a shell script testing tool on the non-MYNAH script name passed in `xmyVar(ScriptName)`.

An effective wrapper script does the following:

1. It either starts the third-party or home-grown testing tool or accesses an already-running instance of the tool.
2. It provides the tool with the name of the script to run, which is available in `xmyVar(ScriptName)`.

3. Optionally, it provides the tool with the directory in which to place any output it produces.
4. When the tool has finished running the non-MYNAH script, the wrapper script examines the results and translates them into MYNAH concepts, using functions such as **xmyUpdateResult** and **xmyCompare** (Sections 6.2.22 and 6.2.3 of the *MYNAH System Scripting Guide*, respectively). The example script *wrapper.tcl* increments the count of good or failed compares based on the result of the third-party tool's execution.

If the non-MYNAH script exists as a Script Object in the MYNAH database, then running it causes the creation of a Runtime Object and, optionally, Result Object(s), just as if it were a MYNAH script. The Script Object contains a language attribute that can be set to "others" to indicate that the script is not a MYNAH Tcl script.

Your sample MYNAH database contains a script named *shell1* that demonstrates this.

6.2 Example

This example assumes that you use a test tool called **trun** and that you have scripts that this tool runs.

The following describes the steps involved in integrating this tool into the MYNAH System.

1. Write a Tcl wrapper script using the guidelines above. Basically, your wrapper script will invoke the **trun** tool, giving **xmyVar(ScriptName)** as an argument.
2. Configure a Script Engine Group that will be used for these scripts. Set the **ExecScript** parameter to the name of your wrapper script.
3. Create Script Objects to represent your existing scripts (or have your user community do this).
 - Set the Script Engine Group entry in these objects to the name of the Group that you created in Step 2.
 - Set the Language entry to *others* to indicate that the scripting language is something other than MYNAH Tcl.

If there are a large number of these scripts you may wish to use the **xmyCreateScriptObject** tool to create the script objects. (See [Section 16.2.14](#) of the *MYNAH System Users Guide* for information the **xmyCreateScriptObject** tool.)

Once these steps are completed, any user can continue to create, run, or analyze these scripts from the MYNAH GUI or from the CLUI. They will do these things in the same way as they would for any MYNAH script.

7. The Person Object

The Person Object is a MYNAH GUI object that lets you, the MYNAH Administrator, control the MYNAH database settings for the MYNAH users. These functions include creating Person Objects and assigning **Inactive** status to a user.

NOTE — See the *MYNAH System Users Guide* for information on using the MYNAH GUI.

A Person Object must exist in the database for each MYNAH user. A Person Object is created in one of two ways:

- If you set the **Welcome New Users** parameter in the *xmyConfig* file to **yes** (see [Section 3.3.1](#)), the GUI automatically creates a Person Object when a new user starts the GUI. In fact, if the **Welcome New Users** parameter is set to **no**, the GUI will exit.

Therefore, setting the **Welcome New Users** parameter to **no** lets you prevent nonauthorized users from using the MYNAH GUI.

- You can create a Person Object in the GUI before the new user starts the GUI. This way, if you set the **Welcome New Users** parameter to **no** (e.g., to limit the number of new users) you can create a Person Object for a specific user before she/he starts the GUI.

NOTE — There can be only one Person Object in the database per UNIX ID.

MYNAH users can edit their general properties attributes (i.e., their name, phone number, and e-mail address) and the keywords associated with only their Person Object. As a MYNAH Administrator, however, you can change these attributes for other users. In addition, you can edit the authority attribute for other users.

NOTE — Users cannot create Person Objects, but they can access the Person Objects for all MYNAH users and edit their own Person Object from the Database Browser.

7.1 Creating a Person Object

In the MYNAH Window, execute

MYNAH->New ->Person

as shown in [Figure 7-1](#).

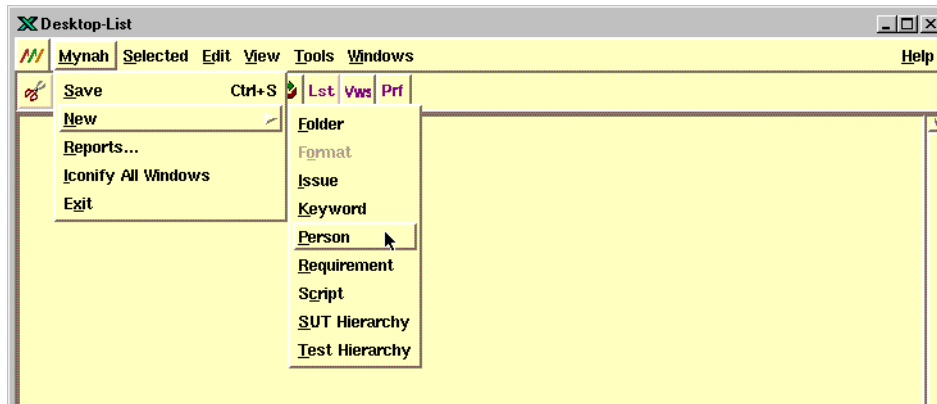


Figure 7-1. Creating a New Person Object

7.1.1 Editing Properties Attributes

Once you've created a Person Object, you can enter user information. Open the new Person Object by either double clicking on the object icon or by selecting the icon and then executing

Selected->Open

An empty Person Object opens. By default, the Person Object will open in the Properties View, as shown in [Figure 7-2](#).

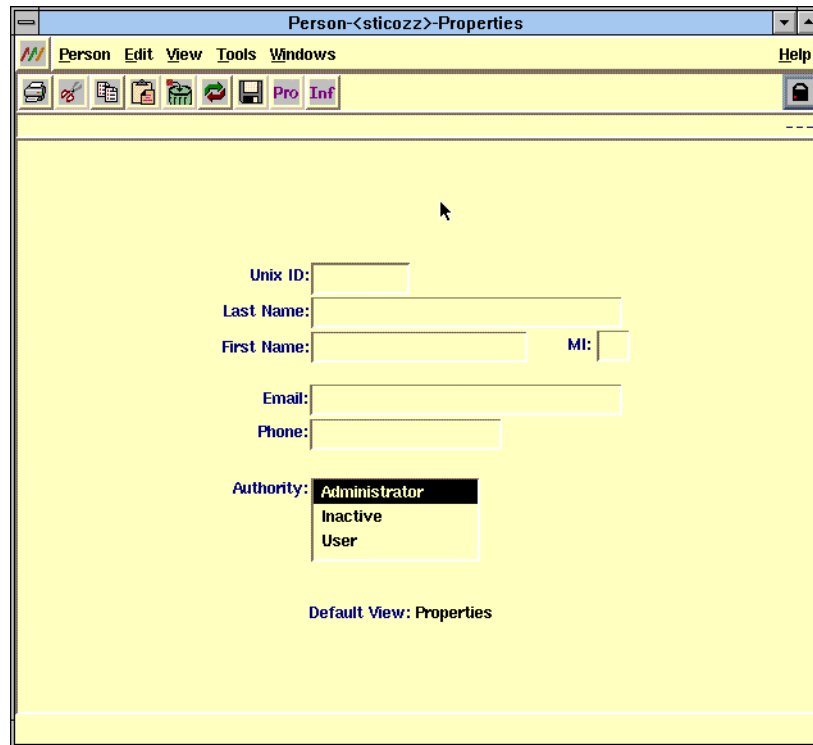


Figure 7-2. New Person Object Properties View

To edit the contents of the Person Object, you must first unlock the window by clicking on the lock icon in the upper right-hand corner. The word EDIT appears in the status area (the bottom right -and corner of the window), signifying that the window is in Edit Mode. You can now enter the user information in the fields in the window.

Enter the user's UNIX ID, name, e-mail address, and phone number. As a MYNAH Administrator, you can also set the user's authority level by selecting one of the three following options in the Authority list:

- Administrator** This gives administrative privileges to a user. This is equivalent to **xmyCmd**'s **setadm** method (see Section 5.8.1 and Appendix A.1.2.9). An administrative user can grant other users Administrator Authority. In addition, an Administrator can edit the attributes of other users.

Inactive

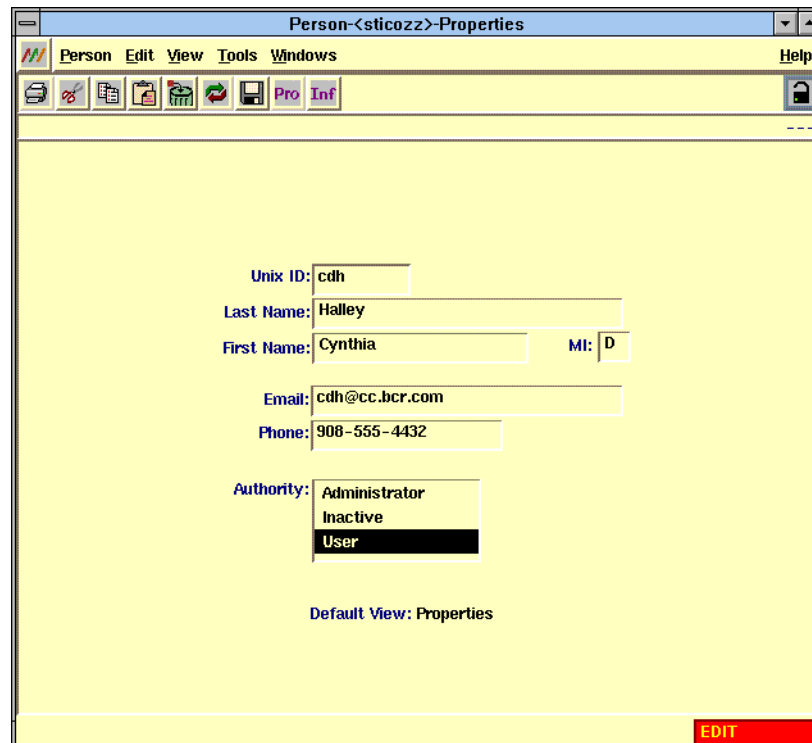
This gives a user inactive status, meaning that the user can no longer access the MYNAH System. The Person Object remains in the database. If you want to reactivate a user, use the **Database Browser** to grant the user either Administrator or User Authority.

NOTE — You cannot cut a Person Object from the database. Giving a Person Object inactive status is the only way of removing MYNAH privileges for a user, e.g., if a user leaves.

User

This gives the user the standard User Authority, such as when you are removing administrative privileges. Someone with User Authority can edit her/his attributes, but cannot change their Authority. In addition, someone with User Authority can view but cannot edit the Person Objects for other users.

Figure 7-3 shows an example of an edited Person Object Properties View.



The screenshot shows a window titled "Person-<sticozz>-Properties" with a menu bar (Person, Edit, View, Tools, Windows, Help) and a toolbar. The main area contains the following fields:

- Unix ID: cdh
- Last Name: Halley
- First Name: Cynthia
- MI: D
- Email: cdh@cc.bcr.com
- Phone: 908-555-4432
- Authority: Administrator, Inactive, User (User is selected)

At the bottom, it says "Default View: Properties" and has a red "EDIT" button.

Figure 7-3. Edited Person Object Properties View

The Person Object is not added to the database until you save it. Execute

Person->Save

The window exits Edit Mode. The Authority field is grayed out.

At a minimum, you must enter the User ID for a new Person Object. You can then:

- Fill in the rest of the fields on the Properties View.
- Save the object.
- Close the window by executing

Person->Close

The system asks you if you want to save your changes if you have not already done so.

- Change to the Information View to edit Keyword associations for the Person Object.

There can be only one Person Object in the database per UNIX ID. If you try adding a Person Object with a UNIX ID that already has a Person Object, the error dialog box in Figure 7-4 appears.

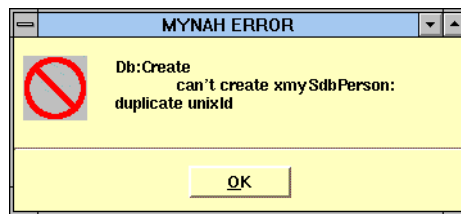


Figure 7-4. Duplicate UNIX ID Error Box

If you've edited a Person Object and then try closing it without first saving it, the dialog box in Figure 7-5 appears.

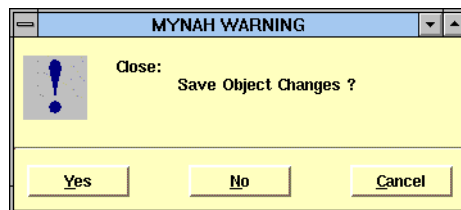


Figure 7-5. Close Dialog Box

- If you click on **Yes**, your changes are saved and the Person Object window closes.
- If you click on **No**, your changes are discarded and the Person Object window closes.
- If you click on **Cancel**, the Person Object window remains open.

7.1.2 Editing Information Attributes

You can associate Keywords with a Person Object. While a user usually edits Keyword associations for her/his Person Object, you can edit the associations for all users.

NOTE — See the *MNAH System Users Guide* for information on Keywords.

To associate Keywords with a Person Object:

1. Execute

View->Change View ->Information

The window in Figure 7-6 appears. Click on the **Lock** icon to unlock the window.

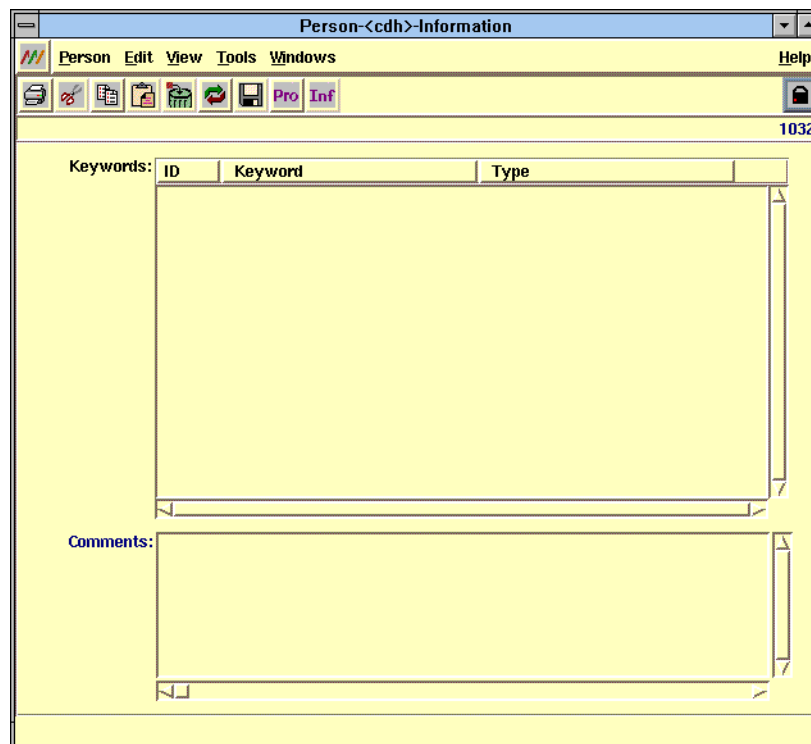


Figure 7-6. Person Object Information View

2. Execute

Tools->Database Browser

The Database Browser (Figure 7-7) appears. If the list of Keyword Objects is not visible, select **Keyword Object** from the **Database Object** pop-up menu.

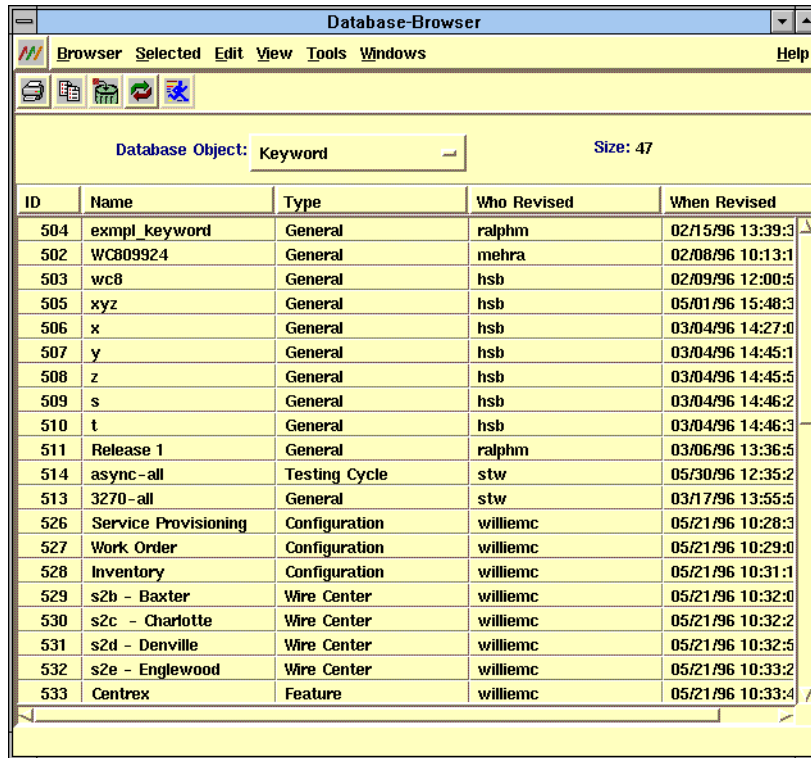


Figure 7-7. Database Browser - Keyword Object

3. Select the Keyword you want by clicking on the Keyword, and execute

Edit->Copy

4. Move the mouse back to the Person Object. Bring the Keywords box into **focus**, i.e., selecting it and making it ready for you to paste the keyword. (See Figure 7-8.)

To focus the Keyword box:

- If there are no Keywords already pasted in the box, click on the ID field name.
- If Keywords have already been added to the Person Object, click on any Keyword.

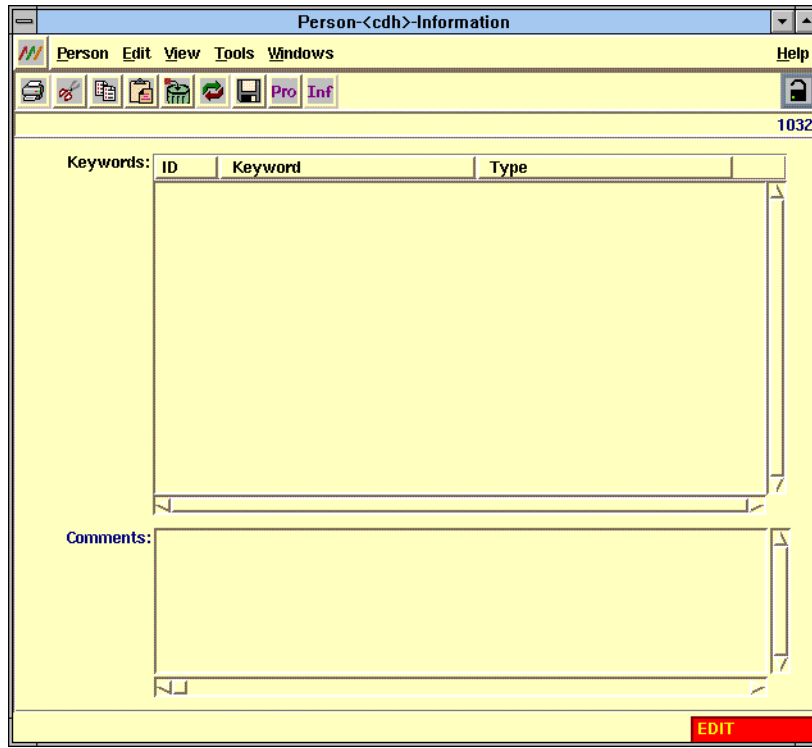


Figure 7-8. Properties View with Keyword Box in "Focus"

5. Execite

Edit->Paste

The Keyword you copied is pasted into the Keyword box (Figure 7-9). You can save/close the window or add more Keyword associations.

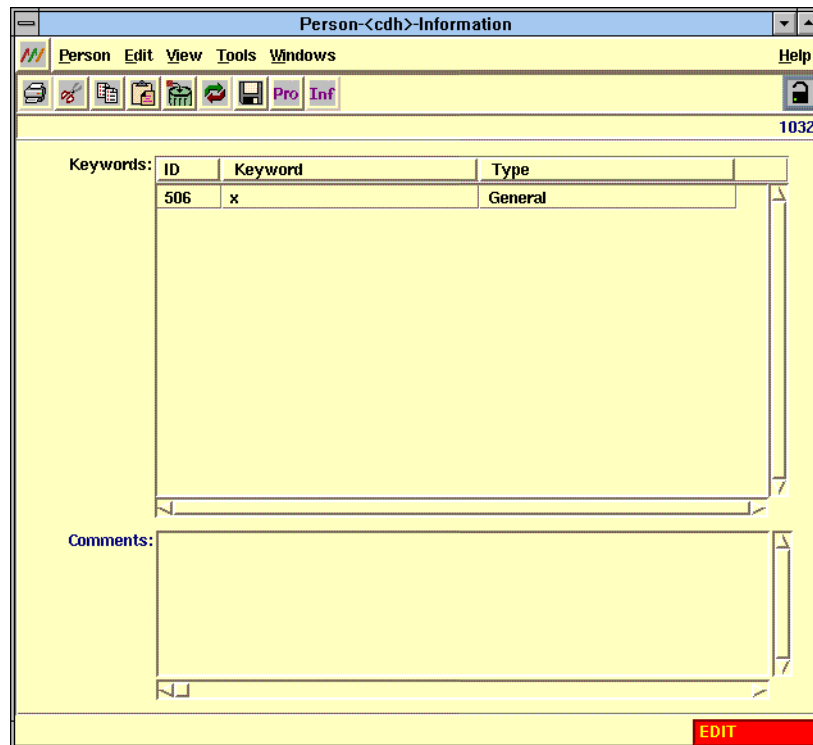


Figure 7-9. Properties View with Keyword Added

7.2 Editing an Existing Person Object

As a MYNAH Administrator, you can edit Person Objects for other users. Your main reason for editing a user's Person Object is to change their Authority, e.g., to grant or remove administrative privileges.

WARNING — You can remove administrative privileges for *all* users, including yourself.

While you can create Person Objects in your MYNAH window, all Person Objects for a MYNAH installation are listed in the Database Browser. In fact, if the **Welcome New Users** parameter in the *xmyConfig* file is set to **yes**, the resulting Person Objects can be

found *only* in the Database Browser. Therefore, you will most often want to edit a user's Person Object from the Database Browser.

In our example, if you wish to apply inactive status to a user who was automatically added, follow these steps.

1. Execute

Tools->Database Browser

If the list of Person Objects is not visible when the Browser appears, select Person from the Database Object menu to access the list of Person Objects (Figure 7-10).

ID	First Name	Last Name	Unix ID	Phone #
518	adam	irgon	adam	
514	ann	roemerman	mgt02	699-7598
507	carolyn	knight	cam	(908) 699-2657
1032	cynthia	halley	cdh	908-555-4432
522	David	Camman	dwc	
1028	elaine	adams	ema	908 699-5807
516	george	horruitiner	gha1	908-699-2382
5	heidi	bruhin	hsb	908 699-8052
1024	Jagdeep	Shiruru	jagdeep	
1023	jatinder	kaur	jkaur	7274
6	jochen	wunn	wunn	908 699 5753
511	kenneth	berczik	ksb	(908) 699-8437
523	lou	smith	lous	699-2040
506	louis	petro	lpetro	908 699-2539
3	mad	administrator	madmin	
1025	mary	mclaughlin	mtm	(908) 699-8183
1031	min	madpt	madminpt	699-6861
508	nancy	mond	nem2	699-5106
1027	Nelson	Fung	nfung	699-7063
509	rachel	barlev	rachel	699-6861

Figure 7-10. Database Browser - List of Person Objects

2. Open the Person Object you want to edit. You can do this by either of the following:

- Clicking on the row listing the object and executing

Selected->Open

- Double clicking on the row listing the object.

- When the Person Objects opens, unlock the window, and select the **Inactive** option from the **Authority** list (Figure 7-11).

