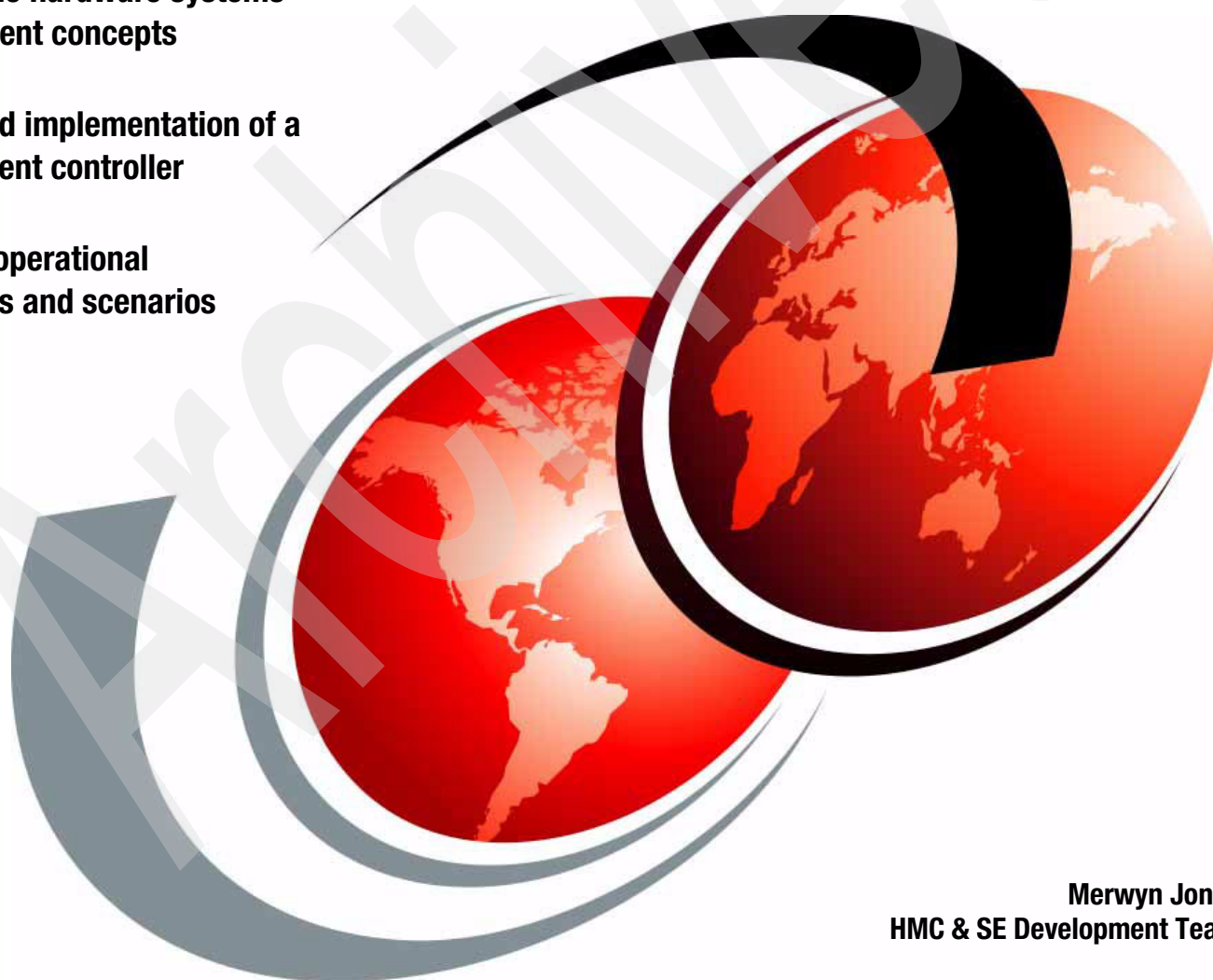


Introduction to the System z Hardware Management Console

Large scale hardware systems
management concepts

Design and implementation of a
management controller

Practical operational
techniques and scenarios



Merwyn Jones
HMC & SE Development Team

Redbooks



International Technical Support Organization

Introduction to the System z Hardware Management Console

February 2010

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

Archived

First Edition (February 2010)