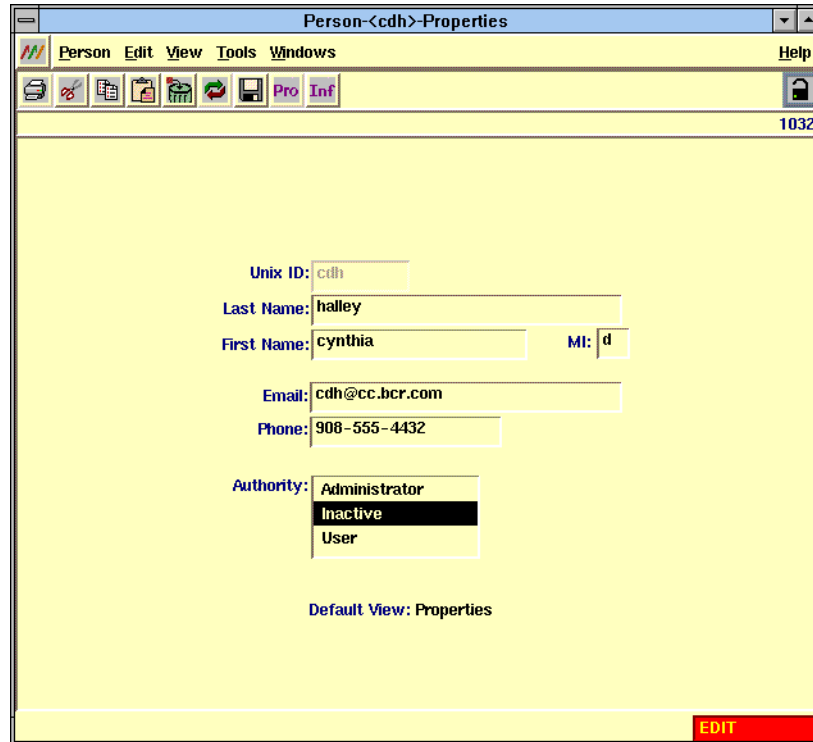


Figure 7-11. Edited Person Object — Granting Administrative Privileges

- Save and close the object.
- If you look at the Database Browser, you will not see any change in the Authority for the Person Object. To view the change, execute

View->Refresh

The Database Browser is redisplayed, and the user is now listed as Inactive (See Figure 7-12).

The screenshot shows a window titled "Database-Browser" with a menu bar (Browser, Selected, Edit, View, Tools, Windows, Help) and a toolbar. Below the toolbar, it displays "Database Object: Person" and "Size: 33". The main area contains a table with the following data:

ID	First Name	Last Name	Unix ID	Phone #
518	adam	irgon	adam	
514	ann	roemerman	mgt02	699-7598
507	carolyn	knight	cam	(908) 699-2657
1032	cynthia	halley	cdh***INACTIVE****	908-555-4432
522	David	Carman	dwc	
1028	elaine	adams	ema	908 699-5807
516	george	horruitiner	gha1	908-699-2382
5	heidi	bruhin	hsb***ADMIN***	908 699-8052
1024	Jagdeep	Shiruru	jagdeep	
1023	jatinder	kaur	jkaur	7274
6	jochen	wunn	wunn	908 699 5753
511	kenneth	berczik	ksb***ADMIN***	(908) 699-8437
523	lou	smith	lous	699-2040
506	louis	petro	lpetro***ADMIN***	908 699-2539
3	mad	administrator	madmin***ADMIN***	
1025	marymary	mclaughlin	mtm	(908) 699-8183
1031	min	madpt	madminpt	699-6861
508	nancy	mond	nem2	699-5106
1027	Nelson	Fung	nfung	699-7063
509	rachel	barlev	rachel	699-6861

Figure 7-12. Refreshed Database Browser

8. Setting up Tag Name Files

This section describes the structure of *Tag Name* file and how to set them up.

8.1 Introduction

Tag name processing is one the Term3270 Package methods of referring to a screen location where scripting statements are written using user-defined labels called **tags** that reference locations on a screen in place of row and column integer values. These definition reside in files called **Tag Name** files. Multiple scripts can reference these files. If the format of the screen changes, only the definitions of the tags related to the screen need to be updated.

The MYNAH script accesses the appropriate *Tag Name* files to determine the row and column coordinates associated with the tag names in the script.

8.2 Tag Names

A tag name is a user-defined name that is associated with a field on a format/screen. Each tag name entry contains

- The user-defined tag name
- The row and column coordinate values of the point at which the field begins
- The length of the field
- Optionally, any additional information such as any display properties at the row and column location or the actual field that appears at these coordinates.

A tag name and its associated row, column, and field length information can be used to reference a field where you can enter data, a format/screen field label, such as **login**, that is used by the format/screen to identify where you enter input, and any other location contained on the format/screen. Tag names generally identify fields on a format/screen where you enter text, thus, the term tag name is sometimes replaced with the term field name.

For example:

```
x_login 2 18 5
```

is a tag name called **x_login** that references a location at row 2 and column 18, and the length of the field is 5 characters.

Tag names and their associated format/screen location data are defined and stored by you (i.e., the MYNAH Administrator) in a UNIX file called a **Tag Name** file. A *Tag Name* file

identifies the tag names and associated format/screen location data for a single format/screen.

8.2.1 Storing Tag Name Files

The Term3270 Package *xmyConfig* entry contains a parameter called *TagDir*. The value assigned to this parameter is used as the directory containing the *Tag Name* files. (See [Section 3.3.2.2](#) for more information on the Term3270 Package *xmyConfig* entry.) The default location of *TagDir* is *\$XMYHOME/data/tagdir*.

A separate *Tag Name* file must be created for each format/screen that will be accessed.

Any user can create *Tag Name* files. The user can then create a directory containing the files, specifying that directory as the option value for the Term3270 Package *TagDir* parameter.

8.2.2 Using Tag Name Files

Tag Name files are loaded into MYNAH scripts by using the Term3270 **format** method. (See Section 9, “Term3270 Extension Package” the *MYNAH System Scripting Guide*, for information on the **format** method.) For example, if there is a *Tag Name* file called *MainMenu*, to load this file into script you could type

```
$3270_conn format MainMenu
```

where **\$3270_conn** is the handle to a Term3270 connection.

All of the Term3270 Package extensions support tag name location processing as well as row/column and label location processing. (See Section 9.3, “Term3270 Location Processing” of the *MYNAH System Scripting Guide* for information on Term3270 location processing. Assuming the *MainMenu* file we loaded contains a tag name called **login**, you can move the cursor to that field represented by this tag name by entering

```
$3270_conn fieldNext -tag login
```

in a script.

8.3 Tag Name Table Format and the Tag Name File

The tag names and their associated data are written in a *Tag Name* file in the format of a table. The tag name table consists of four or more fields of user-provided data. The fifth field and any additional fields are currently not used but may be used in the future. The tab character or any number of spaces are used as the delimiter between all data fields except between the row and the column field, where a tab character, spaces, or a period character can be used.

The structure of a tag name table entry can be one of the following:

- tag name<tab>row<tab>col<tab>length[<tab>optional fields]

for example

```
date 2      70      8      field
```

- tag name<tab>row.col<tab>length[<tab>optional fields]

for example

```
time 3.68 10      field
```

NOTE — Remember, the period can *only* be used between the row and column fields.

The name of the *Tag Name* file must be the name of the format/screen the tag name table describes. This name is called the screen ID.

The screen ID is also referenced by the *Screen Identification* file (Section 9), which the GUI uses to identify each format/screen in the applications(s) that you test. The *Screen Identification* file must be available before users create a connection to a 3270 application. This file is often created by the MYNAH administrator, but users can create their own *Screen Identification* file. The *Screen Identification* file is automatically loaded when you establish a 3270 connection

The *Screen Identification* file contains regular expression patterns (as defined for the **regex(3x)** program) in conjunction with screen location. As a user navigates through an application, the *Screen Identification* file determines if a field matching a pattern is found at a particular location on the current screen. Once this format is determined, the **format** statement is generated, and the **format** is then used to load a *Tag Name* file of the same name.

If a format/screen name contains embedded blank characters, the Script Builder generates format statements with quotation marks around the name, e.g., "TASK 10". You must create the *Tag Name* file with quotation marks surrounding format/screen names, too.

The tag name table must be delimited by **BEGIN** and **END** labels like those shown below. These labels indicate where the tag name table begins and ends. Thus, the tag name table can be embedded within a file that you maintain for other purposes.

```
BEGIN TAG NAME TABLE  
  
END TAG NAME TABLE
```

Keep these points in mind when you create files containing tag name tables.

- **BEGIN** and **END** labels are required.
- Between the **BEGIN** and **END** labels, lines containing only a newline character or lines with a first character of # are treated as comments and ignored.

- Text lines contained in a tag name table cannot exceed 511 characters.
- Duplicate tag names are not permitted in the tag table. If duplicates are created, the last entered tag value pair will be used.
- Tag names cannot exceed 127 characters.

An example of a *Tag Name* file appears in [Figure 8-1](#) (The optional field in this example is being used to store 3270 attribute information.)

```

*** PRINTOUT OF FORM AS DEFINED WITH FIELD CHARACTERS
-----
          1          2          3          4          5          6          7          8
1234567890123456789012345678901234567890123456789012345678901234567890
-----
1:
2:  EXMP01                      EXMP ENTITY SELECTION SCREEN                      06/04/96 08:58
3:
4:
5:
6:
7:
8:
9:                      ENTITY      :
10:
11:                      RECONNECT:  _
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23: Copyright (C) 1989, 96  TELCORIDA,  ALL RIGHTS RESERVED.
24:
-----
NAMES:  EXMP01  EXMP01
BEGIN TAG NAME TABLE
zzsdatl  66      8      field,APNMNN
zzstime  75      5      field,APNMNN
title   2      26     28     label,XPXXXX,EXMP ENTITY SELECTION SCREEN
ent     9      41     1      field,XUXXXX
recon  11     41     1      field,XUXXXX
zzsmg  24     2      79     field,AUHMNN
END TAG NAME TABLE
(VC)

```

Figure 8-1. Example Tag Name File

NOTE — The screen shown in [Figure 8-1](#) is not necessary and in fact will be ignored when the file is loaded into a script since only the lines of the tag name

table delimited by the **BEGIN** and **END** labels are actually loaded into the script.

As you can see from [Figure 8-1](#), the tags refer to fields on the screen. There is a tag named *recon* at row 11 and column 41. This position corresponds to the entry field for the RECONNECT label.

8.4 Enabling Tag Name Processing

Once you've placed the *Tag Name* files in the directory to be used as *TagDir*, you must enter the path to this directory as the argument for the **TagDir** parameter in a *Term3270* entry in the *xmyConfig* file.

When using the MYNAH GUI, you can enable tag name processing by selecting the Location Processing **Tagname** setting on the GUI's Script Builder 3270 Terminal Settings window. See the *MYNAH System Users Guide*, for information on the Script Builder.

If you're creating a script to be run from the command-line or background, you can simply enter the tags as argument for location processing.

NOTE — Scripts written in row/column, label, or tag name processing will load and execute whether or not the Script Builder **Tagname** setting has been selected. However, for scripts written using tag names to load and execute successfully the path to the directory containing the *Tag Name* file must be defined.

8.5 Testing Multiple Releases and/or Multiple Applications

If you want to test multiple releases or applications you can create a *TagDir* for each release/application and create a new *Term3270* connection entry for each *TagDir*. You can then assign each *Term3270* connection to a different SE. The users can now send their scripts to the SE controlling the connection that access the *TagDir* for their release/application.

8.6 Undefined Tag Name Processing

Scripts coded using tag names may contain tag names that are undefined, i.e., they don't exist in the current screen definition files. This condition can occur when a script is used to test multiple releases of an application and the script contains tag name statements that reference a tag name that exists in one release of the application but not in other release(s) of the application.

9. The Screen Identification File

This section of the guide describes how to create and maintain an ASCII file called a *Screen Identification* file, which the GUI uses to identify each format/screen in the application(s) that you test. The *Screen Identification* file makes it possible to generate format/screen names automatically. By entering the location of *Screen Identification* as the parameter for the **ScreenIdentificationFile** parameter of a **Term3270** entry in the *xmyConfig* file, the specified *Screen Identification* file is automatically loaded when you establish a 3270 connection.

NOTE — The Screen Identification File is processed only if Tagname Location Processing is selected.

9.1 Why the Screen Identification File is Needed

The *Screen Identification* file is needed to identify the format/screen the user is on and to generate a format statement during an initial script capture session using the GUI's Script Builder.

The *Screen Identification* file contains regular expression patterns (as defined for the **regex(3X)** program) in conjunction with screen location. As a user navigates through an application, the *Screen Identification* file determines if these two conditions exist

- The format/screen name exists on the format/screen being displayed.
- A regular expression can be defined to uniquely describe the format/screen name

If these conditions determine that a string matching a pattern is found at a particular location on the current screen, the **format-pattern** command ([Section 9.2.2.1](#)) identifies that portion of the screen that matches the pattern as the format for the screen. Once this format is determined, the **format** statement is generated, and the **format** is then used to load a *Tag Name* file of the same name.

9.2 About the Screen Identification File

Automatic identification of the format/screen name is possible if you provide the users a single ASCII *Screen Identification* file that contains format/screen names from the application that they access.

The *Screen Identification* file contains two sections.

The first section contains the *Screen Identification* file formatting commands that determine the regular expression patterns that are used to describe the format/screen name. The formatting commands are described in [Section 9.2.2](#).

The second section contains data lines used to uniquely describe a format/screen. Requirements for the data lines are in [Section 9.2.1](#).

Each format/screen name must appear on a line by itself. The name is followed by one or more lines of data that differentiate that format/screen from any other format/screen in your application.

The data line(s) you provide the users contain a string of text that uniquely identifies the particular format/screen. The data must be preceded by these two pieces of information:

- The line number (i.e., row number) on the specified format/screen on which the text can be found
- The starting position (i.e., column number) on that line where the text begins.

The Term3270 Package uses the data line(s) in the user-supplied *Screen Identification* file to identify the name of the format/screen it is currently displaying in the GUI 3270 connection window.

9.2.1 Requirements for Format/Screen Names and Data Lines

Format/screen names must meet these specifications:

- The name must exist on a line by itself, starting in column 1.
- The name may contain embedded blanks, but trailing blanks will be ignored and leading blanks are not permitted. Names with embedded blanks must not be in quotes.
- The name may not exceed 80 characters.
- The name must be terminated by a new-line character
- The name must be followed by one or more data entry lines, each on its own line.

The data entry line or lines contain the text that uniquely identifies the preceding format/screen name. The data entry line(s) must meet these specifications:

- The data entry line(s) must contain
 - The line number of the point at which the text appears on the format/screen
 - The starting column of the text on that line.
- The data entry line(s) must start with a blank or a tab character, followed by the text line number, a blank or a tab character, the starting position, an optional blank or a tab character, a colon, and the text.
- The text may not exceed 132 characters.
- The text must be delimited by a colon on its left and either a new-line or an end-of-file character on its right.

NOTE — Lines that begin with a pound sign character in column 1 will be treated as comments. Lines that contain only the new-line character will be ignored.

9.2.2 Screen Identification File Formatting Commands

The MYNAH System provides the following commands to help you work with the Screen Identification File.

9.2.2.1 format-pattern

Syntax

```
format-pattern row col pattern
```

Description

The **format-pattern** formatting command searches the currently displayed format starting at the row and column specified for the pattern. Only one row is searched. If a match is made, **format-pattern** generates the **format** commands that will be entered into a script when using the MYNAH GUI.

format-pattern takes the following arguments:

- | | |
|----------------|---|
| row | The row on which the format name is expected to be displayed. |
| col | The column at which the search for the format name should begin. The search is performed starting from this column through the end of the row. |
| pattern | The pattern to be used to search for the format name. This pattern is a regular expression as defined for the regex(3X) program. (For more information, see the <i>UNIX System V Programmer Reference Manual</i> published by AT&T Bell Laboratories.) |

NOTE — The arguments are separated by the blank character.

Example

A user might be testing an application in which screen names containing capital letters, digits, and blanks appear with open and closed parentheses. The screen names may appear on either the first or the second line of the screen, somewhere after column 24.

The following **format-pattern** commands would be used to search for these format names:

```
format-pattern 1 25 "\\(([A-Z0-9 ]+)$0"
```

```
format-pattern 2 25 "\\(([A-Z0-9 ]+)$0"
```

Although the commands specify to search different lines, they have common arguments.

Table 9-1. Example format-pattern Regular Expressions

Argument	Description
[A-Z0-9]	The pattern enclosed in square brackets indicates that a match will occur for any capital letter, digit, or blank.
[A-Z0-9]+	The addition of a repetition character, +, to the defined pattern indicates there can be one or more occurrences of any capital letter, digit, or blank.
([A-Z0-9]+)	The opening and closing parentheses indicate that the pattern should be considered a group, as permitted by the program regex(3X) .
([A-Z0-9]+)\$0	The format-pattern command requires the addition of \$0, which indicates that the preceding group should be stored in the first variable that is hard-coded in a script. The MYNAH System requires one (and only one) variable. The variable is numbered 0, following the "C" programming convention that language arrays start at 0.
\(([A-Z0-9]+)\$0	The open parenthesis is added to the pattern to restrict the search to screen names preceded by an open parenthesis. Since parentheses are treated as special characters by regex(3X) , the open parenthesis is preceded by a back-slash.
"\(([A-Z0-9]+)\$0"	The completed expression must be enclosed in quotation marks.

If the MYNAH System is unable to determine the format/screen name, it will generate the following command in the user's script:

```
format <no-format-name-found>
```


9.2.2.2 format-name-mask

Syntax

```
format-name-mask character
```

Description

The **format-name-mask** formatting command mask any character, meaning that if a masking character is specified, its occurrence in the text strings of the *Screen Identification* file will be treated as a “*match any*” character during comparisons. The mask can be used when a format/screen contains text that cannot be uniquely repeatable, such as time or date stamps.

One character will be masked for each mask character present in the data entry’s text string of the *Screen Identification* file. When the MYNAH System compares the text of the data entry to that of the format/screen, the mask character—if present in the data entry’s text—will match any character, in the same position, in the text from the format/screen currently displayed.

NOTE — When creating the **format-name-mask** declaration, you must enclose the mask character in quotation marks.

Example

A date may appear on a line that you want to enter as a data line in a *Screen Identification* file. However, if you enter an explicit date in the data line, e.g., 06/06/96, a match will only be made on that date. To create a mask character, e.g., @, to match the changing date stamps, enter

```
format-name-mask "@"
```

at the beginning of a *Screen Identification* file. Then, in any data lines that would contain a date stamp, enter @@/@@/@@ in the place on the data line corresponding to the date stamp.

9.3 How the MYNAH System Processes the Screen Identification File

When a Term3270 connection is created, the MYNAH System loads and preprocesses the user-specified *Screen Identification* file specified for that Term3270 connection. (The *Screen Identification* file for each Term3270 connection is specified as an optional parameter for each connection configuration in the *xmyConfig* file.) The format/screen names and associated identifying data are then stored internally.

As format/screens are displayed (when you're using the MYNAH GUI), the MYNAH System first searches for strings of text on the format/screen that occur at row and column positions specified in the internally stored data. If the system finds that there is a string of text at a specified location, it then tries to match the string with text in the stored data.

- If it finds a match, the system uses the format/screen name associated with the matching text to generate a **format** statement and load the correct *Tag Name* file.
- If no match occurs, the system generates a **format** statement with the text *<no-format-name-found>* in your script.

NOTE — If more than one data entry line is specified for a format/screen name entry, each text string specified must match to the currently-displayed format/screen for the match to be successful.

If the system encounters syntax errors in the *Screen Identification* file or the file is inaccessible, the system reports the problems to you in the window in which you started the system or in the console window and stops processing the file.

An example of a user-supplied *Screen Identification* file appears in [Figure 9-1](#).

```
#adding format pattern for example screens
format-pattern 2 2 "[a-zA-Z0-9 ]*$0"

format-pattern 2 75 (MSG[0-9]+)$0

format-name-mask "@"

#####3#####3#####3
select
  2 2:** FACTS MENU **
loppmi
  4 2:** MORE APPEARANCES **
admgrp
  2 2:** ADMINISTRATIVE GROUP TABLE **
apnt
  2 2:** POINT TO POINT **
msg10
  2 76:msg10
msg10
  2 76:MSG10
```

Figure 9-1. Example Screen Identification File

The file contains two **format-pattern** command declarations.

- The first declaration searches for text on the second row and starting from the second column through the end of the row. This pattern will search for any text that matches these patterns
 - The string `[a-zA-Z0-9]` means that a match will be made for any letter, digit, or space.
 - The repetition character, `*`, means that there can be one or more letters, digits, or spaces in the text string.
 - The opening and closing parenthesis means that the pattern should be considered as a group.In other words, this declaration will try to match any string of letters, digits, and spaces on the second row starting at the second column.
- The second declaration also searches on the second row, but starts from row 75 and searches for any text that matches these patterns.
 - The string `[0-9]` means that a match will be made for any digit.
 - The repetition character, `+`, means that there can be one or more digits in the text string.

— The string **MSG** means that this explicit string must appear on the format/screen.

In other words, this declaration will try to match a starting with **MSG** followed by any digits on the second row and starting at the seventy-fifth column

As each format/screen is displayed, the system references data from this file. If the system finds the string **** FACTS MENU **** on row 2 starting at column 2, it will load the *select Tag Name* file. If it does not find, the system will next determine if the string **** MORE APPEARANCES **** occurs on the format/screen and if it satisfies the row/column specifications for a **format-pattern** command.

9.4 Enabling the Use of the Screen Identification File

The *Screen Identification* file is specified as the argument to the **ScreenIdentificationFile** parameter of a **Term3270** entry in the *xmyConfig* file. The specified *Screen Identification* file is automatically loaded when the user establishes a 3270 connection.

10. Output Processing

10.1 Process Output - Errors and Tracing

Process Error messages are routed to the central error handler. Administrative messages are also routed to the central error handler at process startup time and shutdown time. The startup messages include such information as which configuration files were loaded.

When starting the **vxLogFile** process, the location of this file is specified. The recommended location is *\$XMYHOME/syslog/adminlog*.

See [Section 5.14](#) for information on how to view this file.

10.2 Message-Based Tracing

Message-based tracing allows trace level to be dynamically changed on a per-script execution request basis. Tracing remains on (and trace level and file are passed to child script execution requests) until the script engine replies to the execution request. At that time tracing is automatically turned off.

This is used only for background script execution requests and only when there is a problem. It can be set in the GUI's Run Script dialog box (see Section 10.7 of the *MYNAH System Users Guide*) or by using the **xmyCmd submit** command (see Section 16.2.5 of the *MYNAH System Users Guide*).

10.3 Activity Logging

Only the SD, BD, and TD processes perform Activity Logging. Activity Logging consists of logging the following information:

- All requests the process receives or sends
- All script and SE state changes
- Copies of all (administrative and error) messages sent to the Telexel error logger.

Activity Logging information is stored in the respective directory where each process runs, i.e., in the *\$XMYHOME/run/bd*, *\$XMYHOME/run/sd*, *\$XMYHOME/run/td* directories.

As new Activity Log files are created, the existing files are given a version suffix. The current file will have no suffix, the one before that will have a suffix of 0, the one before that a suffix of 1, etc.

SD Activity Log filenames take the form *log.sd.<SD_name>.n* where

- *<SD_Name>* is the name of the SD being logged
- *n* is the version suffix.

BD and TD Activity Log filenames take the form *log.<process>.<hostname>.n* where

- *<process>* is either *bd* or *td*
- *<hostname>* the name of the host being logged
- *n* is the version suffix. (This is only used for backup files.)

Activity Logging is separate from Administrative Logging. Activity logging can be turned on or off, on a per-SD basis, by setting the *xmyConfig* file **ActivityLogging** parameter to either ON or OFF. This is determined at process start-up time and cannot be dynamically changed. By default, activity logging is on. See [Section 3.3.3.3](#) for more information.

Some MYNAH installations may not want to do any activity logging, primarily for efficiency reasons.

NOTE — Whenever the SD writes an error to the Telexel Logger, it will also write the error in the Activity Log, and if Message Based Tracing is on, it will write the error to the message based trace file as well.

NOTE — The Extension Packages that require additional processes may perform activity logging as well.

10.4 Script Output

Whenever a script is executed, the SE may produce files in an output directory. Whether or not files are produced and how much is produced is controlled by the **OutputLevel** entry. This is first set in the *xmyConfig* file, but it can be changed any time from within a script using the **xmyVar(OutputLevel)** array variable (Section 7.13 of the *MYNAH System Scripting Guide*).

There are several types of files: *output*, *SUTimage*, *compares*, *stdout*, and *stderr*.

The following subsection describes the location of named script output only.

10.4.1 Location of the Output Files

By default, the MYNAH System will generate script output in the directory containing the script. The script output is contained in a subdirectory with the same name as the script plus the suffix *.out* and a timestamp.

For example, if the current date and time is September 1, 1999, 4:28:24 PM, the output directory for the current run of the script */home/scripts/script1.tcl* is */home/scripts/script1.tcl.out.19990901.162824*.

NOTE — If the **OutputBackups** parameter is set to 0 (i.e., no backups are retained), the output directory name will be the name of the script with the *prefix out*; the timestamp will not be included. Using the previous example, the output directory will be *out.script1.tcl*.

NOTE — Remember, you must open the permissions of the directory containing the script or the MYNAH System will not be able to generate the output subdirectory.

10.4.1.1 Output Directory Symbolic Link

The MYNAH System creates a symbolic link that points to the output directory created by the latest script execution. The name of the symbolic link will be *<scriptname>.out*. Using this link you can analyze the results using your home grown scripts without needing to change the output directory name for each run.

NOTE — Some UNIX commands do not behave the same way on symbolic links as they do on actual directories, such as in following example:

```
broccoli> ls -l xyz.tcl.out
lrwxrwxrwx  1 madmin mynah    29 Feb 22 13:06 xyz.tcl.out
->./xyz.tcl.out.19980222.130535/
broccoli> ls -l xyz.tcl.out.19980222.130535
total 8
-rw-r--r--  1 madmin mynah    2020 Feb 22 13:06 output
```

10.4.1.2 Determining How Many Output Directories to Retain

How many output directories to retain is configured by the MYNAH Administrator in the *xmyConfig* file.

- If the MYNAH System has been configured so that zero backups are kept, any existing directories for the current script are removed before a new one is created.
- If a finite number of backups are being kept, then when the finite number of backups is reached the oldest output directory is removed after the current run completes.
- If unlimited backups has been requested, no removal takes place.

If an SE finds a directory with the same name and timestamp as the one it is trying to create, it sleeps a second, regenerates the directory name, and tries again, so as not to overwrite another SE's active output directory.

Subsequent runs increment the number of output directories unless the script engine is configured to keep only one copy of each script's output.

10.4.1.3 Other Possible Locations

The default location of the script output directory can be overridden by the MYNAH Administrator setting either the **OutputRoot** or **OutputPath** entry in the *xmyConfig* file in the **General SE** section or a **SEGroup** section. **OutputPath** is set to the full path to a directory in which script output should be created. **OutputRoot** is set to the root directory in which a path to the directory containing script output should be created. For example, if **OutputRoot** is set to */output*, and an execution request for the script */home/scripts/script1.tcl* is received, the script output directory will be */output/home/scripts/script1.tcl.out.<timestamp>*.

10.4.2 Date-Specific Testing and Output Directories

When machine dates and times are manipulated for date-specific testing (such as for Y2K testing), an output directory will be removed if that directory's creation date and time are earlier than existing output directories, depending on the *xmyConfig* file setting for number of output directories to retain.

For example, presume that the system is configured to retain two (2) output directories. If the script *script01.tcl* was submitted for execution on 12/31/1999 at 16:43:04 and again at 16:53:41, the system would generate the output directories

- *script01.tcl.out.19991231.164304*
- *script01.tcl.out.19991231.165341*

Several days later, the machine's date was reset to 12/31/1999, and the same script was resubmitted for execution at 11:30:27, thus causing a third output directory to be created, e.g.,

- *script01.tcl.out.19991231.164304*
- *script01.tcl.out.19991231.165341*
- *script01.tcl.out.19991231.113027*

Since the system is configured to retain only two output directories, the MYNAH System removes the "earliest" by date (*113027*). However, this is the most recently created output directory and the one you should be currently working with.

To minimize the impact of changing the system date, you can use either of the following work-arounds:

- After resetting the machine date, you can take care to manually remove the existing output directories so that there are no existing output directories prior to submitting tests.
- If there is sufficient disk space, simply increase in the *xmyConfig* file the number of output directories to be preserved before submitting tests.

10.4.3 Content of the Output Directory

A script output directory for a single script execution may consist of several files.

NOTE — These files are created during execution, but if a file is empty after the execution has completed, the file is deleted. Thus, you may temporarily see these empty files in your directory.

The <i>CmpMstr</i> file	This file contains the actual values encountered during a comparison, provided you use the Compare Master feature. See Section 5 of the <i>MYNAH System Scripting Guide</i> for information on using the Compare Master.
<i>SUTimage.<conn></i> files	These files contain SUT input and output, where <i><conn></i> is the connection handle, e.g., <i>SUTimage .xmyTermAsync_1</i> . This output can be in the form of screens or messages. There may be one or more files, and these files are created only if a SUT is accessed. If <i>SUTimage</i> events are not written to the <i>output</i> file, then output to the <i>SUTimage</i> files is also not logged.
The <i>compares</i> file	This file contains the differenced output for any compare commands, that is, if your script compares one event against one or more events, the expected and received output are written to this file. If compare events are not being written to the <i>output</i> file, then output to the <i>compares</i> file is also not logged.
The <i>output</i> file	This file contains events as defined in Section 10.4.4 . It is the highest level view of a script execution.
The <i>result</i> file	This file contains the script output presented in a user-readable format, provided you use the user-readable option when submitting the script. For more information, see the xmyCmd mergeOutput description in Section 16.2.12 of the <i>MYNAH System Users Guide</i> .
The <i>stdout</i> and <i>stderr</i> files	These files contain anything that is coded to write to the standard output or error.

For example, after executing (with the user-readable option) the script *abc.tcl*, you may see the following files and directory (assuming your output level included *sutimage*, *compare*, *stderr*, and *stdout* and your script contained compares using the Compare Master feature, access to a 3270 SUT, and commands to write to *stderr* and *stdout*):

- In the script directory you would see

```
abc.tcl
abc.tcl.out
abc.tcl.out.19960612.082440/
```

- Then in the output directory *abc.tcl.out* you would find

```
SUTimage..xmy3270_1
CmpMstr
compare
output
result
stderr
stdout
```

10.4.4 Output file

The *output* file contains script output information formatted using well-defined, colon separated prefixes. The categories of script output are

- child script (childscr) events
- compare events
- error events
- language trace events
- script events
- summary events
- sutimage events
- suttiming events
- test object (testobj) events
- user events

The first five fields of every event line contain the same type of information:

```
<date>:<time>:<category>:<pkg>:<type>
```

where

- *<date>* and *<time>* are date and timestamp with format *YYYYMMDD:HHMMSS*
- *<category>* is one of the listed categories (e.g. *sutimage*)
- *<pkg>* is the MYNAH package producing the message (e.g. *TermAsync*).
- *<type>* is the type of message, which differs for each category of output.

The following subsections describe the format of each category of output.

It is suggested that

- You should always choose to produce, at minimum, *script* events.
- You should never choose *lang* events for an entire script. These events are useful when attempting to debug a script but you should turn them on by using **xmyVar(OutputLevel)** just before the lines of code that are causing the problem.

10.4.4.1 Child Script Events

Child script (childscr) events are logged when connecting to a remote script engine and when any requests or replies related to child script execution occur. Note that the **sendWait** command logs two messages, one of type **send** and one of type **receive**. Each childscr event line takes the form

```
<date>:<time>:childscr:<pkg>:<type>:<conn>:<msgId>:<script name>:<status>:<exit string>
```

where

- *<pkg>* is *general*
- *<type>* is one of *send*, *receive*, *pause*, *resume*, or *cancel*
- *<conn>* is the name of the SE Group to receive the request
- *<msgId>* (*pause*, *resume*, *cancel*, *destroy*, *wait* only) is the message object id
- *<script name>* is the name of the script to be executed remotely
- *<exit string>* (*receive* only) is the Tcl result of script execution. (This string will be truncated if it exceeds the allowable length for output file lines)
- *<status>* is the result of the operation.

10.4.4.2 Compare Events

Compare events detail the differenced output for any **compare** commands. Each compare event line takes the form

```
<date>:<time>:compare:<pkg>:<type>:<label>:<result>:<index>
```

where

- *<type>* is one of *data*, *fcif*, *diff*, *screenRegion*, or *string*
- *<label>* is non-null if the compare specified a label
- *<result>* indicates whether the test passed or failed by specifying one of *good*, *failed*, or *warning*
- *<index>* is the number of characters into the *compare* file where the compare begins.

Example

```
19951201:090000:compare:Term3270:screenRegion::good:226
```

compare commands log the compare event to the *output* file and also writes expected and actual data, if possible, to the *compares* file. (If an expression is being evaluated instead of data being compared, the expression is written to the *compares* file.)

10.4.4.3 Exception (error) Events

The SE logs exception events when an exception occurs in the Tcl interpreter. Exception events contain the value of *errorCode* and *errorInfo* as set by the interpreter and take the form

```
<date>:<time>:error:<pkg>::<message>
```

where

- *<pkg>* is *errorCode* or *errorInfo*
- *<message>* is the error message itself.

10.4.4.4 Language Events

Language events record the lines of your scripts, with the format

```
<date>:<time>:lang:<pkg>:<type>:<command>
```

where

- *<type>* is one of *var*, *command*, *info*
- *<command>* is the Tcl command itself. Variables with unprintable characters or lines longer than 512 characters are truncated.

Examples

```
19950601:120000:lang:tcl:command:set x $y
19950601:120000:lang:tcl:var:x set to 0
19950601:120000:lang:tcl:command:$conn1 send Enter
19950601:120000:lang:3270:info:.xmy3270conn.1 sent 1 byte
19950601:120000:lang:3270:var: attr. at row 1 col 1 set to 0
```

Domain packages should put the connection handle in all messages related to connections, as on the fourth line in the example above.

NOTE — No language events are recorded for the commands in MYNAH procedures, even if *language* events are turned on in the Output Level.

10.4.4.5 Script Events

Script events record high level script activity, e.g., when the script began execution. Script events have the format

```
<date>:<time>:script:<pkg>:<type>:<group>:<exit status>:<exit string>
```

where

- *<pkg>* is null
- *<type>* is one of *start*, *stop*, *cancel*, *pause*, *resume*, or *abort*
- *<group>* is the SE Group to which this SE (the one running the script) belongs
- *<exit status>* is the return code from the Tcl interpreter for types *stop*, *cancel*, and *abort* only
- *<exit string>* is the possibly truncated Tcl result of script execution for types *stop*, *cancel*, and *abort* only.

Example

```
19960613:161728:script::stop:SeGpl:TCL_OK:
```

10.4.4.6 Summary Events

Summary events are written to the output file at script completion time. They include the system variables, the results that are sent to the database, and the symbol table. They have the format

```
<date>:<time>:summary:general:var:xmyVar(<name>) <value>
<date>:<time>:summary:general:results:<results list split into 80 char lines>
<date>:<time>:summary:general:symtbl:<symbol table split into 80 char lines>
```

Examples

```
19960613:161728:summary:general:var:xmyVar(Channel) = xmySE0007SD1
19960613:161728:summary:general:var:xmyVar(DatabaseMode) = 1
19960613:161728:summary:general:var:xmyVar(EngineMode) = stateLess
19960613:161728:summary:general:var:xmyVar(EngineType) = background
19960613:161728:summary:general:var:xmyVar(ExitHandler) =
19960613:161728:summary:general:var:xmyVar(FailedCompares) = 0
19960613:161728:summary:general:var:xmyVar(GoodCompares) = 1
19960613:161728:summary:general:var:xmyVar(LibraryPath) = /home/mynah/lib
19960613:161728:summary:general:var:xmyVar(OutputLevel) = error
19960613:161728:summary:general:var:xmyVar(RuntimeId) =
19960613:161728:summary:general:var:xmyVar(SEGroup) = SeGp1
19960613:161728:summary:general:var:xmyVar(SubmittedBy) = ksb
19960613:161728:summary:general:results: {result {}}
19960613:161728:summary:general:results: {goodCompares 1}
19960613:161728:summary:general:results: {failedCompares 0}
19960613:161728:summary:general:results: {warningCompares 0}
19960613:161728:summary:general:symtbl:<symbol table empty>
```


10.4.4.7 Sutimage Events

If sutimage events are being produced, then each time a screen or message is sent or received a sutimage event is produced having the format

```
<date>:<time>:sutimage:<pkg>:<type>:<conn>:<opt>:<index>:<length>
```

where

- *<pkg>* is the name of the MYNAH package producing the sutimage
- *<type>* is one of *snd*, *rcv*, or *state*. (*state* is used to produce a record of the “current” screen. It is used by the TermAsync package only.)
- *<conn>* is the handle of the connection
- *<opt>* is an optional field used by the domain to indicate the cause of the sutimage
- *<index>* is a pointer to the start location of the sutimage in the *SUTimage* file
- *<length>* is the length (in domain-specific units) of the image.

Example

```
19960613:161727:sutimage:TermAsync:snd:.xmyTermAsync_37::2207:3
```

Each package may add additional fields as needed.

10.4.4.8 SUT Timing (suttiming) Events

Timing events are logged each time a message is received from the SUT. They have the format

```
<date>:<time>:suttiming:<pkg>::<conn>:<command>:<send time>:<seconds>
```

where

- *<conn>* is the connection handle
- *<command>* is the Tcl command that sent data to the SUT
- *<send time>* is the time of the last send in readable format.
- *<seconds>* contains the number of seconds elapsed since the last send.

Example

```
19960613:161726:suttiming:TermAsync::.xmyTermAsync_37::19960613 161726:0.000000
```

10.4.4.9 Test Object Events

Test object (testobj) event lines are produced by the **xmyBegin** (*MYNAH System Scripting Guide*, Section 6.2.2) and **xmyEnd** (*MYNAH System Scripting Guide*, Section 6.2.6) commands. Test object event lines take one of these two forms

```
<date>:<time>:testobj:<pkg>:begin:<id>  
<date>:<time>:testobj:<pkg>:end:<id>:<results>
```

where

- *<pkg>* is null
- *<id>* is the test object id or test block label
- *<results>* contains the top-level results of execution (**xmyVar(GoodCompares)**, **xmyVar(FailedCompares)**, etc.).

10.4.4.10 User Events

The MYNAH System provides a method for script writer to generate their own events to the output file using the **xmyPrint** command. (See Section 6.2.11 of the *MYNAH System Scripting Guide*.)

```
<date>:<time>:user:<pkg>:<type>:<text>
```

where *<pkg>*, *<type>* and *<text>* are provided by you through the MYNAH **xmyPrint** command.

Examples

```
19960611:145636:user:::MAIN  
19960611:145636:user:::LEAVING PROC inits (/users/kjb/scripts/parent/par002.tcl)
```

10.4.5 The result File

The *result* file contains the output from the script output in a user-readable format.

The *result* file is generated when you specify the **-u** option when you submit a script for execution, such as

```
xmyCmd submit -v -E -c -G -u myscript.tcl
```

NOTE — If you do not specify the **-u** option when submitting a script, you can still view the output in this format by using **xmyCmd**'s **mergeOutput** sub-command, for example

```
xmyCmd mergeOutput -u -s SUTimage..xmyTerm3270_1
```

For information on the **mergeOutput** sub-command, see Section 16.2.12 of the *MYNAH System Users Guide*.

The *result* file contains two sections: a **Run** section and **Summary** section.

10.4.5.1 Run Section

The **Run** section contains all of the statements from the *output* file with the contents of other files in the *output* directory merged in at appropriate places. The content depends on the specified Output Level.

The **Run** section displays text using the following conventions:

- Script commands start with "--", for example

```
--xmyTerm3270 connect
```

- Screen output (the screens that are sent and received) contains line numbers, for example, the following text appears on line 20 on a 3270 connection window:

```
20 INPUT APPLICATION NAME AND PRESS ENTER
```

- Return values and other output events appear left-justified, for example:

```
compare:General:data:First Compare:good:0  
COMPARE HEADER - xmyCompare (index:0)  
expr: $counter == 10  
result: 1 (good)  
COMPARE FOOTER - xmyCompare
```

10.4.5.2 Summary Section

The *Summary* section displays the Summary Events from the *output* file (see [Section 10.4.4.6](#)), in a user readable format.

10.4.5.3 result File Example

The following example illustrates a *result* file where the Output Level was set to *all* when the script was submitted.

NOTE — The Sutmage portion contains only excerpts from the **send** and **receive** screens, and the *Summary* section contains only an excerpt from the *output* file's Summary events.

```
***** RUN STARTING 12/8 1998 17:35:42 *****
script::start:SeGp2, /export/home/pt06/scripts/sanity/temp.tcl
--xmyLoadPkg Term3270
--set counter 10
--xmyPrint -text "start 3270"
start 3270
--xmyTerm3270 connect
suttiming:Term3270::.xmyTerm3270_2:sent ::0.000000
--Screen received (12/8 1998 17:35:44):.xmyTerm3270_2::0:0
01
02
03 INPUT APPLICATION NAME AND PRESS ENTER
04
05
06
07
--set conn10 [xmyTerm3270 connect ]
--Screen sent:.xmyTerm3270_4::6018:0
01
02
03
04
05 INPUT APPLICATION NAME AND PRESS ENTER
06 st8
--$conn10 disconnect
--xmyPrint -text "completed 3270"
completed 3270
--xmyCompare -label "First Compare" -expr {$counter == 10}
compare:General:data:First Compare:good:0
COMPARE HEADER - xmyCompare (index:0)
expr: $counter == 10
result: 1 (good)
COMPARE FOOTER - xmyCompare
--xmyExit "Got to the end"
script::stop:SeGp2:TCL_OK:Got to the end
***** RUN ENDED 12/8 1998 17:35:44*****
```

```
***** SUMMARY *****  
xmyVar(Channel) = xmySE0002SD2  
xmyVar(DatabaseMode) = 1  
xmyVar(EngineMode) = connState  
xmyVar(EngineType) = background  
xmyVar(ExitHandler) =  
xmyVar(FailedCompares) = 0  
xmyVar(GoodCompares) = 1  
xmyVar(LibraryPath) = /opt/SUNWmynpt1/mynah/lib  
xmyVar(MaxFails) = 2147483647  
xmyVar(MaxFailsHandler) =  
xmyVar(OutputLevel) = *  
xmyVar(RuntimeId) =  
xmyVar(SEGroup) = SeGp2
```

10.4.6 Setting Output Level

Since all data written to the execution record is prefixed, users can choose the type of data to be written by specifying the prefixes they want to see. This symbolic representation of the output level can be configured on an installation-wide basis, on an SE group-wide basis, or for an individual script. If no level is specified, the default is to print error events and user events only. The default can be overridden with an entry in the **Engine** section of the *xmyConfig* file. The format is a list of positional prefixes for the desired messages. For example, to see error, language, and user trace statements and user events, the following setting would be placed in the configuration file:

```
Engine 3270_conn
      OutputLevel = { error lang user }
```

Error output should always be enabled, or errors will not be seen in the output file, only in the system-wide Telexel error log. Since the date and time stamp always appear, those prefixes do not appear in the output level setting.

The syntax of **OutputLevel** is a Tcl list with zero or more elements representing output file prefixes. Setting **OutputLevel** to { } (the empty list) will cause the output file to be empty.

You can use the star (*) wildcard to write all output to the output directory, as in

```
set xmyVar(OutputLevel) {*}
```

10.4.6.1 Recommended Standalone Engine Output Levels

Standalone Engines require an open Output Level because a child script generated by **xmytclsh** will inherit the OutputLevel from the parent script. Therefore, the recommended OutputLevel is:

```
OutputLevel = ( error, childscr, compare, script, summary,
               sutimage, testobj, user )
```

10.4.6.2 Recommended Embedded Engine Output Levels

Embedded Engines will not produce a summary even if this parameter is specified. In addition, the sutimage level produces a lot of output, so this parameter should be left out of the OutputLevel. Therefore, the recommended Output Level is:

```
OutputLevel = ( error, childscr, compare, script,
               testobj, user )
```

10.4.6.3 Recommended Background Engine Output Levels

Background Engines should produce all OutputLevel except lang and suttiming due to the amount of output that is generated. Therefore, the recommended OutputLevel is:

```
OutputLevel = ( error, childscr, compare, script, summary,  
               sutimage, testobj, user )
```

NOTE — The open Output Level (*) is not recommended for Background Engines.

11. Creating DCE Executables

DCE scripting requires an external application-specific emulated client or server executable, which you must build.

To help you build the the executables, we've included the **xmyDceBuild** program, which takes a **makefile** and invokes **make** to build the executables.

NOTE — Both **gnumake** and the Solaris **make** will work.

In addition, [Appendix C](#) describes other methods of building the executables, including manually building the executables and directly editing the **makefile** and invoking **make**.

The following packages must be installed before you can build the executables:

- DCE development environment, Transarc Version 1.1
- RogueWave's **Tools.h++** header files and libraries (Version 7.0.6)
- The appropriate C++ compiler, as listed in [Table 11-1](#).

Table 11-1. DCE C++ Compilers

Product	Version	Vendor
SPARCworks C++Compiler	4.2	Sun Microsystems
aC++	A.03.10	HP

NOTE — These are only required by the administrative step of building the emulated client and server executables. They are not required when running DCE Tcl scripts.

Prior to using this tool, you must create the template Makefile, *xmyDceMakefile.template*, and place it in the *\$XMYHOME/data/dce* directory. An example template file, *xmyDceMakefile.template.eg*, is placed in the *\$XMYDIR/examples/dce* directory during installation of the MYNAH System. To create the template Makefile

1. Copy the *xmyDceMakefile.template.eg* file from *\$XMYDIR/examples/dce* to *\$XMYHOME/data/dce*, and rename the copy *xmyDceMakefile.template*.
2. Modify *xmyDceMakefile.template* according to the local environment. Specifically, the RW variables must be defined and the CC and CXX variables should be checked to make sure they're correct.

NOTE — If you built DCE programs for a previous release, you must rebuild them with each new MYNAH release.

The syntax, complete description, and examples for using the **xmyDceBuild** program are as follows:

Syntax

```
xmyDceBuild ?-v? ?-template templatefile? ?-client? \  
    ?-server? ?-build buildargs? ?-acf acf? \  
    ?-other other-idls? interface
```

Description

The **xmyDceBuild** program creates the emulated client and emulated server executables needed for the DCE Package.

xmyDceBuild creates a copy of the template *Makefile*, substituting variables based on the interface, ACF file, and other IDL files. **xmyDceBuild** then invokes **make** on the makefile to build the client and/or the server executable. **xmyDceBuild** takes the following options:

-v	verbose
-template <i>templatefile</i>	Location of Makefile template (default is <i>\$XMYHOME/data/dce/xmyDceMakefile.template</i> or <i>\$XMYDIR/data/dce/xmyDceMakefile.template</i>)
-client	Build just the client (default is both)
-server	Build just the server (default is both)
-build <i>buildargs</i>	Build this instead of client or server. <i>buildargs</i> can be any valid option(s) to make. By default <i>buildargs</i> is “client server”, which tells make to build the client and the server. If you say “-client”, then <i>buildargs</i> is set to just “client”; likewise, if you say “-server”, <i>buildargs</i> is set to just “server”.
-acf <i>acf</i>	Specifies the ACF file used to build the executables.
-other <i>other-idls</i>	List of other IDL files (with .idl extension)
<i>interface</i>	IDL file (without .idl extension)

The **-acf** option is not required. If there is a file with the same basename as the interface IDL with the *.acf* extension, that will be used.

Example

```
xmyDceBuild foo
```

Example

```
xmyDceBuild -acf foo.acf -other "a.idl b.idl" foo
```

Appendix A: Administrative Commands

This Appendix contains detailed descriptions of all of the CLUI and Tcl administrative commands.

A.1 Administrative CLUI Commands

The CLUI provides access to a subset of MYNAH functionality. These functions are

- Letting you perform operability tasks (stat, shutdown, and obtain status of other MYNAH processes).
- Providing access to the Telexel error log.
- Letting the user send requests to the SD to execute scripts in the background.
- Letting the user send requests to the SD to retrieve status or take action on a script or group of scripts, and display the results.
- Letting the user send requests to the SD to change user-tunable parameters in the SD (such as the user's maximum concurrency level).
- Letting the MYNAH Administrator send requests to the SD to change administrator-tunable parameters in the SD (such as the overall system concurrency level, or the number of SEs in an SE group).
- Letting the user display information about the current MYNAH configuration.
- Providing the user an interactive Tcl shell with MYNAH extensions.
- Letting the user create reports from information in the database.
- Letting the user create or update database entities, such as User Enum Values, and user entries.
- Giving the user a facility to encrypt a **des** key.
- Letting the user compare files.

This section describes the Administrative Command Line User Interface (CLUI) commands. Administrative CLUI commands are only available to the MYNAH user who has administrative privileges, e.g., *admin*. All other CLUI commands are available to all MYNAH users.

A.1.1 CLUI Command Help Messages

All CLUI commands and subcommands have an **-h** option that displays a brief help message listing the usage and syntax of the subcommand. For example

```
xmyOM query -h
query: display info about the MYNAH configuration
Usage: xmyOM query [-h] [-p<process_name> | -o<oa_name> | -s ]
       -h                : display this message
       -p<process_name> : process name as in config file
       -o<oa_name>      : Operability Agent name as in config file
       -s                : List all systems (OAs) defined in xmyConfigOP
```

In addition, each main command has an **-h** option to tell you what subcommands exist for the command, for example

```
xmyOM -h
Usage:
       xmyOM command_name command_args
The command_args depend on which command_name you use.
To obtain usage information on an individual command_name,
simply type:
       xmyOM command_name -h
Here is a list of all the valid xmyOM command_names and a brief
description of what they do:
start      - start a platform/application process
stop       - stop a platform/application process
status     - display status of platform/application process
query      - display MYNAH configuration
readconfig - request OA to re-read the configuration file
autostop   - request OA to bring down all autostart processes
autostart  - request OA to start up all autostart processes
recycle    - request OA to re-cycle (bring down then up)autostart proc's
shutdown   - stop OA (brings down all autostart processes too)
```

A.1.2 xmyCmd

Usage of the **xmyCmd** subcommands falls into two categories:

1. Administrative subcommands

These **xmyCmd** SD subcommands let a user with administrative privileges set the following SD parameters:

- The overall system concurrency limit
- The default maximum concurrency limit for new users
- The default queuing priority for new users
- Any user's maximum concurrency limit
- Any user's queuing priority
- Debug level (used for development and debugging purposes)
- Increase and decrease the number of SEs in an SE Group as the SD is running.