© Copyright International Business Machines Corporation 2010. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi
Trademarks	xii
 Preface	 xiii
The team who wrote this book	xiii
Acknowledgements	xvi
Become a published author	xvi
Comments welcome	xvi
 Chapter 1. Introduction to System z Hardware	 1
1.1 General introduction: Mainframes in our midst	2
1.2 System z hardware architecture	3
1.2.1 Consolidation of mainframes	3
1.2.2 An overview of the early architectures	4
1.2.3 Early system design	5
1.2.4 Current architecture	7
1.3 The raised floor	7
1.4 Hardware management console	9
1.5 Frames and cages	9
1.6 Processor units	10
1.6.1 Multiprocessors	10
1.6.2 Processor types	11
1.7 Memory hierarchy	13
1.8 Networking the mainframe	14
1.9 Disk devices	14
1.9.1 Types of DASD	16
1.9.2 Basic shared DASD	17
1.10 I/O connectivity (channels)	18
1.10.1 Channel subsystem	18
1.11 System control and partitioning	20
1.11.1 Controlling the mainframe	20
1.11.2 Logically partitioning resources	21
1.12 Exercises and discussions	22
 Chapter 2. Systems management overview	 23
2.1 Introduction to systems management	23
2.1.1 System data	24
2.2 Hardware Management Console introduction	27
2.3 Support element introduction	30
2.4 HMC and SE system management disciplines	32
2.4.1 HMC and SE business management functions	32
2.4.2 HMC and SE configuration management functions	33
2.4.3 HMC and SE performance management functions	36
2.4.4 HMC and SE operations management functions	38
2.4.5 System z serviceability functions	45
2.5 Questions and discussions	47
 Chapter 3. User experience	 49
3.1 HMC user interface overview and background	50

3.2 HMC user interface design	51
3.2.1 Personas, user stories, and stakeholders	52
3.2.2 Initial design	52
3.3 Customer investment: User interface transition/migration.	53
3.3.1 Classic style	53
3.3.2 Tree style	55
3.3.3 How: user interface framework	59
3.4 Consistent user experience.	59
3.4.1 Common UI widgets	59
3.4.2 Panel and task consistency	60
3.5 Technologies.	63
3.5.1 WUI elements	64
3.5.2 Plug-ins	68
3.5.3 UI technology change tolerance	69
3.5.4 Server push (real-time status).	71
3.5.5 Disconnect (roaming users)	72
3.6 User defined resource relationships	72
3.7 Systems management launch in context	73
3.7.1 UI task integration: Launch-in-context.	74
3.8 Questions and discussions	75
Chapter 4. User management	77
4.1 HMC User Management overview	77
4.1.1 Typical scenario	78
4.2 HMC user management defaults and definitions	78
4.3 HMC user management definitions	79
4.4 Manage User Profiles and access	80
4.5 Adding a new user with an operator role	80
4.5.1 Customizing User Properties	82
4.5.2 Testing the new user.	83
4.6 Changing the user password	84
4.6.1 Password profiles	85
4.7 Customize User Controls task.	88
4.7.1 Creating, copying, modifying, or deleting Managed Resource Roles	89
4.7.2 Creating new Managed Resource Roles	89
4.7.3 Creating, copying, modifying, or deleting task roles	91
4.8 Managing users with data replication	92
4.9 User settings	93
4.10 Questions and discussion	96
Chapter 5. Hardware Management Console remote access	97
5.1 Cryptography	97
5.1.1 Encryption	98
5.1.2 Public-key cryptography	98
5.2 Remote access security	98
5.2.1 Diffie-Hellman key agreement protocol	98
5.2.2 Secure communication	99
5.2.3 Certificates	99
5.2.4 Firewalls	100
5.3 Secure Socket Layer (SSL)	100
5.3.1 Hypertext Transfer Protocol Secure	100
5.3.2 HMC certificate management	101
5.3.3 Allowing HMC user IDs Web access to an HMC	104

5.3.4	Accessing an HMC from another HMC	105
5.3.5	Using a Web browser to access an HMC	107
5.3.6	4.4.5 Logging into the HMC using the Web	108
5.4	HMC User authentication	109
5.4.1	4.5.1 LDAP support for user authentication	110
5.5	Questions and discussion	111
Chapter 6.	HMC networking	113
6.1	Overview of HMC networking	114
6.2	Integrating the HMC into your network	114
6.2.1	Identification	115
6.3	Network protocols overview	116
6.3.1	LAN adapters	117
6.3.2	Customize network settings and Name Services tab overview.	119
6.3.3	Routing	120
6.4	Firewall	121
6.4.1	The HMC Firewall	121
6.5	Questions and discussions	122
Chapter 7.	Remote Support Facility	125
7.1	RSF/call home overview	126
7.2	RSF security characteristics	126
7.3	Choosing a connectivity method for remote support.	126
7.3.1	Modem connectivity	127
7.4	Internet connectivity	128
7.4.1	Hardware management console direct Internet SSL connection	129
7.4.2	Using an indirect Internet connection with proxy server	129
7.4.3	Defining HMCs as call home servers	130
7.5	RSF setup	131
7.5.1	Enabling RSF on the HMC	131
7.6	Enabling a CPC to call home using this HMC.	137
7.7	Questions	140
Chapter 8.	HMC-based partitioning	141
8.1	Introduction to logical partitions.	141
8.2	How logical partitions are used	143
8.2.1	Defining logical partitions	144
8.2.2	Partition modes	144
8.2.3	Processor types	145
8.2.4	Shared and dedicated processors	145
8.2.5	Processing weights	146
8.3	LPAR advanced configuration options	146
8.3.1	Dynamically adding and removing logical processors and storage	146
8.3.2	Adjusting a partition's weight	146
8.3.3	LPAR group capacity limits	147
8.3.4	Using Workload Manager	147
8.3.5	Multiple Image Facility.	148
8.3.6	Multiple Channel Subsystem.	148
8.4	Logical partition planning	148
8.5	Creating logical partitions	149
8.5.1	Customizing and deleting activation profiles	149
8.5.2	Defining the general settings	150
8.5.3	Defining logical processors	151
8.5.4	Security	153