In addition, if you have administrative privileges, you can also assign administrative privileges to another user.

This section contains information on these **xmyCmd** subcommands.

All administrative **xmyCmd** subcommands (except **addEnumValue**) accept the following options:

- v** Generates verbose responses.
- h** Displays a brief help message.

All administrative **xmyCmd** subcommands that interact with an SD accept the following options:

- d *sd_name*** Specifies the SD name that is the subject of the command. If this option is not used, the value of the XMYSD variable is used. If this variable is not set, then the default SD name defined in the *xmyConfig* file is used.
- t *time-out*** Specifies the time-out interval. The CLUI waits for this interval to get the responses from SD. If a value of 0 is used, the CLUI will wait indefinitely until all the responses are received. By default, the time-out is 30 seconds.

2. User subcommands.

These **xmyCmd** subcommands include the ability to submit scripts and receive information about scripts.

These subcommands are explained in the *MYNAH System Users Guide*.

A.1.2.1 addEnumValue

Syntax

```
xmyCmd addEnumValue -t type -s string -l string \
-d true/false ?-q?
```

Description

The **addEnumValue** sub- command is used to add User Enum values to the MYNAH database. You can not add new Objects or Fields. When adding new values, the new values are added to a specific Object-Field pair from [Table 5-1](#). For example, you could add new values for the Object **Issue** and the Field **Status**, which would be specified for **xmyCmd addEnumValue** as *issueStatus*.

xmyCmd addEnumValue requires the following parameters:

- t *type*** Specifies the User Enum type for which you want to add a value, where *type* is an Object-Field pair shown in [Table 5-1](#), for example *dataType*.
- s *string*** Specifies a short description of the newly introduced value. This field can be up to 4 characters.
- l *string*** Specifies a long description of the newly introduced value. This field can be up to 15 characters.
- d *true/false*** Specifies if this is the default value for this Enum Value Object/Field pair.

In addition, **xmyCmd addEnumValue** accepts the following optional parameter:

- q** Queries for a list of the existing User Enum values

Only a Mynah administrator can run this comand.

Once you have added a value to a list for a field, it will automatically be provided to the users of the GUI as a valid choice for that field.

If you are adding a value that is the default, **xmyCmd addEnumValue** makes sure there isn't another one already marked as default.

Example

This is an example of adding the User Enum value *Environment* for the Object *issue* and the Field *Type*.

```
xmyCmd addEnumValue -t issueType -s ENV -l Environment -d true
```

In this example you add the value **Phase** for the Object *SutInfo* and the field *Type*.

```
xmyCmd addEnumValue -t sutInfoLevel -s PHS -l Phase -d false
```

Exception

- User is not known to MYNAH database. It means you're not authorized to use MYNAH.
- User is not a MYNAH database administrator. It means that you do not have enough privileges to create a new Enum Value. A **root** user *is not* a MYNAH database administrator.

A.1.2.2 checkAllQueues

Syntax

```
xmyCmd checkAllQueues ?-h?
```

Description

The **checkAllQueues** subcommand goes to each host defined in this configuration of the MYNAH System (i.e., in the *xmyConfigOP* file), checks whether the queues on that host are full, and reports the results back to `stdout`.

checkAllQueues takes no command-line arguments other than the **-h** (help) option.

In order for **xmyCmd checkAllQueues** to work successfully, the MYNAH Administrator must assure that in the *xmyConfigOP* file, each host has, on its responsibilities list, a process called **xmyQueues**. Furthermore, the **xmyQueues** process's **Status** command in the *xmyConfigOP* file must be "**xmyCmd checkQueues**".

NOTE — If you want verbose output, the **Status** command could have the **-v** option in it, e.g.,

```
Status = "xmyCmd checkQueues -v"
```

and if you want to use a different threshold, the **Status** command could be used, e.g.,

```
Status = "xmyCmd checkQueues -limit 85"
```

See [Appendix A.1.2.3](#) for more information about the **checkQueues** subcommand.

A.1.2.3 checkQueues

Syntax

```
xmyCmd checkQueues ?-h? ?-thresh percent? ?-v?
```

Description

The **checkQueues** subcommand checks the queues on the local host to see if they are full or nearly full.

In addition to the basic administrative **xmyCmd** subcommand **-h** and **-v** options listed at the beginning of this section, **checkQueues** accepts the following command-line option:

-thresh *percent* Sets the threshold percentage, above which a warning will be produced. This refers to the percentage of the total queue space in use on the system.

By default, **checkQueues** uses a warning threshold of 90%, which means that it checks to see if 90% of the available queues in the system are in use, and if so it produces a warning message.

Default = 90

A.1.2.4 dfltusrconc

Syntax

```
xmyCmd dfltusrconc ?-d sd_name? ?-t time-out? ?-v? value
```

Description

The **dfltusrconc** subcommand can be used by the administrator to set the default concurrency limit for new users. (New users get their actual concurrency set to this value when they first become known to the SD.)

The concurrency limit is the total number of scripts that can be run at one time.

In addition to the basic administrative **xmyCmd** subcommand options listed at the beginning of this section, **dfltusrconc** requires the following command-line argument:

value Specifies the default concurrency limit value for new users.

A.1.2.5 dfltusrpri

Syntax

```
xmyCmd dfltusrpri ?-d sd_name? ?-t time-out? ?-v? value
```

Description

The **dfltusrpri** subcommand is used by the administrator to set the default queuing priority for new users. (New users get their queuing priority set to this value when they first become known to the SD.)

In addition to the basic administrative **xmyCmd** subcommand options listed at the beginning of this section, **dfltusrpri** requires the following command-line argument:

value Specifies the default queuing priority value for new users.
Recommended range = 0 to 999.

A.1.2.6 sedecr

Syntax

```
xmyCmd sedecr ?-d sd_name? ?-t time-out? ?-v? \  
-e se_group_name delta_value
```

Description

The **sedecr** subcommand decreases the number of SEs in a particular SE group.

In addition to the basic administrative **xmyCmd** subcommand options listed at the beginning of this section, **sedecr** requires the following options/arguments:

-e *se_group_name* Specifies the SE group for which the number of SEs will be decreased.

delta_value Decrease the number of SEs by this amount.

If all SEs in the target SE Group are busy when **xmyCmd sedecr** is executed, the SD marks ***delta_value*** SEs for deletion. The SEs are deleted when they finish executing the current scripts. No running scripts will be interrupted due to issuing **xmyCmd sedecr**.

A.1.2.7 seincr

Syntax

```
xmyCmd seincr ?-d sd_name? ?-t time-out? ?-v? \  
-e se_group_name delta_value
```

Description

The **seincr** subcommand increases the number of SEs in a particular SE group or of all SE Groups.

In addition to the basic administrative **xmyCmd** subcommand options listed at the beginning of this section, **seincr** requires the following options/arguments:

- e *se_group_name*** Specifies the SE group for which the number of SEs will be increased.
- delta_value*** Increase the number of SEs by this amount.

A.1.2.8 sestat

Syntax

```
xmyCmd sestat ?-d sd_name? ?-t time-out? ?-v? ?se_group?
```

Description

The **sestat** subcommand returns the status of a particular SE Group.

In addition to the basic administrative **xmyCmd** subcommand options listed at the beginning of this section, **sestat** accepts the following command-line argument:

se_group Specifies the SE group containing the SEs whose status you want to see. If you do not specify an SE Group, **xmyCmd sestat** will return the status for all SEs in controlled by the SD.

A.1.2.9 setadm

Syntax

```
xmyCmd setadm ?-d sd_name? ?-t time-out? ?-v? login_id
```

Description

The **setadm** subcommand gives administrative privileges to a user (*login_id*) for only the target SD; no changes are made to the MYNAH database. The next time the SD is started the user will not have administrative privileges. Only an administrator can execute this subcommand.

In addition to the basic administrative **xmyCmd** subcommand options listed at the beginning of this section, **setadm** requires the following argument:

login_id Login id of the user to be given administrative privileges.

A.1.2.10 sysconc

Syntax

```
xmyCmd sysconc ?-d sd_name? ?-t time-out? ?-v? value
```

Description

The **sysconc** subcommand is used by the administrator to set the overall system concurrency limit.

In addition to the basic administrative **xmyCmd** subcommand options listed at the beginning of this section, **sysconc** requires the following argument:

value Specifies the overall system concurrency limit.

A.1.2.11 unsetadm

Syntax

```
xmyCmd unsetadm ?-d sd_name? ?-t time-out? ?-v? login_id
```

Description

The **unsetadm** subcommand removes administrative privileges from a user, which can be yourself. Only an administrator can execute this subcommand.

NOTE — No change is made to the database. To permanently remove administrative privileges from a user you must do so using the Person Object for that user.

In addition to the basic administrative **xmyCmd** subcommand options listed at the beginning of this section, **unsetadm** requires the following argument:

login_id Login id of the user for whom administrative privileges will be removed.

A.1.2.12 `usrmaxconc`**Syntax**

```
xmyCmd usrmaxconc -u login_id ?-d sd_name? ?-t time-out? \  
?-v? value
```

Description

The **usrmaxconc** subcommand is used by the administrator to set *any* user's maximum concurrency.

In addition to the basic administrative **xmyCmd** subcommand options listed at the beginning of this section, **usrmaxconc** requires the following options/arguments:

- u *login_id*** Specifies the user whose maximum concurrency is to be set.
- value*** Specifies the maximum concurrency limit value.

A.1.2.13 `usrpriority`

Syntax

```
xmyCmd usrpriority -u login_id ?-d sd_name? \  
?-t time-out? ?-v? value
```

Description

The **usrpriority** subcommand can be used by the administrator to update the queuing priority of another user.

In addition to the basic administrative **xmyCmd** subcommand options listed at the beginning of this section, **usrpriority** requires the following options/arguments:

-u *login_id* Specifies the user whose queuing priority is to be set.
value Specifies the queuing priority for the specified user.

Queuing priority adheres to the following standards:

- The lower the number the higher the priority.
- 0 is the lowest queuing priority value and the highest priority.
- The upper limit of a queuing priority value is 999.

A.1.3 xmyOM

The **xmyOM** command subcommands can be used to control and administer the platform and application processes that constitute MYNAH. These subcommands are implemented as part of the Operability Manager (OM), which is discussed in Section 4.

A user, at any point in time, can invoke the **xmyOM** command from any machine in the network.

A.1.3.1 autostart

Syntax

```
xmyOM autostart host_name
```

Description

The **autostart** subcommand causes the OA on the specified host (*host_name*) to start up all autostart processes defined for that OA, i.e., all processes in that OA's responsibility list whose **Autostart** entry is set to *yes*.

Example

```
xmyOM autostart selene
```

A.1.3.2 autostop

Syntax

```
xmyOM autostop host_name
```

Description

The **autostop** subcommand causes the OA on the specified host (*host_name*) to shut down all the autostart processes defined for that OA. It shuts them down in the reverse order from which they were started up, however, the OA continues to run after shutting them down. This distinguishes the **autostop** subcommand from the **shutdown** subcommand in that with the **shutdown** subcommand, not only does the OA shut down all its autostart processes, but it also shuts itself down.

Example

```
xmyOM autostop selene
```

A.1.3.3 query

Syntax

```
xmyOM query ?-o oa_name | -p logical_process_name? | -s?
```

Description

The **query** subcommand displays configuration file entries (from the *xmyConfigOP* file) for OAs and the managed processes.

query takes the following options:

- o oa_name** Displays the processes the entered Operability Agent (OA) is responsible for. (See [Section 4.3.2](#) for information on OAs.)
- p logical_process_name** Displays information for the entered process as defined in the *xmyConfigOP* file.
- s** Displays a list of all OAs (i.e., all hosts) defined in the *xmyConfigOP* file.

If no option is used, then all OA entries in the configuration file are displayed.

Example

```
xmyOM query -o mimicr
```

The OA on mimicr is responsible for the following processes:

vxGateway

xmyBD

xmySD2

A.1.3.4 readconfig

Syntax

```
xmyOM readconfig
```

Description

The **readconfig** subcommand instructs the OAs on all hosts to read the configuration file. **readconfig** should be used only if some changes have been made to the *xmyConfigOP* file. The OM broadcasts **readconfig** to all the OAs in the MYNAH configuration.

Example

```
xmyOM readconfig  
OA(selene): config file read  
OA(mimir): config file read
```

A.1.3.5 recycle

Syntax

```
xmyOM recycle host_name
```

Description

The **recycle** subcommand shuts down and restarts all Autostart processes on the specified host (*host_name*). This is identical to an **xmyOM autostop *host_name*** followed by an **xmyOM autostart *host_name***.

Example

```
xmyOM recycle selene
```


A.1.3.6 shutdown

Syntax

```
xmyOM shutdown oa_name
```

Description

The **shutdown** subcommand gracefully terminates an OA. The OA first stops all autostart processes it is responsible for, terminating them in the reverse order from which they were brought up. Lastly, the OA shuts itself down.

If you want to bring only the OA and not the Autostart processes down, you must be logged into the OA's machine and then use the **xmyStopOA** command. This will only work if you are **root** or the person who started the OA, usually **madmin**.

oa_name is the name of the OA as it appears in the *xmyConfigOP* file.

A.1.3.7 start/stop/status

Syntax

```
xmyOM start ?-o oa_name? logical_process_name
xmyOM stop ?-o oa_name? logical_process_name
xmyOM status ?-o oa_name? logical_process_name
```

Description

These subcommands can be used to start, stop, or get the status of the application or platform processes. The application or platform processes on which these subcommands operate must be present in the responsibility list of at least one OA in the *xmyConfigOP* file.

logical_process_name is the name of the process as it appears in the responsibility list of the OA in the *xmyConfigOP* file.

If **-o oa_name** is not supplied, **xmyOM** verifies that the *logical_process_name* appears in exactly one OA's responsibility list and forwards the request to the OA. If **-o oa_name** is supplied, **xmyOM** makes sure that the *logical_process_name* appears in that OA's responsibility list and forwards the request to that OA.

Example

```
xmyOM status -o selene vxIpCDir
vxIpCMgr: vxErSrv00      on selene      .. selected. no action.
vxIpCMgr: vxErrorServer on selene      .. selected. no action.
vxIpCMgr: vxLogDestFile on selene      .. selected. no action.
vxIpCMgr: xmyBDselene   on selene      .. selected. no action.
vxIpCMgr: xmySDSD1      on selene      .. selected. no action.
vxIpCMgr: xmySE0000SD1  on selene      .. selected. no action.
vxIpCMgr: xmySE0001SD1  on selene      .. selected. no action.
vxIpCMgr: xmySE0002SD1  on selene      .. selected. no action.
vxIpCMgr: xmySE0003SD1  on selene      .. selected. no action.
vxIpCMgr: xmyTD000      on selene      .. selected. no action.
vxIpCMgr: xmyTDselene   on selene      .. selected. no action.
```

A.2 Administrative Tcl Commands

The administrative Tcl commands let you perform administrative tasks for a specific testing package (e.g., Async or 3270). These commands are in fact MYNAH extension methods that must be executed using a Tcl interpreter, such as **xmytclsh** or the Script Builder. In addition, you can include these commands in scripts, which can be executed using the **xmyCmd submit** command.

Currently, there is only one administrative Tcl command, **xmyMsgDel**.

A.2.1 xmyMsgDel

Syntax

```
xmyMsgDel -topcom topcom_name ?-all? \  
          ?-timeBef timestring? ?-timeAft timestring?  
  
xmyMsgDel -printcom printcom_name ?-all? \  
          ?-timeBef timestring? ?-timeAft timestring?  
  
xmyMsgDel -handler handler_name ?-subDir subDir? ?-all? \  
          ?-timeBef timestring? ?-timeAft timestring?
```

Return

Prints the statistics of deleted messages on stdout.

Description

The **xmyMsgDel** method is used to delete message files. The options indicate the criteria based on which the deletion is done.

This method can be used by MYNAH administrators to periodically clean up the Message Directory. It is advisable that this procedure be used by administrators only. Administrators will be able to run the procedure to delete all the message files or they may use it according to a centralized policy for retention of messages. The policy could also include steps to backup the message directory before doing the clean up. If all messages are deleted from the message directory, the message directory is also removed.

NOTE — Messages for App-to-app over TOPCOM will be deleted, but the directory will remain even if it is empty.

Deletion of messages by other users may result in loss of messages that are of particular interest to some other user(s). Thus, there is a risk of accidental deletion of messages.

The valid attributes are

-subDir *subDir* Indicates the subdirectory containing the messages to be deleted. This attribute is valid with the **-handler** form only. If this attribute is not present, message files from all subdirectories will be deleted.

For more information on the **-subDir** attribute, referer to the entry in the Message Response Directory section (**xmyMsgDir** class) of the MYNAH SystemScripting Guide.

-all Deletes all the message files. The **-all** attribute only deletes the message files from the Message Directories. It does not delete any other user created files. Based on the naming convention of the files, a determination is made if the file is a message file or a user defined file.

-timeBef *timestring* Specifies the time before which the message files should be deleted. *timestring* should be a date and/or tim string in standard Tcl format, for example

```
-timeBef "1 Jan 1997"
```

-timeAft *timestring* Specifies the time after which the message files should be deleted. *timestring* should be a date and/or time string in standard Tcl format, for example

```
-timeBef "22 Sep 1997 11:00:00"
```

If both **-timeBef** and **-timeAft** are specified, chronologically speaking, **-timeAft** should be before **-timeBef**. This combination is useful for specifying a window of time. All the message files having timestrings in the specified window will be deleted.

Example

In this example, all messages for the handler **app_1** after January 1, 1997 and before January 1, 2000 will be deleted for all subdirectories.

```
> xmyMsgDel -handler app_1 -timeAft "1 Jan 97" \  
           -timeBef "1 Jan 2000"
```

In the following example, all messages for the topcom handler **top_1** before September 22, 1997, 11:00 AM will be deleted.

```
> xmyMsgDel -topcom top_1 -timeBef "22 Sep 1997 11:00:00"
```

This time, all messages for the Printcom handler **print_1** will be deleted.

```
> xmyMsgDel -printcom print_1 -all
```

Exceptions

Handler not found in the config file

Invalid options specified

Appendix B: Example Installation Files

This appendix contains examples of start-up scripts and example profile and system files mentioned in [Section 2](#).

B.1 Example MYNAH System Files

B.1.1 Example MYNAH Installation Session

This is an example BAIST installation session of the MYNAH System software.

```
Telcordia Application Installation Setup Tool
- - - - -
                BAIST 2.1

COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
                All Rights Reserved.

PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.

                A UNIX Packaging and Installation Tool

                For further information on BAIST
                Contact: Raymond C. Gray
                        BAIST Project Manager
                        (732)699-7960
```

```
                Please Wait ...
WARNING: You are using LOCAL BCRDB
***** BAIST Installation Tool Release 2.1 *****
Initializing. Please wait.
```

```
BAIST 2.1 FLOW CONTROL
PRODUCT MENU
  Archived Products List
```

- 1) MYNAH 5.4
- 2) TELEXEL 7.3
- 3) TOPCOM 5.4.1

- I) Installed Products
- R) Registered Products
- E) Exit

```
      Enter your Selection: 1
Load MYNAH? [Y]: y
Extracting Application Profile

Getting the name of PRODUCT_HOME dir

Where should the MYNAH directory be created ?
Enter a full pathname starting with /, or q to Quit processing
: /opt/SUNWmyn
Getting the name of RELEASE_HOME dir
Making the PRODUCT_HOME dir
Making the RELEASE_HOME dir

Who should own the MYNAH application ?
Enter a valid login name :admin

Running PRELOAD script

Extracting the MYNAH application
Please wait. This may take a while ...
100226 of 100226 blocks loaded 100% COMPLETED

Running POSTLOAD script

Where should XMYHOME be installed? [/opt/SUNWmyn/MYNAH/releases/MYNAH_5.0]
/u/sol/XmyHome
is /u/sol/XmyHome correct yes/no:
yes
Remember to:

1) edit /u/sol/XmyHome/config/xmyProfile
   edit port number
   edit LSHOST machine name
   edit ORACLE server machine name (if using ORACLE)
   edit vxIpcDirectory machine name

   add . /u/sol/XmyHome/xmyProfile to user profiles

2) export your DISPLAY variable
   export DISPLAY=machine:0.0
```


Product MYNAH MYNAH_5.0 loaded successfully

BAIST 2.1 FLOW CONTROL
PRODUCT MENU
Archived Products List

- 1) MYNAH 5.4
- 2) TELEXEL 7.3
- 3) TOPCOM 5.4.1

- I) Installed Products
- R) Registered Products
- E) Exit

Enter your Selection:**E**

B.1.2 Example System Changes File

This file contains changes that should be added to the */etc/system* file.

Once the changes have been made, type

```
reboot -- -rt
```

to reboot the system. The system will be reconfigured with the changes to */etc/system* incorporated in the kernel. You must do this as **root** on each system running a MYNAH component or on the ORACLE server.

```
set maxusers=128
set pt_cnt=60
# Settings for Message Queue parameters
#   MSGMNI : # of message queue identifiers
#   MSGTQL : # of system message headers
#   MSGMAP : # of entries in message map
#   MSGSSZ : segment size of message
#   MSGMNB : maximum bytes on queue
#   MSGMAX : maximum message size
#   MSGSEG : # of message segments
set msgsys:msginfo_msgmni=800
set msgsys:msginfo_msgtql=2000
set msgsys:msginfo_msgmap=1600
set msgsys:msginfo_msgssz=64
set msgsys:msginfo_msgmnb=65535
set msgsys:msginfo_msgmax=65535
set msgsys:msginfo_msgseg=16384
# Settings for Shared Memory parameters
#   SHMSEG : segments per process
#   SHMMAX : maximum shared memory segment size
#   SHMMNI : # shared memory identifiers
set shmsys:shminfo_shmseg=20
set shmsys:shminfo_shmmax=234881024
set shmsys:shminfo_shmmni=300
# Settings for Semaphore parameters
#   SEMMNS : # of semaphores in system
#   SEMMNI : # of semaphores identifiers
#   SEMMNU : # of "undo" in system
#   SEMUME :
#   SEMMAP :
set semsys:seminfo_semmns=7500
set semsys:seminfo_semmni=300
set semsys:seminfo_semmnu=300
set semsys:seminfo_semume=20
set semsys:seminfo_semmap=300
#
set s_xxx:max_ccbs=16
set s_xxx:x29_default=1
```

B.1.3 Example MYNAH xmyProfile File

This is an example of the **ksh** *xmyProfile* files as it appears immediately following installation of the MYNAH software.

You must make the following changes to this file to reflect your environment:

- Update the directories may need to be updated
- Replace *<hostname>* with the actual machine names.

NOTE — For the *oracleservername* entry, which is the name of your system running Oracle, this may or may not be the same as the system machine on which you installed the MYNAH System software.

- Replace *<port>* with the actual Telexel port (e.g., 22100)
- Replace *XXXXmyn* with either *SUNWmyn* or *HPUXmyn*, depending on your operating system.

After you have made the necessary changes to this file it can be sourced into the *.profile* files for all MYNAH users.

```
# COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
# All Rights Reserved.
#
# PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia
# and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)xmyProfile.eg53.4
# created on 98/11/03 at 13:15:28
#
#
# @(#)sample profile for madmin 96/06/11 SOL
#
# This is an example of a MYNAH System profile that
# can be sourced into MYNAH user's profiles to give
# them access to the system.
#
#
# XMYDIR is the path to a directory which is symbolically linked to
# the current release of MYNAH, e.g.,
# ln -s /opt/XXXXmyn/MYNAH/releases/MYNAH_5.0 /opt/XXXXmyn/mynah
#
XMYDIR=/opt/XXXXmyn/mynah
#
# XMYHOME is the path to a directory which is the MYNAH run environment
```

```
#
XMYHOME=/opt/XXXXmyn/mynah

#
# create or append to LD_LIBRARY_PATH/SHLIB_PATH
#
LD_LIBRARY_PATH='env | grep LD_LIBRARY_PATH'

if test -n "${LD_LIBRARY_PATH}"
then
    LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${XMYDIR}/lib:/usr/openwin/lib
else
    LD_LIBRARY_PATH=${XMYDIR}/lib:/usr/openwin/lib
fi

SHLIB_PATH='env | grep SHLIB_PATH'

if test -n "${SHLIB_PATH}"
then
    SHLIB_PATH=${SHLIB_PATH}:${XMYDIR}/lib
else
    SHLIB_PATH=${XMYDIR}/lib
fi

#
# create or append to MANPATH
#
MANPATH='env | grep MANPATH'

if test -n "${MANPATH}"
then
    MANPATH=${MANPATH}:${XMYDIR}/man
else
    MANPATH=${XMYDIR}/man
fi

PATH=${PATH}:${XMYDIR}/bin
if [ "$(uname)" = "SunOS" ]
then
    PATH=${PATH}:/usr/openwin/bin
elif [ "$(uname)" = "HP-UX" ]
then
    PATH=${PATH}:/usr/bin/X11
else
    echo "xmyProfile Warning: unrecognized platform $(uname)"
    echo "Please make sure that command xterm is in the path"
fi

TCL_LIBRARY=${XMYDIR}/lib/tcl
TCLX_LIBRARY=${XMYDIR}/lib/tcl

export XMYDIR XMYHOME LD_LIBRARY_PATH SHLIB_PATH PATH TCL_LIBRARY TCLX_LIBRARY

#
# The host name that the MYNAH license server runs on
#
```

```
LSHOST=<hostname>
export LSHOST

#
# Required Telexel variables.
#
# 1) Verify the host name for the environment variable
#    vxIpcDirectory.
# 2) Verify that vxIpcPort is a valid, unused port number.
#     a) Valid numbers are in the range 1024 and 65000.
#     b) If port number is already in use you will get an error
#        message similiar to "address already in use". The command
#        netstat will show port numbers currently in use. The file
#        /etc/services shows well known used ports.
# 3) If /opt/XXXXtel/telexel is not where you installed Telexel, update
#    the environment variable TELDIR to the correct location.
#

vxIpcDirectory=<hostname>
vxIpcPort=<port>
TELDIR=/opt/XXXXtel/telexel
PATH=${PATH}:${TELDIR}/bin
VXFILES=/var/tmp/telexel.mynah

#
# append to LD_LIBRARY_PATH/SHLIB_PATH
#
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${TELDIR}/lib
SHLIB_PATH=${SHLIB_PATH}:${TELDIR}/lib

#
# append to MANPATH
#
MANPATH=${MANPATH}:${TELDIR}/man

export vxIpcDirectory vxIpcPort TELDIR VXFILES LD_LIBRARY_PATH SHLIB_PATH PATH
MANPATH

#
# If Oracle is being used with MYNAH uncomment the following 4 statements,
# however, leave the last 2 statements of the following 4 statements
# commented if Oracle and MYNAH are installed on the same machine.
#
#oracleservername=<hostname>
#ORACLE_SID=mynah5
#TWO_TASK=mynah5
#export TWO_TASK

#
# If ORACLE is being used with MYNAH uncomment the following statements.
# Verify and correct if necessary that ORACLE_HOME is being set to the
# correct path. TNS_ADMIN must only be set if the TNS Listener files
# installed during the Oracle installation are not installed in one of
# locations that are searched automatically by Oracle.
#

#ORACLE_HOME=/opt/XXXXora/7.3.4
```

```
#ORACLE_TERM=sun5
#PATH=${PATH}:${ORACLE_HOME}/bin
#LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${ORACLE_HOME}/lib
#SHLIB_PATH=${SHLIB_PATH}:${ORACLE_HOME}/lib
#export PATH ORACLE_HOME ORACLE_TERM LD_LIBRARY_PATH
#export EPC_DISABLE=TRUE ORACLE_SID

#export TNS_ADMIN=${ORACLE_HOME}/network/admin

#
# If TOPCOM is being used uncomment the following and update
# /opt/XXXXtop/topcom is not where TOPCOM is installed
#
# TOPCOM=/opt/XXXXtop/topcom
# TBIN=${TOPCOM}/bin
# ETCPATH=/usr/public/isode/etc
# PATH=${PATH}:${TBIN}
# export TOPCOM TBIN PATH ETCPATH

#
# If I/O Concepts is being used uncomment the following statement and
# update /opt/XXXXioc/ioconcepts if this is not where I/O Concepts is
# installed. Also set IOCLM_HOST to be the name of the host name on
# which is I/O Concepts license server runs.
#
# LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/usr/dt/lib
# IOCLM_HOST=<hostname>
# export LD_LIBRARY_PATH IOCLM_HOST

#
#
# optional command line editor enable
#
EDITOR=vi
set -o vi
```

B.1.4 Example MYNAH xmyLogin File

This is an example of the `cs` `xmyLogin` file as it appears immediately following installation of the MYNAH software.

You must make the following changes to this file to reflect your environment:

- Update the directories may need to be updated
- Replace `<hostname>` with the actual machine names.
- Replace `<oracleservername>` with the name of the machine on which you installed the Oracle software.
- Replace `<port>` with the actual Telexel port (e.g., 22100).
- Replace `XXXXmyn` with either `SUNWmyn` or `HPUXmyn`, depending on your operating system.

After you have made the necessary changes to this file it can be sourced into the `.login` files for all MYNAH users.

```
# COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
# All Rights Reserved.
#
# PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia
# and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)xmyLogin.eg53.3
# created on 98/11/20 at 11:04:59
#
# @(#)sample login for madmin 96/06/11 SOL (csh version)
#
# This is an example of a MYNAH System profile that
# can be sourced into MYNAH user's .login to give
# them access to the system.
#
#
# XMYDIR is the path to a directory which is symbolically linked to
# the current release of MYNAH, e.g.,
# ln -s /opt/XXXXmyn/MYNAH/releases/MYNAH_5.0 /opt/XXXXmyn/mynah
#
setenv XMYDIR /opt/XXXXmyn/mynah
#
# XMYHOME is the path to a directory which is the MYNAH run environment
#
setenv XMYHOME /opt/XXXXmyn/mynah
```

```

#
# create or append to LD_LIBRARY_PATH/SHLIB_PATH
#
if ( $?LD_LIBRARY_PATH ) then
    setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:${XMYDIR}/lib:/usr/openwin/lib"
else
    setenv LD_LIBRARY_PATH "${XMYDIR}/lib:/usr/openwin/lib"
endif

if ( $?SHLIB_PATH ) then
    setenv SHLIB_PATH "${SHLIB_PATH}:${XMYDIR}/lib:/usr/openwin/lib"
else
    setenv SHLIB_PATH "${XMYDIR}/lib:/usr/openwin/lib"
endif

#
# create or append to MANPATH
#

if ( $?MANPATH ) then
    setenv MANPATH "${MANPATH}:${XMYDIR}/man"
else
    setenv MANPATH "${XMYDIR}/man"
endif

setenv TCL_LIBRARY "${XMYDIR}/lib/tcl"
setenv TCLX_LIBRARY "${XMYDIR}/lib/tcl"
set path = ($path $XMYDIR/bin)

#
# The host name that the MYNAH license server runs on
#
setenv LSHOST <hostname>

#
# Required Telexel variables.
#
# 1) Verify the host name for the environment variable
#    vxIpcDirectory.
# 2) Verify that vxIpcPort is a valid, unused port number.
#    a) Valid numbers are in the range 1024 and 65000.
#    b) If port number is already in use you will get an error
#       message similar to "address already in use". The command
#       netstat will show port numbers currently in use. The file
#       /etc/services shows well known used ports.
# 3) If /opt/XXXXtel/telexel is not where you installed Telexel, update
#    the environment variable TELDIR to the correct location.
#

setenv vxIpcDirectory <hostname>
setenv vxIpcPort <port>
setenv TELDIR /opt/XXXXtel/telexel
set path = ($path $TELDIR/bin)
setenv VXFILES /var/tmp/telexel.mynah
setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:${TELDIR}/lib"
setenv SHLIB_PATH "${SHLIB_PATH}:${TELDIR}/lib"
setenv MANPATH "${MANPATH}:${TELDIR}/man"

```



```
#
# If Oracle is being used with MYNAH uncomment the following statement.
#
#setenv TWO_TASK mynah5

#
# If ORACLE is being used with MYNAH uncomment the following statements.
# Verify and correct if necessary that ORACLE_HOME is being set to the
# correct path. TNS_ADMIN must only be set if the TNS Listener files
# installed during the Oracle installation are not installed in one of
# locations that are searched automatically by Oracle.
#

#setenv ORACLE_HOME /opt/XXXXora/7.3.4
#setenv ORACLE_SID mynah5
#setenv ORACLE_TERM sun5
#set path = ( $path $ORACLE_HOME/bin)
#setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:${ORACLE_HOME}/lib"
#setenv SHLIB_PATH "${SHLIB_PATH}:${ORACLE_HOME}/lib"
#setenv EPC_DISABLE TRUE
#
#setenv TNS_ADMIN ${ORACLE_HOME}/network/admin

#
# If TOPCOM is being used uncomment the following and update
# /opt/XXXXtop/topcom is not where TOPCOM is installed
#
# setenv TOPCOM /opt/XXXXtop/topcom
# setenv TBIN ${TOPCOM}/bin
# setenv ETC_PATH /usr/public/isode/etc
# set path = ( $path $TBIN)

#
# If I/O Concepts is being used uncomment the following statement and
# update /opt/XXXXioc/ioconcepts if this is not where I/O Concepts is
# installed. Also set IOCLM_HOST to be the name of the host name on
# which is I/O Concepts license server runs.
#
# setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:/usr/dt/lib"
# setenv IOCLM_HOST <hostname>

#
# optional command line editor enable
#
setenv EDITOR vi
```

B.1.5 Example Solaris MYNAH Start-up File (S99mynah.eg)

This is an example Solaris MYNAH processes start-up file. Rename this file, e.g., *S99mynah*, and place it in */etc/rc3.d* with **root** as the owner. This is for the server only. You may need to update the required paths and directories.

NOTE — A logical link to this file should be set up in */etc/rc0.d* as follows:

```
ln -s /etc/rc3.d/S99mynah K01mynah

#!/bin/ksh

# COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
# All Rights Reserved.
#
# PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia
# and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)S99mynah.eg 50.7
# created on 96/09/18 at 15:16:02

#
# This is a sample startup file for MYNAH. It should be copied
# to the /etc/rc3.d directory with root as the owner. This is
# for the server only. Directories by need to be updated.
#
# Also a logical link to this file should be set
# up in /etc/rc0.d as follows:
# ln -s /etc/rc3.d/S99mynah K01mynah

vxIpcPort=<IPC Port>
vxIpcDirectory=<hostname>
TELDIR=/opt/SUNWtel/telexel
XMYDIR=/opt/SUNWmyn/mynah
XMYHOME=/opt/SUNWmyn/mynah
export vxIpcPort vxIpcDirectory TELDIR XMYDIR XMYHOME
LD_LIBRARY_PATH=${XMYDIR}/lib:${TELDIR}/lib:/usr/openwin/lib
PATH=/usr/bin:/usr/sbin:/usr/ccs/bin:/usr/openwin/bin:/usr/ucb:/etc:
    ${XMYDIR}/bin:${TELDIR}/bin
export LD_LIBRARY_PATH PATH

case "$1" in
'start')
    # Start up Mynah and Telexel

# removing Mynah pip files that might be left over
if [ -f ${XMYHOME}/run/oa/pip.oa.<hostname> ]; then
    (echo 'Deleting stale Mynah pip files.') >/dev/console
    (echo 'Deleting stale Mynah pip files.') >>/var/adm/messages
    (/bin/su - madadmin -c "${XMYDIR}/bin/.xmyRemovePips all"; )
```

```
fi

if [ -f ${XMYDIR}/bin/xmyStartOA ]; then
    (echo 'starting Mynah & Telexel processes.') >/dev/console
    (echo 'starting Mynah & Telexel processes.') >>/var/adm/messages
    ( /bin/su - madmin -c "${XMYDIR}/bin/xmyStartUp"; )
#    (echo 'starting Mynah Collector Process.') >/dev/console
#    (echo 'starting Mynah Collector Process.') >>/var/adm/messages
#    ( /bin/su - madmin -c "${XMYDIR}/bin/xmyStartCL -n <collector name>"; )
fi

    ;;

'stop')

if [ -f ${XMYDIR}/bin/xmyStopOA ]; then
    (echo 'stopping Mynah processes.') >/dev/console
    (echo 'stopping Mynah processes.') >>/var/adm/messages
    ( /bin/su - madmin -c "${XMYDIR}/bin/xmyShutdown <hostname>"; )
#    ( /bin/su - madmin -c "${XMYDIR}/bin/xmyStopCL -n <collector name>"; )
fi

    ;;

*)

    ;;

esac
```

B.2 Example Telexel Installation Session

This is an example BAIST installation session of the Telexel System software.

```

Telcordia Application Installation Setup Tool
- - - - -
                BAIST 2.1

COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
                All Rights Reserved.

PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.

                A UNIX Packaging and Installation Tool

                For further information on BAIST
                Contact: Raymond C. Gray
                        BAIST Project Manager
                        (732)699-7960

```

Please Wait ...

WARNING: You are using LOCAL BCRDB
 ***** BAIST Installation Tool Release 2.1 *****
 Initializing. Please wait.

```

BAIST 2.1 FLOW CONTROL
PRODUCT MENU
  Archived Products List

```

- 1) MYNAH
- 2) TOPCOM
- 3) TELEXEL

- I) Installed Products
- R) Registered Products
- E) Exit

```

Enter your Selection: 3
Load TELEXEL? [Y]: Y
Extracting Application Profile

Getting the name of PRODUCT_HOME dir

Where should the TELEXEL directory be created ?
Enter a full pathname starting with /, or q to Quit processing
: /opt/SUNWtel
Getting the name of RELEASE_HOME dir

```

Making the PRODUCT_HOME dir
Making the RELEASE_HOME dir

Who should own the TELEXEL application ?
Enter a valid login name :**admin**

Running PRELOAD script

Extracting the TELEXEL application
Please wait. This may take a while ...
9096 of 9096 blocks loaded 100% COMPLETED
Product TELEXEL TELEXEL_6.0 loaded successfully

BAIST 2.1 FLOW CONTROL
PRODUCT MENU
Archived Products List

- 1) MYNAH 5.3
- 2) TELEXEL 7.1
- 3) TOPCOM 5.3.8

- I) Installed Products
- R) Registered Products
- E) Exit

Enter your Selection:**E**

B.3 Delivered Example Oracle Files

This section contains all of the example Oracle start-up and configuration files that are delivered with the MYNAH System.

B.3.1 Example Solaris Oracle Start-up File (S96oracle)

This is an example Solaris Oracle start-up file. It should be moved to */etc/rc3.d* with **root** as the owner. You may need to update the required directories.

```
#!/bin/ksh

# COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
# All Rights Reserved.
#
# PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia
# and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)S96oracle.eg50.5
# created on 96/09/26 at 17:24:32

# This is a sample startup file for oracle. It should
# be moved to the /etc/rc3.d directory with root as the
# owner. Directories may need to be updated.

#
# Required update
#
# If /opt/SUNWora/oracle is not where you installed Oracle, update
# the environment variable ORACLE_HOME to the correct location.
#
ORACLE_HOME=/opt/SUNWora/oracle
export ORACLE_HOME

case "$1" in
'start')
    # Start up Oracle

# Delete nasty Oracle tmp directory
if [ -d /var/tmp/o ]; then
    rm -rf /var/tmp/o
fi

if [ -f ${ORACLE_HOME}/bin/dbstart ]; then
    (echo 'starting oracle7.') >/dev/console
    (echo 'starting oracle7.') >/var/adm/messages
    ( /bin/su - oracle -c "${ORACLE_HOME}/bin/dbstart"; )
    (echo 'starting oracle sqlnet server.') >/dev/console
```

```
(echo `starting oracle sqlnet server.`) >/var/adm/messages
( /bin/su - oracle -c "export PATH=${ORACLE_HOME}/bin:
    $PATH ; ${ORACLE_HOME}/bin/tcpctl start"; )
# Start sqlnet V2 listner
# (echo `starting oracle sqlnet V2 server.`) >/dev/console
# (echo `starting oracle sqlnet V2 server.`) >/var/adm/messages
( /bin/su - oracle -c " export PATH=${ORACLE_HOME}/bin:/usr/ccs/bin:
    $PATH ; ${ORACLE_HOME}/bin/lsnrctl start"; )

fi

;;
`stop`)

if [ -f ${ORACLE_HOME}/bin/tcpctl ]; then
    (echo `stopping oracle7.`) >/dev/console
    (echo `stopping oracle7.`) >/var/adm/messages
    ( /bin/su - oracle -c "${ORACLE_HOME}/bin/dbshut"; )
# ( /bin/su - oracle -c " export PATH=${ORACLE_HOME}/bin:$PATH ; \
#     ${ORACLE_HOME}/bin/lsnrctl stop"; )
fi
;;
*)
;;
esac
```

B.3.2 Example Oracle Configuration File

This is an example Oracle configuration file, *configmynah5.ora*. For this example the Oracle software is installed into */opt/SUNWora/7.2.3.2*. If you use a different directory, the directory statements in this example will need to be updated.

```
# COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
# All Rights Reserved.
#
# PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia
# and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)configmynah5.ora 50.2
# created on 96/07/16 at 17:26:27

#
# $Header: cnfg.orc 7001200.2 93/04/26 14:58:22 eruben Osd<unix> $ Copyr (c) 1992
Oracle
#

#####
# NOTE:
# This is an example configmynah5.ora file. Directories will need
# to be updated if different. For this example the oracle software
# is installed into /opt/SUNWora/oracle.
#####

# cnfg.ora - instance configuration parameters

control_files          = (/opt/SUNWora/oracle/dbs/ctrl1mynah5.ctl,
                        /opt/SUNWora/oracle/mynah5/datafiles/d02/ctrl2mynah5.ctl,
                        /opt/SUNWora/oracle/mynah5/datafiles/d03/ctrl3mynah5.ctl)

# Below for possible future use...
#init_sql_files        = (?/dbs/sql.bsq,
#                          ?/rdbms/admin/catalog.sql,
#                          ?/rdbms/admin/expvew.sql)
background_dump_dest   = /opt/SUNWora/oracle/rdbms/log
core_dump_dest         = /tmp
user_dump_dest         = /opt/SUNWora/oracle/rdbms/log
#log_archive_dest= /opt/SUNWora/oracle/dbs/arch/arch.log
db_block_size         = 2048

db_name                = mynah5
```


B.3.3 Example Oracle Initialization Script (initmynah5.ora)

This is an example Oracle initialization file, *initmynah5.ora*. Directories will need to be updated if different. For this example the oracle software is installed into */opt/SUNWora/7.2.3.2*.

```
# COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
# All Rights Reserved.
#
# PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia
# and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)initmynah5.ora 50.2
# created on 96/07/16 at 17:26:40

# COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
# All Rights Reserved.
#
# PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia
# and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)initmynah5.ora 50.1
# created on 96/07/03 at 15:58:57

#
# $Header: initx.orc 7001300.3 93/06/16 12:28:26 mkrishna Osd<unix> $ Copyr (c)
1992 Oracle
#

#####
# NOTE:
# This is an example initmynah5.ora file. Directories will need
# to be updated if different. For this example the oracle software
# is installed into /opt/SUNWora/oracle.
#####

# include database configuration parameters
ifile = /opt/SUNWora/oracle/dbs/configmynah5.ora

#rollback_segments = (r01)
rollback_segments= (r01,r02,r03,r04)

#####
# Example INIT.ORA file
#
# This file is provided by Oracle Corporation to help you customize
# your RDBMS installation for your site. Important system parameters
# are discussed, and example settings given.
```

```

#
# Some parameter settings are generic to any size installation.
# For parameters that require different values in different size
# installations, three scenarios have been provided: SMALL, MEDIUM
# and LARGE. Any parameter that needs to be tuned according to
# installation size will have three settings, each one commented
# according to installation size.
#
# Use the following table to approximate the SGA size needed for the
# three scenarios provided in this file:
#
#          -----Installation/Database Size-----
#          SMALL          MEDIUM          LARGE
# Block      2K      4500K          6800K          17000K
# Size      4K      5500K          8800K          21000K
#
# To set up a database that multiple instances will be using, place
# all instance-specific parameters in one file, and then have all
# of these files point to a master file using the IFILE command.
# This way, when you change a public
# parameter, it will automatically change on all instances. This is
# necessary, since all instances must run with the same value for many
# parameters. For example, if you choose to use private rollback segments,
# these must be specified in different files, but since all gc_*
# parameters must be the same on all instances, they should be in one file.
#
# INSTRUCTIONS: Edit this file and the other INIT files it calls for
# your site, either by using the values provided here or by providing
# your own. Then place an IFILE= line into each instance-specific
# INIT file that points at this file.
#####

# tuning parameters

db_files =1020

db_file_multiblock_read_count = 8          # SMALL
# db_file_multiblock_read_count = 16      # MEDIUM
# db_file_multiblock_read_count = 32      # LARGE

db_block_buffers = 200                    # SMALL
# db_block_buffers = 550                  # MEDIUM
# db_block_buffers = 3200                 # LARGE

shared_pool_size = 3500000                # SMALL
# shared_pool_size = 6000000             # MEDIUM
# shared_pool_size = 9000000             # LARGE

log_checkpoint_interval = 10000

processes = 50                            # SMALL
# processes = 100                        # MEDIUM
# processes = 200                        # LARGE

dml_locks = 100                           # SMALL
# dml_locks = 200                        # MEDIUM
# dml_locks = 500                        # LARGE

```

```
log_buffer = 8192 # SMALL
# log_buffer = 32768 # MEDIUM
# log_buffer = 163840 # LARGE

sequence_cache_entries = 10 # SMALL
# sequence_cache_entries = 30 # MEDIUM
# sequence_cache_entries = 100 # LARGE

sequence_cache_hash_buckets = 10 # SMALL
# sequence_cache_hash_buckets = 23 # MEDIUM
# sequence_cache_hash_buckets = 89 # LARGE

# audit_trail = true # if you want auditing
# timed_statistics = true # if you want timed statistics
max_dump_file_size = 10240 # limit trace file size to 5 Meg each

# log_archive_start = true # if you want automatic archiving

mts_dispatchers="ipc,1"
mts_max_dispatchers=10
mts_servers=1
mts_max_servers=10
mts_service=mynah5
mts_listener_address="(ADDRESS=(PROTOCOL=ipc)(KEY=mynah5))"
nls_date_format = "DD-MON-YYYY"
remote_os_authent = true
```

B.3.4 Example Oracle *crdbmynah5.sql* File

This is an example Oracle *crdbmynah5.sql* file. For this example the Oracle software is installed into */opt/SUNWora/7.2.3.2*. If you use a different directory, the directory statements in this example will need to be updated.

```
# COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
# All Rights Reserved.
#
# PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia
# and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)crdbmynah5.sql50.2
# created on 96/07/16 at 17:26:15

REM #####
REM # NOTE:
REM # This is an example crdbmynah5.sql file. Directories will need
REM # to be updated if different. For this example the oracle software
REM # is installed into /opt/SUNWora/oracle.
REM #####

REM * Set terminal output and command echoing on; log output of this script.
REM *
#set termout on
#set echo on
spool /opt/SUNWora/oracle/dbs/crdbmynah5.lst

REM * Start the <sid> instance (ORACLE_SID here must be set to <sid>).
REM *

REM * Create the <dbname> database.
REM * SYSTEM tablespace configuration guidelines:
REM *   General-Purpose ORACLE RDBMS    5Mb
REM *   Additional dictionary for applications10-50Mb
REM * Redo Log File configuration guidelines:
REM *   Use 3+ redo log files to relieve ``cannot allocate new log...`` waits.
REM *   Use ~100Kb per redo log file per connection to reduce checkpoints.
REM *
create database "mynah5"
maxinstances 1
maxlogfiles 16
character set "US7ASCII"
datafile
  '/opt/SUNWora/oracle/mynah5/datafiles/d01/systmynah5.dbf' size 25M
logfile
  '/opt/SUNWora/oracle/mynah5/logfiles/log1mynah5.dbf' size 500k,
  '/opt/SUNWora/oracle/mynah5/logfiles/log2mynah5.dbf' size 500k,
  '/opt/SUNWora/oracle/mynah5/logfiles/log3mynah5.dbf' size 500k;

disconnect
spool off
```

B.3.5 Example Oracle *crdb2mynah5.sql* File

This is an example Oracle *crdb2mynah5.sql* file. For this example the Oracle software is installed into */opt/SUNWora/7.2.3.2*. If you use a different directory, the directory statements in this example will need to be updated.