8.5.5 Storage	154
8.5.6 Options	156
8.5.7 Load	157
8.5.8 Crypto	158
8.6 Activating a logical partition	160
8.6.1 Initial program load	162
8.7 Deactivating the logical partition	162
8.8 Dynamic LPAR change controls	164
8.8.1 Change LPAR Controls panel	164
8.8.2 Logical Processor Add panel	165
8.8.3 Change LPAR Group Controls panels	166
8.8.4 Change LPAR I/O Priority Queuing panel	167
8.9 Questions and discussions	167
Chapter 9. Automation	169
9.1 Simple network management protocol	170
9.1.1 SNMP definitions	170
9.1.2 The SNMP architecture	171
9.1.3 Goals of the SNMP architecture	171
9.1.4 SNMP model	171
9.1.5 User datagram protocol	172
9.1.6 Asynchronous request/response protocol	173
9.1.7 SNMP agent	174
9.1.8 SNMP subagent	174
9.1.9 SNMP manager	174
9.1.10 Understanding Management Information Bases	175
9.1.11 Representation of management information	176
9.1.12 Identification of object instances	179
9.1.13 SNMP operations	181
9.1.14 Table traversal	181
9.2 SNMP on the HMC	182
9.2.1 Object identifier conventions	183
9.2.2 The SNMP Java API	185
9.3 Common Information Model	189
9.3.1 What is CIM	189
9.3.2 Managed Object Format	190
9.4 The CIM schema	195
9.4.1 Profiles	197
9.5 System z-specific schema	197
9.5.1 The Java CIM client API	201
9.6 Questions and discussions	205
Chapter 10. Problem analysis	207
10.1 Overview of problem analysis	208
10.2 RAS	208
10.2.1 Reliability	208
10.2.2 Availability	209
10.2.3 Serviceability	209
10.3 Autonomic computing	209
10.4 Problem analysis from start to finish	210
10.4.1 Problem analysis framework	210
10.4.2 Analysis phase	213
10.4.3 Analysis routines	215

10.5 First failure data capture	216
10.5.1 Data collection	217
10.5.2 The phases of first failure data capture	217
10.5.3 Trace	218
10.5.4 Notification	219
10.6 Problem Analysis initialization	220
10.7 Questions and discussions	221
Chapter 11. Change management	223
11.1 Overview	224
11.1.1 System z code maintenance and upgrade environment	224
11.1.2 Change management	225
11.1.3 System z code change request process	225
11.1.4 Engineering change stream overview	226
11.1.5 Fix release and availability	226
11.1.6 System z change management	226
11.1.7 Retrieve, activate, and install LIC updates	226
11.2 Alternate Support Elements	227
11.2.1 Mirroring	228
11.2.2 Switching	228
11.3 Strategy for applying fixes	228
11.3.1 Backing up the code on your platform	228
11.4 Strategy for updating driver	231
11.4.1 Updating drivers disruptively	231
11.4.2 Updating drivers concurrently	231
11.5 Underlying technology	233
11.5.1 Alternate Support Element	233
11.5.2 Making mirroring efficient	234
11.5.3 Primary Support Element outage tracker	234
11.6 Fixes	235
11.6.1 Dependencies between fixes	235
11.6.2 Hex patching	236
11.6.3 Cloning	238
11.6.4 Alerts	239
11.7 Concurrent driver upgrade	241
11.8 Problems and discussions	245
Chapter 12. Input / Output	247
12.1 Parallel channels	247
12.2 ESCON channels	248
12.3 FICON channels	249
12.4 Channel-to-channel	249
12.5 OSA channels	250
12.5.1 Internal OSA channels	251
12.6 Plugging cards in the machine	251
12.6.1 <i>I/O system control units and devices</i>	253
12.6.2 I/O system switches and directors	255
12.7 Logical devices and logical partitions	256
12.7.1 Multiple channel subsystem: removing the 256 channel limit	257
12.8 Defining the I/O to the hardware (IOCP)	257
12.8.1 Current system I/O specifications	260
12.9 Working with the channels	261
12.9.1 Advanced facilities	262