This script takes care off all commands necessary to create an OFA compliant database after the **create database** command has succeeded.

```
# COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
# All Rights Reserved.
#
# PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia
# and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)crdb2mynah5.sql50.2
# created on 96/07/16 at 17:26:03

# COPYRIGHT (c) 1996 Telcordia Technologies, Inc.,
# All Rights Reserved.
#
# PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia
# and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)crdb2mynah5.sql50.1
# created on 96/07/03 at 15:58:45

REM #####
REM # NOTE:
REM # This is an example crdb2mynah5.sql file. Directories will need
REM # to be updated if different. For this example the oracle software
REM # is installed into /cowboyl/opt/SUNWora/oracle.
REM #####

REM * This script takes care off all commands necessary to create
REM * an OFA compliant database after the CREATE DATABASE command has
REM * succeeded.

REM * Set terminal output and command echoing on; log output of this script.
REM *
#set termout on
#set echo on
#spool 2-rdbms.lst
spool /cowboyl/opt/SUNWora/oracle/dbs/crdb2mynah5s1.lst

connect internal
```

```
REM # install data dictionary views:
@/cowboyl/opt/SUNWora/oracle/rdbms/admin/catalog.sql

REM * Create additional rollback segment in SYSTEM before creating tablespace.
REM *
connect internal
create rollback segment r0 tablespace system
  storage (initial 16k next 16k minextents 2 maxextents 20);

REM * Use ALTER ROLLBACK SEGMENT ONLINE to put r0 online without shutting
REM * down and restarting the database.
REM *
alter rollback segment r0 online;

REM * Create a tablespace for rollback segments.
REM * Rollback segment configuration guidelines:
REM *   1 rollback segments for every 4 concurrent xactions.
REM *   No more than 50 rollback segments.
REM *   All rollback segments the same size.
REM *   Between 2 and 4 homogeneously-sized extents per rollback segment.
REM *   Attempt to keep rollback segments to 4 extents.
REM *
create tablespace rbs datafile
  '/cowboyl/opt/SUNWora/oracle/mynah5/datafiles/d02/rbsmynah5sl.dbf' size
4M
default storage (
  initial      128k
  next        128k
  pctincrease  0
  minextents  2
);

REM * Create a tablespace for temporary segments.
REM * Temporary tablespace configuration guidelines:
REM *   Initial and next extent sizes = k * SORT_AREA_SIZE, k in {1,2,3,...}.
REM *
create tablespace temp datafile
  '/cowboyl/opt/SUNWora/oracle/mynah5/datafiles/d01/tempmynah5sl.dbf' size
550k
default storage (
  initial      256k
  next        256k
  pctincrease  0
  optimal     1M
);

REM * Create a tablespace for database tools.
REM *
create tablespace tools datafile
  '/cowboyl/opt/SUNWora/oracle/mynah5/datafiles/d03/toolmynah5sl.dbf' size
15M;

REM * Create a tablespace for mynah5 databases.
REM *
create tablespace my5 datafile
  '/cowboyl/opt/SUNWora/oracle/mynah5/datafiles/d03/my5mynah5sl.dbf' size
15M;
```

```
REM * Create a tablespace for miscellaneous database user activity.
REM *
create tablespace users datafile
    '\cowboy1/opt/SUNWora/oracle/mynah5/datafiles/d01/usrmynah5sl.dbf' size
1M;

REM * Create rollback segments.
REM *
create rollback segment r01 tablespace rbs;
create rollback segment r02 tablespace rbs;
create rollback segment r03 tablespace rbs;
create rollback segment r04 tablespace rbs;

REM * Use ALTER ROLLBACK SEGMENT ONLINE to put rollback segments online
REM * without shutting down and restarting the database. Only put one
REM * of the rollback segments online at this time so that it will always
REM * be the one used. When the user shuts down the database and starts
REM * it up with initSID.ora, all four will be brought online.
REM *
alter rollback segment r01 online;
REM * alter rollback segment r02 online;
REM * alter rollback segment r03 online;
REM * alter rollback segment r04 online;

REM * Since we've created and brought online 2 more rollback segments,
REM * we no longer need the second rollback segment in the SYSTEM tablespace.
alter rollback segment r0 offline;
drop rollback segment r0;

REM * Alter SYS and SYSTEM users.
REM *
alter user sys temporary tablespace temp;
#revoke resource from system;
#revoke resource on system from system;
#grant resource on tools to system;
alter user system default tablespace tools temporary tablespace temp;

REM * For each DBA user, run DBA synonyms SQL script. Don't forget that EACH
REM * DBA USER created in the future needs dba_syn.sql run from its account.
REM *
connect system/manager
@/cowboy1/opt/SUNWora/oracle/rdbms/admin/catdbsyn.sql

spool off
```

B.3.6 Example Oracle *crdb3mynah5.sql* File

This is an example Oracle *crdb3mynah5.sql* file. For this example the Oracle software is installed into */opt/SUNWora/7.2.3.2*. If you use a different directory, the directory statements in this example will need to be updated.

```

/*
 * COPYRIGHT (c) 1997 Telcordia Technologies, Inc.,
 * All Rights Reserved.
 *
 * PROPRIETARY - TELCORIDA AND AUTHORIZED CLIENTS ONLY.
 *
 * This document contains proprietary information that shall
 * be distributed or routed only within Telcordia
 * and its authorized clients, except
 * with written permission of Telcordia.
 */

/*
 * @(#)crdb3mynah5.sql52.1
 * created on 97/05/27 at 09:58:52
 */

static char crdb3mynah5_sql[] = " @(#)crdb3mynah5.sql52.1";

/* Prevent warning message from CC about sccs string defined but not used */
#ifdef __cplusplus
inline void static dummy_crdb3mynah5_sql() { if (crdb3mynah5_sql); }
#endif

Rem
Rem cd $ORACLE_HOME/rdbms/admin
Rem
Rem $ORACLE_HOME/bin/sqlplus sys/change_on_install << "EOF
connect sys/change_on_install
@$ORACLE_HOME/rdbms/admin/standard
@$ORACLE_HOME/rdbms/admin/catproc
@$ORACLE_HOME/rdbms/admin/catalog
@$ORACLE_HOME/rdbms/admin/dbmspipe

create user mynah identified by mynah
default tablespace my5
temporary tablespace temp;

grant execute any procedure to mynah;

grant connect to mynah;

grant resource to mynah;

commit;
Rem exit;
Rem EOF

Rem cd $ORACLE_HOME/sqlplus/admin
Rem $ORACLE_HOME/bin/sqlplus system/manager << "EOF

```



```
connect system/manager  
@$ORACLE_HOME/sqlplus/admin/pupbld  
exit;  
Rem EOF
```

B.3.7 Example xmyCreateSequences Execution

This program will create all the sequences needed by the MYNAH database in order to operate. It assumes that the database tables have been created. If not, then quit this process and create the tables before retrying.

Continue [y/n]: **y**

Database connection opened.

```
Sending sql: CREATE SEQUENCE xmyCompareResult_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyCompound_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyData_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyDocument_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyField_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyFieldValue_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyFormat_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyHierarchy_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyHierarchyNode_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyIssue_Sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyKeyword_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyPerson_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyProcedure_Sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyProcedureLibrary_Sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyRelation_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyRequirement_Sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyResult_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyResultDelta_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyResource_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyResourceUsage_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyRunTime_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyScript_Sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyStep_sequence MINVALUE 1 CACHE 500
ending sql: CREATE SEQUENCE xmyStepList_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmySutInfo_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyTest_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyTestVersion_Sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyTuple_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyUserEnumValue_sequence MINVALUE 1 CACHE 500
Sending sql: CREATE SEQUENCE xmyValueGroup_sequence MINVALUE 1 CACHE 500
```

B.3.8 Example xmyCreateTemplates Execution

```
Creating template objects.  
Person template successfully locked  
Mynah administrator successfully created  
UserEnumValue template successfully created  
UserEnumValue template successfully locked  
Installing userEnumValue templates.  
Template userEnumValue created  
defaults userEnumValues installed.  
continuing with template objects.  
Person template successfully locked  
Script template successfully created  
Script template successfully locked  
Hierarchy template successfully created  
Hierarchy template successfully locked  
HierarchyNode template successfully created  
HierarchyNode template successfully locked  
HierarchyNode for demo successfully created  
SutInfo template successfully created  
SutInfo template successfully locked  
sutInfo demo successfully created  
RunTime template successfully created  
RunTime template successfully locked  
Keyword template successfully created  
Keyword template successfully locked  
Format template successfully created  
Format template successfully locked  
Compound template successfully created  
Compound template successfully locked  
Field template successfully created  
Field template successfully locked  
Test template successfully created  
Test template successfully locked  
forTestVersionHierarchyNode template successfully created  
TestVersion template successfully created  
TestVersion template successfully locked  
StepList template successfully created  
StepList template successfully locked  
Step template successfully created  
Step template successfully locked  
Result template successfully created  
Result Template successfully locked  
ResultDelta template successfully created  
ResultDelta template successfully locked  
CompareResult template successfully created  
CompareResult template successfully locked  
Issue template successfully created  
Issue template successfully locked  
procedureLibrary template successfully created  
ProcedureLibrary template successfully locked  
procedure template successfully created  
Procedure template successfully locked  
requiremnt template successfully created  
Requirement template successfully locked  
FieldValue template successfully created  
FieldValue template successfully locked
```

Relation template successfully created
Relation template successfully locked
Tuple template successfully created
Tuple template successfully locked
ValueGroup template successfully created
ValueGroup template successfully locked
Template Objects created

B.3.9 xample xmyDropSequences

```
*WARNING*: this program will DELETE all the sequences used
by mynah database in order to operate.
Continue [y/n]: y
Database connection opened.
Sending sql: DROP SEQUENCE xmyCompareResult_sequence
Sending sql: DROP SEQUENCE xmyCompound_sequence
Sending sql: DROP SEQUENCE xmyData_sequence
Sending sql: DROP SEQUENCE xmyDocument_sequence
Sending sql: DROP SEQUENCE xmyField_sequence
Sending sql: DROP SEQUENCE xmyFieldValue_sequence
Sending sql: DROP SEQUENCE xmyFormat_sequence
Sending sql: DROP SEQUENCE xmyHierarchy_sequence
Sending sql: DROP SEQUENCE xmyHierarchyNode_sequence
Sending sql: DROP SEQUENCE xmyIssue_Sequence
Sending sql: DROP SEQUENCE xmyKeyword_sequence
Sending sql: DROP SEQUENCE xmyPerson_sequence
Sending sql: DROP SEQUENCE xmyProcedure_Sequence
Sending sql: DROP SEQUENCE xmyProcedureLibrary_Sequence
Sending sql: DROP SEQUENCE xmyRelation_sequence
Sending sql: DROP SEQUENCE xmyRequirement_Sequence
Sending sql: DROP SEQUENCE xmyResource_sequence
Sending sql: DROP SEQUENCE xmyResourceUsage_sequence
Sending sql: DROP SEQUENCE xmyResult_sequence
Sending sql: DROP SEQUENCE xmyResultDelta_sequence
Sending sql: DROP SEQUENCE xmyRunTime_sequence
Sending sql: DROP SEQUENCE xmyScript_Sequence
Sending sql: DROP SEQUENCE xmyStep_sequence
Sending sql: DROP SEQUENCE xmyStepList_sequence
Sending sql: DROP SEQUENCE xmySutInfo_sequence
Sending sql: DROP SEQUENCE xmyTest_sequence
Sending sql: DROP SEQUENCE xmyTestVersion_Sequence
Sending sql: DROP SEQUENCE xmyTuple_sequence
Sending sql: DROP SEQUENCE xmyUserEnumValue_sequence
Sending sql: DROP SEQUENCE xmyValueGroup_sequence
```

B.3.10 Example xmyCreate Tables

```
Database connection opened.
Tables do not exist, creating them.
```

B.3.11 Example xmyDropTables

```
*WARNING*: this program will DELETE all the tables
in the mynah database
Continue [y/n]: y
Database connection opened.
Tables were there. Dropping all of them!
```

B.3.12 Example root.sh Run

```
# ./root.sh
Running ORACLE7 root.sh script...
The following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME=  /opt/SUNWora/V7.2.3.2
    ORACLE_SID=   mynah5
Are these settings correct (Y/N)? [Y]: Y

Enter the full pathname of the local bin directory [/opt/bin]:
/usr/local/bin

Checking for "oracle" user id...
ORACLE_HOME does not match the home directory for oracle.
Okay to continue? [N]: Y

Creating /var/opt/oracle/oratab file...
Updating /var/opt/oracle/oratab file...

Please raise the ORACLE owner's ulimit as per the IUG.

Leaving common section of ORACLE7 root.sh.
Setting orasrv file protections
```

Appendix C: Building the DCE Emulated Client and Emulated Server

This appendix describes the steps required to manually build the DCE emulated client and the emulated server executables using the tools developed by the MYNAH System. If possible, the administrator should use the template Makefile, described in the following sections.

If this is not possible, the manual steps required to build the executables are described in Appendix C.2.

C.1 Template Makefile

The following template makefile may be used for building the emulated client and server executables. It is distributed under `$XMYDIR/examples/dce/xmyDceMakefile.eg`. This makefile has been tested with GNU's **gnumake** and with Sun's **make**.

The variables near the beginning of the file should be set as shown in Table C-1.

Table C-1. DCE Makefile variables

Variable Name	Description	Default Value
TCLDCE	Root of MYNAH DCE installation	<code>\$(XMYDIR)</code>
TCLDCELIB	MYNAH libraries	<code>\$(TCLDCE)/lib</code>
TCLDCEINC	MYNAH header files	<code>\$(TCLDCE)/include</code>
PARSER	Path to MYNAH's idl parser	<code>\$(TCLDCE)/bin/xmyDceParser</code>
LIBRARY	Path to MYNAH DCE library	<code>\$(TCLDCE)/lib/libxmyDce.a</code>
RW	Root of RogueWave Tools.h++ installation	None. Enter the path for your system. This value is required.
RWLIB	RogueWave libraries	<code>\$(RW)/lib</code>
RWINC	RogueWave header files	<code>\$(RW)/include</code>
TELDIR	Root of TELEXEL installation	None. Enter the path for your system. This value is required.
TELLIB	TELEXEL libraries	<code>\$(TELDIR)/lib</code>
TELINC	TELEXEL header files	<code>\$(TELDIR)/include</code>
INTERFACE	Basename of interface	None. Enter the name of the interface without the suffix. This value is required.
DEBUGFLAG	Used to turn on debugging	<i>unset</i>
CC	command for C compiler	<i>clcc</i>
CXX	command for C++ compiler	CC

NOTE — This delivered example template makefile can be used to compile servers and clients on Solaris. If you are using HP-UX, you must make the following changes:

1. Change the **CC** setting to the *aCC* compiler.
2. Add **-D_PTHREADS_DRAFT4** to the **CFLAGS** and **CXXFLAGS** settings.
3. The variable **\$RW** is no longer required since it is a part of *aCC*.
4. Change **-lrwtool_mt** to **-lrwtool** in the **CLIENT_LIBS** and **SERVER_LIBS** settings.

```
# COPYRIGHT (c) 1996 Telcordia Technologies Inc,
# All Rights Reserved.
#
# PROPRIETARY - Telcordia AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia Technologies
# (Telcordia), and its authorized clients, except
# with written permission of Telcordia.
#
# @(#)xmyDceMakefile.eg 54.2
# created on 99/05/05 at 16:45:23

TCLDCE      = $(XMYDIR)
TCLDCELIB   = $(TCLDCE)/lib
TCLDCEINC   = $(TCLDCE)/include

PARSER      = $(TCLDCE)/bin/xmyDceParser
LIBRARY     = $(TCLDCELIB)/libxmyDce.a

#RW         = path to RogueWave root
RWINC       = $(RW)/include
RWLIB       = $(RW)/lib

# TELEXEL trace is no longer required. It is now optional.
#
# If compiling and linking with TELEXEL trace:
# - add -DTRACE -I$(TELINC) to CFLAGS and CXXFLAGS
# - add -L$(TELLIB) -lvx to SERVER_LIBS and CLIENT_LIBS
# - change LIBRARY above to $(TCLDCELIB)/libxmyDceTrace.a
# - ensure TELDIR, TELINC, TELLIB are set correctly

#TELDIR     = path to TELEXEL root
TELINC      = $(TELDIR)/include
TELLIB     = $(TELDIR)/lib

INTERFACE   = name-of-interface (without suffix)
ACF_FILE    = # name of acf file (with suffix)
OTHER_IDLS  = # other idl files
```



```
# uncomment the next line to generate debuggable executables
DEBUGFLAG      = -g

default:       client server

#####

SERVER_LIBS    = $(LIBRARY) -L$(RW)/lib -lrwtool_mt -L$(XMYDIR)/lib -ltclx -ltcl
               -lm -ldce -lnsl -lsocket -lthread
CLIENT_LIBS    = $(LIBRARY) -L$(RW)/lib -lrwtool_mt -L$(XMYDIR)/lib -ltclx -ltcl
               -lm -ldce -lnsl -lsocket -lthread

CLIENT_OBJS    = $(INTERFACE)_cstub.o $(INTERFACE)-data.o $(INTERFACE)-client.o
               $(INTERFACE)-client-AppInit.o $(INTERFACE)-client-main.o
SERVER_OBJS    = $(INTERFACE)_sstub.o $(INTERFACE)-data.o $(INTERFACE)-server.o
               $(INTERFACE)-server-AppInit.o $(INTERFACE)-server-main.o

CFLAGS         = -I$(TCLDCEINC) -I$(RW)/include -I. $(DEBUGFLAG) -D_REENTRANT
               -Dvolatile=
CXXFLAGS       = -I$(TCLDCEINC) -I$(RW)/include -I. $(DEBUGFLAG) -D_REENTRANT
               -Dvolatile=
LDFLAGS        = $(DEBUGFLAG)

CC             = cc
CXX            = CC

#####

client:        .parser_run .idl_run $(CLIENT_OBJS) $(LIBRARY)
               $(CXX) -o $@.new $(CLIENT_OBJS) $(CLIENT_LIBS)
               @rm -f $@; mv $@.new $@

client.pure:   .parser_run .idl_run $(CLIENT_OBJS) $(LIBRARY)
               purify $(CXX) -o $@.new $(CLIENT_OBJS) $(CLIENT_LIBS)
               @rm -f $@; mv $@.new $@

server:       .parser_run .idl_run $(SERVER_OBJS) $(LIBRARY)
               $(CXX) -o $@.new $(SERVER_OBJS) $(SERVER_LIBS)
               @rm -f $@; mv $@.new $@

server.pure:  .parser_run .idl_run $(SERVER_OBJS) $(LIBRARY)
               purify $(CXX) -o $@.new $(SERVER_OBJS) $(SERVER_LIBS)
               @rm -f $@; mv $@.new $@

#####

OTHER_HEADERS = $(OTHER_IDLS:.idl=.h)

PARSER_OUTPUTS = $(INTERFACE)-client.C $(INTERFACE)-client-AppInit.c \
                 $(INTERFACE)-client-main.C \
                 $(INTERFACE)-server.C $(INTERFACE)-server-AppInit.c \
                 $(INTERFACE)-server-main.C \
                 $(INTERFACE)-data.C $(INTERFACE)-data.h \
                 .parser_run

IDL_OUTPUTS    = $(INTERFACE)_cstub.o $(INTERFACE)_sstub.o $(INTERFACE).h \
                 .idl_run
```

```

$(PARSER_OUTPUTS):    $(INTERFACE).idl $(ACF_FILE) $(PARSER)
    @rm -f .parser_run
    $(PARSER) $(INTERFACE).idl
    @touch .parser_run

$(IDL_OUTPUTS):      $(INTERFACE).idl $(ACF_FILE) $(OTHER_HEADERS)
    @rm -f .idl_run
    idl -cc_cmd '$(CC) -c $(DEBUGFLAG) $(CFLAGS)' $(INTERFACE).idl
    @touch .idl_run

parser_outputs:      $(PARSER_OUTPUTS)
idl_outputs:         $(IDL_OUTPUTS)

.C.o:
    $(CXX) -c $(CXXFLAGS) $<

.c.o:
    $(CC) -c $(CFLAGS) $<

.idl.h:
    idl -client none -server none $<

#####

clean:
    rm -f client server *.o *~ *.bak

clean-parser:
    rm -f $(PARSER_OUTPUTS)

clean-idl:
    rm -f $(IDL_OUTPUTS) $(OTHER_HEADERS)

clean-all:    clean clean-parser clean-idl

$(INTERFACE)-client.o:    $(INTERFACE).h $(IDL_OUTPUTS) $(OTHER_HEADERS)
$(INTERFACE)-server.o:   $(INTERFACE).h $(IDL_OUTPUTS) $(OTHER_HEADERS)
$(INTERFACE)-data.o:     $(INTERFACE).h $(IDL_OUTPUTS) $(OTHER_HEADERS)

.SUFFIXES:.idl

# MYNAH/DCE header files
HEADERS = $(TCLDCEINC)/xmyDceAnomaly.h $(TCLDCEINC)/xmyDceAsync.h \
$(TCLDCEINC)/xmyDceBinding.h $(TCLDCEINC)/xmyDceBindingCollection.h \
$(TCLDCEINC)/xmyDceBool.h $(TCLDCEINC)/xmyDceBuiltin.h \
$(TCLDCEINC)/xmyDceRt.h $(TCLDCEINC)/xmyDceRtInterface.h \
$(TCLDCEINC)/xmyDceRtInterfaceData.h $(TCLDCEINC)/xmyDceRtMethod.h \
$(TCLDCEINC)/xmyDceRtMethodTable.h $(TCLDCEINC)/xmyDceServer.h \
$(TCLDCEINC)/xmyDceRtServerMain.h $(TCLDCEINC)/xmyDceStatus.h \
$(TCLDCEINC)/xmyDceString.h $(TCLDCEINC)/xmyDceSupport.h \
$(TCLDCEINC)/xmyDceTclHandle.h $(TCLDCEINC)/xmyDceType.h \
$(TCLDCEINC)/xmyDceTcl.h $(TCLDCEINC)/xmyDceTclX.h \
$(TCLDCEINC)/tcl.h $(TCLDCEINC)/tclExtend.h

$(CLIENT_OBJS): $(HEADERS)
$(SERVER_OBJS): $(HEADERS)

```

C.2 Manual Steps

The MYNAHSystem delivers two components: the code generator (**xmyDceParser**) and the runtime library (**libxmyDce.a** and header files).

The **idl** tool must first be run on the interface files. This produces **interface_sstub.o**, **interface_cstub.o**, and **interface.h**. This tool is a standard DCE tool, and is available with the DCE development environment.

The **xmyDceParser** tool must then be run on the main interface file. This produces **interface-server.C**, **interface-server-main.C**, **interface-server-AppInit.c**, **interface-client.C**, **interface-client-main.C**, **interface-client-AppInit.c**, **interface-data.C**, **interface-data.h**, **interface-client-stubs.tcl**, and **interface-client-stubs.tcl**.

To build the emulated server, **interface-server.C**, **interface-server-main.C**, **interface-server-AppInit.c**, **interface-data.C** need to be compiled and linked with **interface_sstub.o** and the following libraries: **xmyDce**, **rwtool_mt**, **tclx**, **tcl**, **m**, **dce**, **nsl**, **socket**, **thread**, and **vx** (**vx** is only required if using TELEXEL trace).

To build the emulated client, **interface-client.C**, **interface-client-main.C**, **interface-client-AppInit.c**, **interface-data.C** need to be compiled and linked with **interface_cstub.o** and the following libraries: **xmyDce**, **rwtool_mt**, **tclx**, **tcl**, **m**, **dce**, **nsl**, **socket**, **thread**, and **vx** (**vx** is only required if using TELEXEL trace).

Appendix D: QA Partner Configuration Options

In addition to the entries listed in Section 3.3.2.4.1, many QA Partner configuration options may be added to the *xmyConfig* file's **GuiTool_qap** entry. This appendix lists the available options.

See the QA Partner documentation for the use of these options.

The tag for each option is simply the name of the option (ie, OPT_WINDOW_TIMEOUT).

The list of options are

XmScale

OPT_WINDOW_TIMEOUT
OPT_WINDOW_SIZE_TOLERANCE
OPT_WINDOW_RETRY
OPT_WINDOW_MOVE_TOLERANCE
OPT_VERIFY_UNIQUE
OPT_VERIFY_EXPOSED
OPT_VERIFY_ENABLED
OPT_VERIFY_CTRLTYPE
OPT_VERIFY_COORD
OPT_VERIFY_ACTIVE
OPT_TRIM_ITEM_SPACE
OPT_REL1_CLASS_LIBRARY
OPT_RADIO_LIST
OPT_MOUSE_DELAY
OPT_MENU_PICK_BEFORE_GET
OPT_MATCH_INVOKE_POPUP
OPT_MATCH_ITEM_CASE
OPT_KEYBOARD_DELAY
OPT_CLOSE_WINDOW_MENUS
OPT_CLOSE_WINDOW_BUTTONS
OPT_CLOSE_DIALOG_KEYS
OPT_CLOSE_CONFIRM_BUTTONS
OPT_COMPATIBLE_TAGS
OPT_BITMAP_PIXEL_TOLERANCE
OPT_BITMAP_MATCH_TIMEOUT
OPT_BITMAP_MATCH_INTERVAL
OPT_BITMAP_MATCH_COUNT

FlushResults

ResultsDir

ShowToolbar

IncludeRecordTag

OPT_INCLUDE_TAGS

OPT_USE_FILES
OPT_BITWISE_OPERATORS
OPT_TRACE
OPT_KEEP_COUNT
OPT_PROMPT_OPTS
OPT_PATH
OPT_ARGS
OPT_HOSTNAME

Appendix E: Architecture

The following sections contain basic information on aspects of the MYNAH System architecture.

E.1 Communications

The MYNAH System relies upon Telcordia's Telexel System for IPC services.

E.1.1 Platform Processes

Several Telexel processes must be up and running prior to an attempt to communicate. These processes are considered platform processes for the MYNAH System. They are required for *each* separate configuration of the MYNAH System. They are

- | | |
|---------------------|---|
| vxipcDir | One occurrence of this process must be running. It provides the directory name service for all other processes. |
| vxipcGateway | One occurrence of this process must be running <i>on each host</i> in the MYNAH System configuration. This process provides inter-machine communications. |

E.1.2 Channel Names

Once these platform processes are up and running, MYNAH processes can communicate with other processes through Telexel channel names. Each process in the system is given a unique channel name at bring up time. The unique channel name is derived from the logical name of the process that is stated in the configuration file. The channel name is composed of, at minimum

- the xmy prefix
- the type of process (e.g., SD)
- a unique identifier (e.g., any string of characters).

For example, **xmySD1** and **xmySD2** would be valid channel names for two SD processes.

The valid process types are:

- | | |
|----|-------------------|
| SD | Script Dispatcher |
| SE | Script Engine |
| BD | Boot Daemon |

TD	Trigger Daemon
GU	Graphical User Interface Process
SH	Command Line Tcl Shell
AD	Command Line Administration
DO	Command Line Do
OA	Operability Agent
OM	Operability Manager

Channel names are limited to a total of 30 characters.

If you are monitoring your system and you want to know how many MYNAH processes are running, you can use the Telexel **vxIpcMgr** and **vxIpcProcesses** tools. These produce such information as all active channel names. For example a simple pipe through **grep**, such as

```
vxIpcProcesses | grep xmyGU
```

will tell how many GUIs are running across your system.

E.2 Administrative Logging

All MYNAH process administrative and error logging is done through the Telexel Logger. This includes

- Start up information
- Shut down information
- Errors.

The Telexel Logger facility is available to all MYNAH processes. By using this facility, all MYNAH errors messages are located in one location. This location is known as the Error Log, which is *\$XMYHOME/syslog/adminLog*.

The Telexel Logger facility allows actions to be specified for particular messages (e.g. e-mail notification to the administrator).

The Telexel Logger provides a tool called **vxFilterLogFile** for filtering information in the log.

Appendix F: Auxiliary Terminfo File Information

This appendix information relating to the auxiliary *terminfo* file, which is specified using the **AuxTerminfo** configuration parameter (Section 3.3.2.1).

F.1 Escape Sequences Syntax

Table 11-2 lists the syntax of the basic escape sequences that can be used by the auxiliary *terminfo* file.

Table 11-2. Escape Sequences

Escape Sequence	Corresponding Character
\E	Escape
\n	Newline
\r	Return
\t	Tab
\b	Backspace
\f	Formfeed
\^	^ symbol
\\	\ symbol
\###	Character from octal code corresponding to ###, where ### is 000 through 777
^char	Control char (e.g., ^a -> ctrl-a)

See the **infocmp(1)** and **terminfo(4)** manpages for more detailed information.

F.1.1 Terminal Capabilities

Table 11-3 lists the only terminal capabilities that are supported by the MYNAH System TermAsync package.

Table 11-3. Terminal Capabilities

am	cup	home	rmir
blink	cupC	ich	rmso
bel	cupR	ich1	rmul
bold	cuu	ignore	sc
clear	cuu1	il	sgr0
cols	dch	il1	smacs
cnorm	dch1	ind	smir
csr	dl	is2	smso
cub	dl1	it	smul
cub1	ed	lines	tbc
cud	el	rc	lookup
cud1	el1	rev	
cuf	ht	ri	
cuf1	hts	rmacs	

F.1.2 Key Capabilities

Table 11-4 lists the key capabilities that let you modify what escape sequence is sent for special keys (e.g., F1, F2, ARROW_UP, ARROW_DOWN, ...).

Table 11-4. Key Capabilities

kbs	kf12	kf29	kf46
kcuu1	kf13	kf30	kf47
kcud1	kf14	kf31	kf48
kcuf1	kf15	kf32	kf49
kcub1	kf16	kf33	kf50
kent	kf17	kf34	kf51
kf10	kf18	kf35	kf52
kf1	kf19	kf36	kf53
kf2	kf20	kf37	kf54
kf3	kf21	kf38	kf55
kf4	kf22	kf39	kf56
kf5	kf23	kf40	kf57
kf6	kf24	kf41	kf58
kf7	kf25	kf42	kf59
kf8	kf26	kf43	kf60
kf9	kf27	kf44	
kf11	kf28	kf45	

Glossary

A

AID Key — Any 3270 special program key which causes the current screen to be sent to the 3270 SUT, and causes the SUT to transmit screen data back to the client.

Aggregate — A term used to identify a portion of a Flexible Computer Interface Format (FCIF) message. An aggregate contains zero or more tag-value pairs and is contained in an FCIF section.

App-to-App — Package that allows a user to send, receive, and analyze messages to and from a SUT over an application-to-application interface or a binary synchronous printer interface.

Array — A collection of associated variable elements.

Asynchronous Terminal Interface — An interface to an operating system or application that sends and receives data in arbitrarily-sized blocks at arbitrary times.

Attributes — **1.** The values defining the characteristics of a class or a class's connection, e.g., blinking, highlighted. **2.** The sub-commands used to specify or return attribute values. **3.** A category of methods and attributes that are used to find information about the SUT's configuration characteristics.

B

Background Execution

Environment — The combination of the Script Dispatchers, the Script Engine Groups, and the associated SEs.

Background Script Engine — SE process that communicates over an channel to a controlling process.

BD — See Boot Daemon

BEE — See Background Execution Environment

Binary Synchronous Communication — An IBM communications protocol that provides access to a 3270 data stream.

Boot Daemon — A platform process required by an SD that manages the SEs running on its machine.

BSC — See Binary Synchronous Communication

BSE — See Background Script Engine

C

Character Position — The manner in which screen positions are referred to. The screen can be viewed as one long string, where the indices of that string map to a position on the screen. For instance, the first value of the string has a character position of 1, which would have a corresponding row/column value of { 1 1}. The maximum character position, or the last position on the screen, varies from model type to model type, as different model types have different screen sizes. Model 2's maximum character

position is (24 x 80) 1920, while the bigger Model 5 has a maximum character position of (27 x 132) 3564.

Child Script — A Tcl script that is submitted for execution by a parent script

Class — A specific area or category of functionality.

Class Command — A command that gives you control over a class or category of functions.

Clear Tag-value Database — An ASCII file containing two columns separated by spaces or tabs. The first column contains the tag, and the second column contains the values.

CLUI — See Command Line User Interface

Command Line Script Engine — A MYNAH process that accepts Tcl commands from stdin and produces results on stdout. The Command Line Script Engine (CSE) does not interface with the MYNAH System database, but does, however, open a channel.

Command Line User Interface — A set of commands you can use to perform operations from the UNIX™ command line. The CLUI was designed for experienced users who prefer to use the UNIX command line rather than the GUI

Concatenate — To put two items together, end to end. For example, if you concatenated the strings "uvw" and "xyz", you get "uvwxyz". If you concatenate two files, the new file contains the contents of both files, presented sequentially.

Concurrency Group — The set of all scripts that run on one Script Dispatcher

Config file — The MYNAH Configuration File (named *xmyConfig*) that resides in the directory *\$XMYHOME/config*.

CSE — See Command Line Script Engine

D

des — A UNIX command to encrypt or decrypt data using the Data Encryption Standard.

Domain — A type of interface provided to the System Under Test. Examples of domains are the asynchronous terminal interface of an application, the application-to-application interface of an application, and the synchronous printer interface of an application.

Domain Connection — Any specific input/output interface to a SUT.

E

EAB — See Extended Attribute Bytes, used in 3270 to provide more information about a field, such as color attributes.

EHLAPI — See Emulator High Level Language Application Programmatic Interface.

Elements — Components of a list or array.

Embedded Script Engines — Script Engines that graphically display the screens associated with Term3270 or TermAsync Packages. Embedded Script Engines (ESEs) offer script

execution functionality through class methods. ESEs are not separate processes.

Emulator High Level Language Application Programmatic Interface — The IBM specification API for interacting with a 3270 host providing the essential functionality underneath the MYNAH 5.0 3270 Terminal domain.

Encrypted Database — A clear tag-value database that has been encrypted using `des`.

ESE — See Embedded Script Engine

Exception — Any event that can abort a script.

Extended Attribute Bytes — Used by the Term3270 Package to provide more information about a field, such as color attributes.

Extended Tcl — See TclX.

Extensions — Commands and procedures that expand Tcl's capabilities.

F

FCIF — Flexible Computer Interface Format. FCIF is a text format developed at Telcordia for communicating messages between processes.

FMM — See Flexible Message Manager (TraxWay-provided wrapper to Telexel IPC)

Flexible Message Manager — A TraxWay-provided wrapper to Telexel IPC. An inter-process communication module used by the MYNAH System that uses the Telexel

directory daemon underneath to do the actual IPC processing.

Focusing — Selecting a MYNAH GUI element and making it ready for you to enter information.

G

Global Array — An array of elements that are available to all domains.

GUI — Graphical User Interface.

H

Handle — A reference to an instance.

hllc() — The native EHLLAPI call. EHLLAPI makes use of one function, the `hllc()` function, which always takes four parameters. These four parameters determine what EHLLAPI function to execute, the input parameters to that function, and, afterwards, return the output of that function's execution, should there be any. The four parameters are commonly referred to as (and passed to the `hllc()` function in this order): Function Number, Data String, Data String Length or Buffer Size, and Presentation Space. This design refers to the specific EHLLAPI function simply as `hllc(function_number)`. For instance, the EHLLAPI function Connect Presentation Space corresponds to `hllc(1)`.

I

Icon — An X-Window that has been closed using a window manager function.

Iconified — The state of an X-Window after it has become an icon.

Instances — Connections made to SUTs using a class command.

IPC — Inter-process Communication Telexel IPC processes

J

Job Status Container — MYNAH GUI tool used to monitor the scripts that have submitted to the BEE.

L

List — An ordered collection of elements.

Log File — A file containing a record of activity for a software product.

M

Mask — A way to identify data that will be ignored during a comparison.

Methods — Sub-commands used to perform particular actions on instances you create in Tcl.

MYNAH System — An advanced software environment that can be used in all phases of software testing to exercise and analyze mainframe, minicomputer, and workstation applications. The MYNAH System can also be used for task automation and rapid application development.

O

OA — See Operability Agent

OM — See Operability Manager

Operability Agent — A MYNAH process that manages all MYNAH required processes on a host, communicating the start, stop and status requests to individual processes

and then communicating the reply back to the OM.

Operability Management — The MYNAH mechanism, consisting of the OA and OM, that lets the MYNAH Administrator start, stop, and get status of all of the MYNAH processes from any host.

Operability Manager — A set of commands used interact with an OA to manage all MYNAH processes. You can start or stop a process or you can determine if a process is running status

P

Parent Script — A Tcl script that submits other scripts for execution.

-position *position* — One of the ways of specifying screen location to a 3270 Tcl command. The position is a list of two integer values, row and column in that order. Example: -position { 1 1 }.

Presentation space — The 3270 screen that the EHLLAPI function call will affect and/or perform its action upon.

PRINTCOM — A program that interfaces to applications on a host computer over a binary synchronous communication line, receiving and capturing the printer output sent by the applications.

Process — An executable program that is active (running).

Prt3270 — MYNAH Package that lets a user simulate PRINTCOM processes.

R

Regression Testing — The testing of a previously verified application after changes have been made to the application.

Requesting Process — A process that sends a script-execution request to the SD (this does not include an SE sending a child-script-execution request to the SD).

RESDB — Remote Execution Server DataBase

Resource, X-Window System — A default value that can be changed by a user. Sets of resources are commonly stored in the `~/.Xdefaults` file (i.e., the file called ".Xdefaults" in your home directory).

Root Script — A script submitted to an SD via the GUI, CLUI, CSE, or an embedded SE, but not from one of the SEs that is controlled by the SD.

Runtime — A state in which a script is being executed.

Runtime Analysis — data Analysis that occurs during the execution of a test, and provides verification that the application being tested performed as expected. An example of runtime analysis is a comparison statement in a test script.

S

Screen Definition File — A file containing a tag name table. One file exists for each screen in a user's application. The file may contain other information in addition to the tag name table. The tag name table is

delimited in this file by "begin" and "end" statements.

Screen IDs File — A file containing the names of screens and other information to uniquely identify one screen from another.

Script — A file that contains one or more instructions to be performed by a domain.

Script Dispatcher — MYNAH process that provides scheduling and concurrency control for background execution of user scripts.

Script Engine — An extended Tcl interpreter that runs user scripts.

Script Engine Group — A logical set of BSEs controlled by an SD that all run on one host and run in the same mode. When a script is submitted to the BEE, it is submitted to run in a particular SE Group. It will run on one of the SEs in that group, but it doesn't matter which SE in the group it runs on.

Script Builder — A MYNAH GUI tool used to create script code by capturing interactions with a system, importing templates and procedures, and existing script code. A Standalone Script Builder can be run independently of the rest of the MYNAH GUI.

Script Object — A MYNAH GUI object used to create and track script code.

SD — See Script Dispatcher.

SE — See Script Engine.

SNA — See Systems Network Architecture.

Standalone Script Engine — A MYNAH process that accepts Tcl commands from **stdin** and produces results on **stdout**. The standalone SE does not interface with the MYNAH System database, but does, however, open a channel.

String — In Tcl: A set of characters that represents the current value of a variable. In some cases, strings show what will appear on the screen or in a printout.

SUT — System Under Test.

Symbol Table — User-supplied data associated with a script. Symbol tables contain symbol-value pairs; they can be read and modified by the script during execution.

Synchronous Terminal Interface — An interface to an operating system or application that sends and receives data in blocks of predefined size at regular intervals.

Systems Network Architecture — An IBM communications protocol that provides access to a 3270 data stream.

System Under Test — The system you wish to test or which contains the application you wish to automate.

T

TagDir — A directory containing Tag Name files.

Tag Table — A formatted table in a screen definition file, containing screen information for a single screen. For each user-identified screen field this table has a name for the field (called a tag name), the field's row and

column location, and the number of characters in the field.

Tag Name File — A file containing tag-value pairs, used for locating items on a synchronous screen.

Tag-value Pair — A pair of items, the first being a variable, the second being the value of that variable. Tag-value pairs reside in a symbol table.

Tags — User defined labels used by the Term3270 Package for locating items on a synchronous screen.

Tcl — Tool Command Language. An interpretive programming language, implemented as a library of C procedures, developed by John Ousterhout. Tcl is the basis for the MYNAH scripting language.

TclX — Extended Tcl, flavor of Tcl used by the MYNAH System under license from NeoSoft.

Term3270 — Package that performs 3270 synchronous terminal emulation, allowing users to build scripts that simulate an interactive work session with a SUT.

TermAsync — Package that performs asynchronous terminal emulation, allowing users to build scripts that simulate an interactive work session with a SUT.

Terminal Emulation — The use of software to emulate a type of hardware terminal (e.g., vt100, 3278).

TOPCOM — A software product that provides an interface to allow an application to establish and accept Transaction Oriented Protocol (TOP) sessions with a foreign partner, to send

and receive messages to and from the partner, and to terminate the sessions. TOPCOM uses X.25 or TCP/IP transport services to transport the application data messages and TOP protocol messages between the two partners.

TOP — Extension Package that lets a user simulate TOPCOM processes.

V

Variable — A user defined quantity that can assume a value.

Confidential — Restricted Access

MYNAH System Administration Guide
Glossary

Issue 5, August 1999
Release 5.4

Index

A

- Activity Logging, 10-1
 - SDs, 3-38
- Adding User Enum Values, 5-4
- Administrative Commands
 - CLUI, 1-16, A-1 to A-24
 - Tcl, 1-17, A-25
- Administrative Privileges
 - Granting
 - Using Person Objects, 7-3
 - Using xmyCmd setadm, 5-16, A-13
 - Removing
 - Using Person Objects, 7-4
 - Using xmyCmd unsetadm, 5-16, A-15
- Application-To-Application
 - Collector Processes
 - See CL
 - See TOP/PRT3270 Package
- Asynchronous Terminals
 - See TermAsync Package
- Authority Level, 7-3
 - Administrator, 7-3
 - Changing, 7-9
 - Inactive, 7-4
 - See Also Administrative Privileges
 - See Also Person Objects
 - User, 7-4
- Autostart Processes, 4-1, 4-5
 - Starting, 4-13
 - Stopping, 4-14
- Auxiliary Terminfo File, 3-15

B

- Background Execution Environment
 - See BEE
- Background SEs, 1-14
- BAIST, 2-8
 - Creating Version Directories, 2-2
 - Pre-installation Considerations, 2-8
 - Setting Installation Directory, 2-8

UNIX Shell, 2-8

- Batch Package, Required Software, 1-12
- BD, 1-2, 4-2
 - Definition, 1-2
 - Operability Commands, 3-46
 - Start, Stop, and Status Commands, 3-46
- BEE, 1-3 to 1-6
 - Changing SE Execution Modes, 1-5
 - Definition, 1-4
 - Use of Multiple SDs, 1-6
 - Using SE Groups, 1-5
 - Why Use, 1-4
- Boot Daemon
 - See BD
- Buffer Size, 3-15
- Building DCE executables, 11-1, C-1, C-5

C

- Checking the Queues on all Hosts, 5-11
- Checking the Queues on the Local Host, 5-10
- Child Script Events, 10-9
- CL, 1-3
 - Message Response Directory, 1-3
 - Operability Commands, 3-48
 - Start, Stop, and Status Commands, 3-48
 - Starting and Stopping, 3-48
- Clocks, Synchronizing, 3-57
- CLUI, 1-3
 - Administrative Commands, 1-16, A-1 to A-24
 - Help Messages, A-2
 - overview, 1-16
 - xmyCmd, A-3 to A-17
 - Adding Enumerated Values, A-4
 - Basic Options, A-3
 - Checking Queues, A-6
 - Checking Queues On Local Host, A-7
 - Decreasing the Number of SEs, A-10
 - Increasing the Number of SEs, A-11
 - SD Related Options, A-3

- SE Groups
 - Finding the Status of, A-12
 - Setting Concurrency Limits for a Specific User, A-16
 - Setting Default Concurrency Limits, A-8
 - Setting Queuing Priorities for New Users, A-9
 - Setting Queuing Priorities for Specific Users, A-17
 - Setting System Concurrency Limits, A-14
- xmyOM, A-18 to A-24
 - Displaying Configuration Processes, A-20
 - OA Reading the Configuration File, A-21
 - Shutting Down an OA, A-23
 - Start, Stop, Status Subcommands, A-23
 - Starting Autostart Processes, A-18
 - Stopping and Restarting Autostart Processes, A-22
 - Stopping Autostart Processes, A-19
- Command Line User Interface
 - See CLUI
- Command-line SEs, 1-14
- Compare Events, 10-9
- compares File, 10-6
- Comparisons
 - compares File, 10-6
- Concurrency
 - Definition, 1-2
 - Enforcing Limits, 3-39
 - Levels, 3-39
 - System Limit, A-14
- Concurrency Levels
 - Definition, 5-7
 - Setting Limits for a Specific User, 5-8, A-16
 - Setting Limits for New Users, 5-9, A-8
 - Setting System Limits, 5-7, A-14
- Configuration
 - Activity Logging, 3-38
 - Batch Package, 3-56
 - Configuring a Minimal MYNAH System, 2-44
 - Configuring MYNAH Database, 2-28
 - Configuring the Telexel System, 2-16 Database, Specifying Whether to Use, 3-11
 - General Entries, 3-11 to 3-13
 - GTSE, 3-27 to 3-28
 - GUI Test Tools, 3-25 to 3-30
 - See Also GTSE
 - See Also QA Partner
 - Message Collector, 3-23
 - Minimal System, 2-44
 - OA, 3-50
 - OM Port Number, 3-13
 - Port Numbers, 3-13, 3-16, 3-24, 3-26
 - QA Partner, 3-25 to 3-27
 - SDs, 3-38
 - Specifying Default, 3-11
 - SE Groups, 3-37
 - SEs, 3-31 to 3-36
 - Specialized Concerns, 3-56
 - Telexel, 2-16
 - Term3270 Package, 3-16 to 3-19
 - See Also Term3270 Package
 - TermAsync Package, 3-14 to 3-15
 - See Also TermAsync Package
 - TOP/PRT3270 Package, 3-19 to 3-22
 - See Also Message Collector
 - See Also TOP/PRT3270 Package
 - xmyConfig File
 - General Entries, 3-11 to 3-13
 - Package Entries, 3-14
 - GUI Test Tools, 3-25 to 3-30
 - GTSE, 3-27 to 3-28
 - QA Partner, 3-25 to 3-27
 - Message Collector, 3-23
 - Term3270 Package, 3-16 to 3-19
 - TermAsync Package, 3-14 to 3-15
 - TOP/PRT3270 Package, 3-19 to 3-22
 - ScreenIdentificationFile, 9-1, 9-8
 - See Also xmyConfig File
 - Syntax, 3-8 to 3-40
 - TagDir, 8-5
 - xmyConfigOP File
 - General Entries, 3-44

- See also xmyConfigOP File
 - Syntax, 3-43 to 3-51
 - Configuration Parameters
 - xmyConfig File
 - ActivityLogging, 3-38
 - AgentPort, 3-26
 - Agents, 3-28
 - AuxTerminfo, 3-15
 - Bin, 3-26
 - BufferSize, 3-15
 - CollectKeyCount, 3-19
 - Command, 3-26
 - CompareInvisibleFields, 3-17
 - ConversionMode, 3-21
 - Database, 3-11
 - Debug, 3-26, 3-28
 - DefaultEngineGroup, 3-38
 - DefaultSD, 3-11
 - Displays, 3-27
 - Engine, 3-37
 - EngineGroups, 3-38
 - ExecScript, 3-33
 - Handler, 3-24
 - Home, 3-26
 - Host, 3-16, 3-24, 3-37, 3-38
 - IniFile, 3-26
 - InitialWait, 3-18
 - InitialWaitExpect, 3-18
 - Key, 3-34
 - Lib, 3-26
 - LibraryPath, 3-33
 - LicenseFile, 3-26
 - ListenMode, 3-22
 - MatchProcedure, 3-22
 - MaxMsgs, 3-22
 - MaxSegmentLen, 3-22
 - MessageDirectory, 3-23
 - Mode, 3-32
 - Model, 3-16
 - MynahPort, 3-26
 - NumEngines, 3-37
 - OMPort, 3-13
 - OutputBackups, 3-34
 - OutputLevel, 3-35
 - OutputPath, 3-36
 - OutputRoot, 3-36
 - Package, 3-27
 - Port, 3-16, 3-24
 - ProcRepository, 3-34
 - Protocol, 3-21
 - ScreenIdentificationFile, 3-18
 - Shell, 3-15
 - ShowAttributes, 3-14, 3-17
 - StartupScript, 3-33
 - TableSize, 3-24
 - TagDir, 3-17
 - Term3270, 3-34
 - TermAsync, 3-36
 - Terminal, 3-14
 - Timeout, 3-14, 3-18, 3-21
 - TN3270E, 3-17
 - TopDefaultDTN, 3-22
 - TopDefaultPSN, 3-22
 - TopQueue, 3-21
 - TopRecvSession, 3-21
 - TopSendSession, 3-21
 - Type, 3-27
 - UnderlineUnprotectedFields, 3-19
 - UserConcurrencies, 3-39
 - UseVirtualDisplay, 3-28
 - VendorPath, 3-18
 - WelcomeNewUsers, 3-12
 - xmyConfigOP File
 - AutoStart, 3-44
 - Mynah, 3-44
 - Responsibility, 3-50
 - Start, 3-44
 - Status, 3-45
 - Stop, 3-45
 - ConnOnly Mode, 1-14, 1-15
 - Creating New MYNAH Configurations, 5-18
 - Creating the MYNAH System Database, 2-32
 - csh
 - Setting BAIST Installation Directory, 2-8
 - Use of xmyLogin File, 2-5
 - Customer Support, 1-19
- ## D
- Database
 - Dropping Oracle, 2-33
 - Installing Oracle, 2-20 to 2-35
 - MYNAH System
 - Configuring, 2-28

- Running Without a Database, 2-47, 3-11
- See Also* Oracle
- Specifying Whether to Use, 3-11
- Verifying, 2-34
- DCE
 - building executables for, 11-1, C-1, C-5
 - Executables, 11-1, C-1, C-5
- DCE Package
 - Required Software, 1-11
- Decoding licensing codes, 4-9
- Decreasing the Number of SEs, 5-15, A-10
- Displaying Operability configuration settings, 4-17

E

- Editing Person Object Attributes, 7-2
- Embedded SEs, 1-14
- Engine Groups
 - See* SE Groups
- entry_name, 3-8
- /etc/services, Changes, 2-13
- /etc/system, Changes, 2-13
- Exception (error) Events, 10-10
- Execution Modes, 3-32
 - ConnOnly, 1-14, 1-15
 - FullState, 1-14, 1-15
 - Stateless, 1-14, 1-15
- Extensions, 1-1

F

- File Archive Installation, 2-12, 2-15, 2-40
- FullState Mode, 1-14, 1-15
- Function Keys, Collect Count, 3-19

G

- Granting Administrative Privileges
 - Using Person Objects, 7-9
 - Using xmyCmd setadm, 5-16, A-13
- Graphical User Interface
 - See* GUI

- GTSE
 - Configuration
 - Create Virtual X11 Displays, 3-28
 - GUI Test Tool LogicalName, 3-27
 - Save Debugging Information, 3-28
 - Vendor Package Type, 3-27
 - X11 Displays, 3-27
 - GUI Test Script Engine, 3-25
- GUI, 1-3
 - Creating a Person Object, 7-2
 - Person Object, 7-1 to 7-11
- GUI Package
 - Required Software, 1-11
- GUI Test Script Engine
 - See* GTSE, 3-27

H

- Hardware Requirements, 1-17
- Hosts
 - Message Collector, 3-24
 - SDs, 3-38
 - SE Groups, 3-37
 - Term3270 Package, 3-16
 - TOP/PRT3270 Package, 3-24
- HP-UX, 1-18
 - Delivered Oracle Start-up Files, 4-22
 - Delivered Start-up Files, 4-21
 - Oracle Package, 2-21
 - Start-up Mechanism, 4-10

I

- Inactive Status, 7-4
- Increasing the Number of SEs, 5-14, A-11
- Installation
 - Assumptions and Recommendations, 2-2
 - BAIST considerations, 2-8
 - Changes to /etc/services, 2-13
 - Changes to /etc/system, 2-13
 - Configuring a Minimal System, 2-44
 - Environment Settings, 2-3
 - HP-UX
 - Automounting, 2-7
 - Installing License Keys, 2-43
 - Installing the MYNAH Software, 2-11

- madadmin logid, 2-7
- mynah Group ID, 2-7
- MYNAH System, 2-11 to 2-13
- Obtaining License Keys, 2-5
- Oracle, 2-20 to 2-35
- Recommended Directory Structure, 2-3
- Steps, 2-1
- Telexel, 2-14 to 2-19
- Term3270 Package Software, 2-36
- Verifying the MYNAH System, 2-46
- Integrating Other Tools, 3-33

K

- Keys, 3-34
- ksh
 - Setting BAIST Installation Directory, 2-8
 - Use of xmyProfile File, 2-4

L

- Language Events, 10-10
- License Keys
 - Installing, 2-43
 - for X-Direct TN3270, 2-37
 - Obtaining, 2-5
- License Server, 2-6
- Licenses
 - Installing Keys, 2-43
 - License Server, 2-6
 - Obtaining License Keys, 2-5
- Licensing, 4-7
 - Dashes in directory path or name, 2-7, 4-7
 - Decoding licensing codes, 4-9
 - Monitoring installed licenses, 4-8
 - Starting the license server, 4-7
 - Stopping the license server, 4-8
- Limits on Number of, 3-31
- Location Processing
 - Tag Name Files, 8-1
- LogicalNames, 3-8, 3-31
 - SE User Defined, 3-31

M

- Message Collector
 - Configuration
 - Hosts, 3-24
 - Message Directory, 3-23
 - Number of Received Messages Kept, 3-24
 - Port Number, 3-24
 - Protocol Handler, 3-24
- Message Directory, 3-23
- Message Response Directory, 1-3
- Message-Based Tracing, 10-1
 - methods, 1-16
- Model Type, 3-16
- Monitoring installed licenses, 4-8
- Monitoring MYNAH Log Files, 5-24
- Monitoring Telexel Processes, 5-20
- Monitoring the MYNAH System, 5-21
- MYNAH System
 - Creating the Database, 2-32
 - Database
 - Creating, 2-32
 - Dropping, 2-33
 - Disk space requirements
 - Disk space requirements, 1-8
 - Dropping the Database, 2-33
 - Installation, 2-11 to 2-13
 - Configuring a Minimal System, 2-44
 - File Archive, 2-12
 - Installing the Software, 2-11
 - Post-Installation Steps, 2-13
 - Pre-Installation Steps, 2-11
 - Verifying, 2-46
 - Integrating Testing Tools, 6-1
 - Networking, 1-18
 - Overview, 1-1
 - Processes
 - Automatically Starting, 3-44
 - Responsibility List, 3-50
 - Starting, 3-44
 - Commands, 3-46
 - Status of, Obtaining, 3-45
 - Commands, 3-46
 - Stopping, 3-45
 - Commands, 3-46

- Synchronizing Clocks, 3-57
 - Upgrading From a Previous Release, 2-13
 - Verifying, 2-46
- O**
- OA, 4-1
 - Configuration Parameter, 3-50
 - Definition, 4-4
 - Starting Telexel Directory Services, 3-45
 - Stopping an OA, 4-15
 - On a specified host, 4-15
 - On the local host, 4-15
 - Stopping and restarting a host, 4-16
 - xmyConfigOP Entry, 3-50
 - Obtaining a user's parameter value statistics, 5-6
 - Obtaining the status of processes, 4-16
 - OM, 4-1
 - Definition, 4-4
 - OM Port number, 4-5
 - Port Number, Specifying, 3-13
 - xmyOM Subcommands, 4-5
 - Operability, 1-16
 - Operability Agent
 - See OA
 - Operability Management, 4-1 to 4-23
 - Autostart Processes, 4-1, 4-5
 - Starting, 4-13
 - Stopping, 4-14
 - Basic Steps, 4-1
 - Design, 4-4
 - Displaying Operability configuration settings, 4-17
 - Licensing, 4-7
 - Decoding licensing codes, 4-9
 - Monitoring installed licenses, 4-8
 - Starting the license server, 4-7
 - Stopping the license server, 4-8
 - OA, 4-1
 - Obtaining the status of processes, 4-16
 - OM, 4-1
 - Overview, 4-2
 - See Also OA
 - See Also OM
 - Specifying if Process is a MYNAH Process, 3-44
 - Starting processes, 4-13
 - Stopping processes
 - Stopping a specific host, 4-14
 - Stopping an Autostart process, 4-14
 - Stopping an OA, 4-15
 - Stopping an OA and Autostart Processes
 - On a specified host, 4-15
 - On the local host, 4-15
 - Stopping and restarting a host, 4-16
 - xmyOM Subcommands, 4-5
 - autostart, 4-13
 - autostop, 4-14
 - query, 4-17
 - recycle, 4-16
 - shutdown, 4-15
 - start, 4-13
 - status, 4-16
 - stop, 4-14
 - .xmyStartup Command, 3-45
 - Operability Manager
 - See OM
 - Operating System, 1-18
 - HP-UX, 1-18
 - Solaris, 1-18
 - Operating System Requirements, 1-17
 - Options, 3-8
 - Oracle
 - Configuring MYNAH Database, 2-28
 - Database Disk Configuration, 2-23
 - Environment Variables, 2-22
 - Installation
 - Error Messages, 2-27
 - Installing the Software, 2-25
 - Verifying, 2-27
 - Pre-Installation Steps, 2-20
 - Required Packages, 2-21
 - Required Processes, 2-34
 - Verifying, 2-34
 - ORACLE_SID, 2-22, 2-29, 2-30, 2-31
 - Output
 - Activity Logging, 10-1
 - Child Script Events, 10-9
 - Compare Events, 10-9
 - Content of the Output Directory, 10-6

- Directory, short name, 10-3
 - Exception (error) Events, 10-10
 - Language Events, 10-10
 - Level, 3-35
 - Location of the Output Files, 10-3
 - Message-Based Tracing, 10-1
 - no backup, ramifications, 10-3
 - Number of Backups, 3-34
 - Output Directory
 - compares File, 10-6
 - output* File, 10-6
 - result* File, 10-6
 - See also compares File
 - See also *output* File
 - See also *result* File
 - See also SUTimage Files
 - stderr File, 10-6
 - stdout File, 10-6
 - SUTimage Files, 10-6
 - Output file, 10-8
 - output* File, 10-6
 - Recommended Background Engine
 - Output Levels, 10-19
 - Recommended Embedded Engine Output Levels, 10-18
 - Recommended Standalone Engine Output Levels, 10-18
 - result* File, 10-6
 - result file, 10-15
 - Root File System Path, 3-36
 - Script Events, 10-11
 - Script Output Directories, 3-36
 - Setting Output Level, 10-18
 - Summary Events, 10-12
 - SUT Timing (suttiming) Events, 10-13
 - Sutimage Events, 10-13
 - SUTimage Files, 10-6
 - User Events, 10-14
 - output* File, 10-6
 - Output Level, 3-35
 - OutputBackups, 10-3
- P**
- Parameters, 3-8
 - Person Objects, 7-1 to 7-12
 - Creating, 7-1
 - From the GUI, 7-2
 - Duplicate IDs, 7-5
 - Editing
 - Attributes, 7-2, 7-6
 - Authority Level, 7-9
 - Granting Administrative Privileges, 7-3
 - Information View, 7-6
 - Focusing a Keywords Box, 7-7
 - Limits per UNIX ID, 7-1
 - Properties View, 7-3
 - Removing Administrative Privileges, 7-4
 - Restrictions on Users, 7-1
 - Saving, 7-5
 - Port Numbers, 3-13, 3-16, 3-24, 3-26
 - Post-Installation Steps
 - MYNAH System, 2-13
 - Telexel, 2-16
 - TOPCOM, 2-40
 - Pre-Installation Steps
 - MYNAH System, 2-11
 - Oracle, 2-20
 - Telexel, 2-14
 - Privileges
 - Specifying, 7-3
 - Procedures
 - Making an Index of, 3-57
 - Procedures, Library, 3-34
 - Processed
 - Starting
 - Commands, 3-46
 - Processes
 - Automatically Starting, 3-44
 - Responsibility List, 3-50
 - Specifying if a MYNAH Process, 3-44
 - Starting, 3-44
 - Status of, Obtaining, 3-45
 - Commands, 3-46
 - Stopping, 3-45
 - Commands, 3-46
 - On local host, 4-15
 - ProcRepository, 3-34
 - Protocol, Specifying, 3-21

Q

- QA Partner
 - Agents, 3-28
 - Configuration
 - Agent Port, 3-26
 - Changes to Xdefaults File, 3-27
 - "Click-to-Type" Behavior, 3-27
 - Debugging Information, 3-26
 - Executables Directory, 3-26
 - Initialization Files, 3-26
 - Installation Home Directory, 3-26
 - Libraries, 3-26
 - License Environment Variable, 3-26
 - MYNAH Port, 3-26
 - QA Partner Specific Configuration
 - Options, D-1
 - Specifying Starting Command, 3-26
 - Options, D-1
- Queues
 - Checking
 - All Hosts, 5-11
 - Local Host, 5-10
 - Operability Configuration, 3-51
- Queuing Priority
 - Definition, 5-12
 - Setting for New Users, 5-12, A-9
 - Setting for Specific Users, 5-13, A-17

R

- Recommended Background Engine Output
 - Levels, 10-19
- Recommended Embedded Engine Output
 - Levels, 10-18
- Recommended Standalone Engine Output
 - Levels, 10-18
- Regular Expressions, 8-3, 9-1, 9-4
- Removing Administrative Privileges
 - Using Person Objects, 7-9
 - Using xmyCmd unsetadm, 5-16, A-15
- Removing local pip files, 4-11
- Removing Locks on Database Objects, 5-19
- Reports, see Output
- Required Software Packages, 1-9

Requirements

- Batch Package, 1-12
- DCE Package, 1-11
- GUI Package, 1-11
- Hardware, 1-17
- Minimal Configuration, 1-18
- Miscellaneous, 1-13
- Operating System, 1-17
- Required Software Packages, 1-9
- Single User Configuration, 1-18
- Term3270 Package, 1-9
- TermAsync Package, 1-10
- Third Party Software, 1-9
- TOP/PRT3270 Package, 1-10, 1-12
- result File, 10-15
- result* File, 10-6
- Running Concurrent Scripts, 1-2

S

- Screen Identification File, 3-18, 9-1 to 9-8
 - Commands
 - format-name-mask, 9-5
 - format-pattern, 9-3, 9-7
 - Contents requirements, 9-2
 - Enabling, 9-8
 - Example, 9-7
 - Processing, 9-6
 - Reasons for, 9-1
 - Regular Expressions, 9-1, 9-4
- Script Dispatcher
 - See SDs
- Script Engine
 - See SEs
- Script Events, 10-11
- Script Objects, 1-5
- Script Queues
 - Checking on all Hosts, 5-11
 - Checking on the Local Host, 5-10
 - Definition, 5-10
- Scripts
 - Concurrently Running, 1-2
 - Execution Modes, 1-15
 - Synchronizing, 1-2
 - Wrapper Basics, 6-1

- SD
 - Start, Stop, and Status Commands, 3-46
- SD Configuration
 - SE Groups
 - Setting Default, 3-38
 - Specifying, 3-38
- SDs, 1-2
 - Configuration
 - Activity Logging, 3-38
 - Concurrencies, Enforce Limits, 3-39
 - Host, 3-38
 - Specifying SE Groups, 3-38
 - Configuring, 3-38
 - Obtaining current global parameter values, 5-5
 - Operability Commands, 3-46
 - Recovering the State of, 3-47
 - Specifying Default, 3-11
 - Starting, 3-47
 - Stopping, 3-47
 - Using Multiple, 1-6
- SE Groups, 1-2, 1-5
 - Configuration
 - Host, 3-37
 - SE Configuration Entry, 3-37
 - Specifying an SE, 3-37
 - Specifying Number of SEs, 3-37
 - Configuring, 3-37
 - Finding the Status of, 5-17, A-12
 - Limits on Number of SEs, 3-31
 - Specifying for an SD, 3-38
- Sensitive Data. Providing Decryption Key, 3-34
- SEs, 1-2, 3-31, 5-14
 - Background, 1-14
 - Command-line, 1-14
 - Configuration
 - Decryption Key, 3-34
 - Execution Modes, 3-32
 - Execution Script, 3-33
 - Integrating Other Tools, 3-33
 - Library Paths, 3-33
 - Modes, 3-32
 - Non-MYNAH Scripts, 3-33
 - Output
 - Root File System Path, 3-36
 - Script Output Directories, 3-36
 - Output Backups, 3-34
 - Output Level, 3-35
 - Procedure Library, 3-34
 - Script Output Directories, 3-36
 - Startup Script, 3-33
 - Term3270 Package, 3-34, 3-36
 - Configuring, 3-31 to 3-36
 - Decreasing the Number of SEs, 5-15, A-10
 - Embedded, 1-14, 3-31
 - Execution Modes
 - ConnOnly, 1-14, 1-15
 - Effective of Changing, 1-5
 - FullState, 1-14, 1-15
 - Setting, 3-32
 - Stateless, 1-14, 1-15
 - Increasing the Number of SEs, 5-14, A-11
 - LogicalNames, 3-31
 - Setting the Number in an SE Group, 3-37
 - Specifying in an SE Group, 3-37
 - Standalone, 3-31
- Setting Administrative Privileges
 - Using the CLUI, 5-16
 - Using the GUI, 7-1 to 7-11
- Setting Concurrency Limits for a Specific User, 5-8, A-16
- Setting Concurrency Limits for New Users, 5-9, A-8
- Setting Output Level, 10-18
- Setting Queuing Priorities for New Users, 5-12, A-9
- Setting Queuing Priorities for Specific Users, 5-13, A-17
- Setting System Concurrency Limits, 5-7, A-14
- Setting the Number in an SE Group, 5-14
- sh, Setting BAIST Installation Directory, 2-8
- Software Packages
 - Required, 1-9
- Software Packages, Required for
 - 3230 Terminal Package, 1-9
 - 3270 Printer Package, 1-10
 - App-to-App Package, 1-12
 - Batch Package, 1-12
 - DCE Package, 1-11
 - GUI Package, 1-11

- Solaris, 1-18
 - Delivered Oracle Start-up Files, 4-21
 - Delivered Start-up Files, 4-20
 - Oracle Packages, 2-21
 - Start-up Mechanism, 4-10
 - Starting and Stopping MYNAH Processes
 - See Operability Management
 - Starting processes, 4-13
 - Starting the license server, 4-7
 - Startup Script, 3-33
 - Stateless Mode, 1-14, 1-15
 - Stopping an OA, 4-15
 - On a specified host, 4-15
 - On the local host, 4-15
 - Stopping and restarting a host, 4-16
 - Stopping processes
 - Stopping a specific host, 4-14
 - Stopping an Autostart process, 4-14
 - Stopping an OA, 4-15
 - Stopping an OA and Autostart Processes
 - On a specified host, 4-15
 - On the local host, 4-15
 - Stopping and restarting a host, 4-16
 - Stopping the license server, 4-8
 - Summary Events, 10-12
 - SUT, 1-1
 - Ability to Handle Concurrency, 1-6
 - Communications Between MYNAH System and SUT, 1-17
 - SE Connection Modes, 1-15
 - SUT Timing (suttiming) Events, 10-13
 - Sutimage Events, 10-13
 - SUTimage Files, 10-6
 - SUTimages File
 - Writing Attributes To, 3-14, 3-17
 - Symbol Table, 3-30
 - See Also QA Partner
 - Symbol Tables, 3-33
 - Synchronizing Clocks, 3-57
 - Synchronous Printers
 - See TOP/PRT3270 Package
 - Synchronous Terminals
 - See Term3270 Package
 - System Overview, 1-1
 - System Requirements, 1-8
 - System Under Test
 - See SUT
- ## T
- Tag Name Files, 8-1 to 8-5
 - Enabling, 8-5
 - Processing Undefined tag names, 8-5
 - Storing, 8-2
 - Tag Name Tables
 - Character limits, 8-4
 - Comments in a table, 8-3
 - Format, 8-2
 - Line Limits, 8-4
 - Using blank characters, 8-3
 - Tag Names
 - Definition, 8-1
 - Using, 8-2
 - tagdir, Specifying, 3-17
 - Tcl, 1-1
 - Administrative Commands, 1-17, A-25
 - ProcRepository, 3-34
 - TclX, 1-1
 - TD, 1-2
 - Operability Commands, 3-46
 - Start, Stop, and Status Commands, 3-46
 - Telexel
 - Configuring, 2-16
 - Installation, 2-14 to 2-19
 - File Archive, 2-15
 - Post-Installation Steps, 2-16
 - Pre-Installation Steps, 2-14
 - Specifying the Telexel Port, 2-16
 - Starting the Directory Service, 3-45
 - Verifying, 2-18
 - Telexel Processes
 - Monitoring, 5-20
 - Term3270 Package
 - Configuration
 - Count Function Keys, 3-19
 - Host, Specifying, 3-16
 - Initial Wait String, 3-18
 - Invisible Fields, 3-17
 - Model Type, 3-16
 - Port Number, 3-16

- Screen Identification File, 3-18
 - SUTimages File, 3-17
 - tagdir, Specifying, 3-17
 - Timeout, 3-18
 - TN3270E Mode, 3-17
 - Waiting for Initial Data, 3-18
 - Installing Required Software, 2-36
 - Location Processing
 - Screen Identification File, 9-1
 - Tag Name Files, 8-1
 - Port Number, 3-16
 - Required Software, 1-9
 - SE Configuration Entry, 3-34, 3-36
 - TermAsync Package
 - Configuration
 - Auxiliary Termino File, 3-15
 - Buffer Size, 3-15
 - SUTimages File, 3-14
 - Terminal Setting, 3-14
 - Timeout, 3-14
 - Required Software, 1-10
 - Terminal Setting, 3-14
 - Test Object Events
 - Output
 - Test Object Events, 10-14
 - Testing Tools
 - Integrating Other, 6-1
 - Third Party Software, 1-9
 - Timeout
 - Term3270 Package, 3-18
 - TermAsync Package, 3-14
 - TOP/PRT3270 Package, 3-21
 - TOPCOM, 1-12
 - Installation
 - Post-Installation Steps, 2-40
 - TOPCOM System
 - Installation
 - File Archive, 2-40
 - TOP/PRT3270 Package
 - Configuration
 - Conversion Mode, 3-21
 - Destination Transaction Name, 3-22
 - Listen Mode, 3-22
 - Match Procedure for Received Messages, 3-22
 - Maximum Appended Messages, 3-22
 - Presentation Services Name, 3-22
 - Protocol, 3-21
 - Send Session Number, 3-21
 - TCIS and TCIS2 Conversions Sizes, 3-22
 - Timeout, 3-21
 - TOP Queue, 3-21
 - Receive Session Number, 3-21
 - Required Software
 - Application-To-Application, 1-12
 - Synchronous Printer, 1-10
 - Trigger Daemon
 - See TD
 - TWO_TASK, 2-22, 2-29, 2-30, 2-31
- U**
- User Enum Values
 - Adding, 5-4
 - Definition, 5-2
 - Delivered values, 5-2
 - User Events, 10-14
- V**
- Verifying
 - Oracle, 2-27
 - Telexel, 2-18
 - vxIpcProcesses, 5-23
- W**
- Waiting, 3-18
 - Wrapper Script Basics, 6-1
- X**
- xmyConfig File
 - Configuration Parameters
 - General Entries
 - Database, 3-11
 - DefaultSD, 3-11

- OMPort, 3-13
- WelcomeNewUsers, 3-12
- GUI Test Script Engine Entries
 - Agents, 3-28
 - Debug, 3-28
 - Displays, 3-27
 - Package, 3-27
 - Type, 3-27
 - UseVirtualDisplay, 3-28
- MsgCollector Entries
 - Handler, 3-24
 - Host, 3-24
 - MessageDirectory, 3-23
 - Port, 3-24
 - TableSize, 3-24
- QA Partner Entries
 - AgentPort, 3-26
 - Bin, 3-26
 - Command, 3-26
 - Debug, 3-26
 - Home, 3-26
 - IniFile, 3-26
 - Lib, 3-26
 - LicenseFile, 3-26
 - MynahPort, 3-26
- Script Engine Entries
 - ExecScript, 3-33
 - Key, 3-34
 - LibraryPath, 3-33
 - Mode, 3-32
 - OutputBackups, 3-34
 - OutputLevel, 3-35
 - OutputPath, 3-36
 - OutputRoot, 3-36
 - ProcRepository, 3-34
 - StartupScript, 3-33
 - Term3270, 3-34
 - TermAsync, 3-36
- SD Entries
 - ActivityLogging, 3-38
 - DefaultEngineGroup, 3-38
 - EngineGroups, 3-38
 - Host, 3-38
 - UserConcurrencies, 3-39
- SE Group Entries
 - Engine, 3-37
 - Host, 3-37
 - NumEngines, 3-37
- Term3270 Entries
 - CollectKeyCount, 3-19
 - CompareInvisibleFields, 3-17
 - Host, 3-16
 - InitialWait, 3-18
 - InitialWaitExpect, 3-18
 - Model, 3-16
 - Port, 3-16
 - ScreenIdentificationFile, 3-18
 - ShowAttributes, 3-17
 - TagDir, 3-17
 - Timeout, 3-18
 - TN3270E, 3-17
 - UnderlineUnprotectedFields, 3-19
 - VendorPath, 3-18
- TermAsync Entries
 - AuxTerminfo, 3-15
 - BufferSize, 3-15
 - Shell, 3-15
 - ShowAttributes, 3-14
 - Terminal, 3-14
 - Timeout, 3-14
- TOP/PRT3270 Entries
 - ConversionMode, 3-21
 - ListenMode, 3-22
 - MatchProcedure, 3-22
 - MaxMsgs, 3-22
 - MaxSegmentLen, 3-22
 - Protocol, 3-21
 - Timeout, 3-21
 - TopDefaultDTN, 3-22
 - TopDefaultPSN, 3-22
 - TopQueue, 3-21
 - TopRecvSession, 3-21
 - TopSendSession, 3-21
- Definition, 3-1
- entry_name, 3-8
- Example, 3-40
- LogicalName, 3-8
- Options, 3-8
- Parameters, 3-8
- ScreenIdentificationFile, 9-1, 9-8
- SDs, 3-38
- SE Groups, 3-37
- SEs, 3-31 to 3-36
- Setting SE Execution Modes, 3-32
- Syntax, 3-8

- TagDir, 8-5
- Usage, 3-8
- xmyConfigOP File
 - Configuration Parameters
 - AutoStart, 3-44
 - Mynah, 3-44
 - Responsibility, 3-50
 - Start, 3-44
 - Status, 3-45
 - Stop, 3-45
 - Definition, 3-3
 - Example, 3-51
 - Syntax, 3-43
- xmyDceBuild
 - building DCE executables, 11-1
 - makefile, C-1
- \$XMYDIR, 2-4, 2-12
- \$XMYHOME, 2-4, 2-12
- xmyLogin, 2-3
- xmyProfile, 2-3
- xmyShutDown, 4-15
- xmyStartUp, 4-12
- xmyStartup, 3-45

Confidential — Restricted Access

MYNAH System Administration Guide
Index

Issue 5, August 1999
Release 5.4