12.10 Conclusion	266
12.11 Questions and discussions	267
Chapter 13. System Activity Display	269
13.1 History	269
13.2 System z architecture as it relates to SAD	269
13.2.1 Storage Keys	270
13.2.2 Specialty engines	270
13.2.3 LPARs	270
13.2.4 Channels	271
13.2.5 High usage and low usage filtering	273
13.2.6 Usage cases	273
13.3 Profiles	273
13.4 Using SAD	282
13.4.1 Monitoring activity with a predefined profile	282
13.4.2 Monitoring a server	283
13.5 SAD implementation design	285
13.5.1 Polling loop	286
13.5.2 Calculation of statistics	286
13.5.3 SAD data flow	287
13.6 Creating a custom profile	288
13.7 Questions and discussions	292
Chapter 14. Capacity on demand	293
14.1 Overview of Capacity on Demand for System z	294
14.2 Fundamental components	295
14.2.1 Reconfigurable firmware	295
14.2.2 Unused hardware	295
14.2.3 Process controls	296
14.3 Capacity on demand records	296
14.3.1 Resource limits	297
14.3.2 Time limits	297
14.3.3 Consumption limits	297
14.3.4 Customer options for setting limits	298
14.4 Capacity on demand offerings	299
14.4.1 Model capacity	299
14.4.2 Software license charges	299
14.4.3 On/Off Capacity on Demand	300
14.4.4 Capacity Backup	300
14.4.5 Capacity for Planned Events	301
14.4.6 Customer Initiated Upgrade	301
14.4.7 Ordering records for capacity on demand offerings	302
14.5 Firmware process controls	302
14.5.1 Support element LIC	302
14.5.2 Machine LIC	303
14.6 Managing CoD records	304
14.6.1 Support Element user interface	304
14.6.2 Other record management options	307
14.6.3 Other considerations and activities	308
14.7 CoD record management scenarios	309
14.7.1 Peak workloads	309
14.7.2 Expense control	311
14.7.3 Disaster recovery	312

14.7.4 More scenarios and instructions	313
14.8 Questions and discussion	313
Chapter 15. Repair and Verify	315
15.1 Repair flow	316
15.2 Customer impact considerations	316
15.2.1 Redundant (N+1) power	317
15.2.2 Input/Out	317
15.2.3 Concurrent book repair (CBR)	318
15.3 Other options.	319
15.4 Sample sequence of panels for a repair	319
15.5 Questions and discussions	335
Appendix A. Chapter keys	337
A.1 Chapter 1	337
A.2 Chapter 2	337
A.3 Chapter 3	338
A.4 Chapter 4	339
A.5 Chapter 5	340
A.6 Chapter 6	340
A.7 Chapter 7	341
A.8 Chapter 8	341
A.9 Chapter 9	342
A.10 Chapter 10	343
A.11 Chapter 11	343
A.12 Chapter 12	344
A.13 Chapter 13	344
A.14 Chapter 14	345
A.15 Chapter 15	346
Related publications	347
IBM Redbooks	347
Other publications	347
Other materials.	348
How to get Redbooks.	349
Help from IBM	349

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®	Lotus®	S/390®
DS8000®	OS/390®	Sysplex Timer®
ECKD™	Parallel Sysplex®	System z10™
Enterprise Storage Server®	POWER®	System z9®
ESCON®	PR/SM™	System z®
eServer™	Processor Resource/Systems	Tivoli®
FICON®	Manager™	VTAM®
FlashCopy®	pSeries®	z/Architecture®
GDPS®	Redbooks®	z/OS®
HiperSockets™	Redpaper™	z/VM®
IBM®	Redbooks (logo)  ®	z/VSE™
IMS™	Resource Link™	z9®
iSeries®	RETAIN®	zSeries®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

In this textbook, we provide students with the background knowledge and skills that are necessary to begin using the basic functions and features of the System z® Hardware Management Console (HMC) and System z Support Elements (SE). This book is part of a series of textbooks that are designed to introduce students to mainframe concepts and to help prepare them for a career in large systems computing.

For optimal learning, we assume that the students are literate in personal computing and have some computer science or information systems background. Others who can benefit from this textbook include z/VM® and z/OS® professionals who want to expand their knowledge of other aspects of the mainframe computing environment.

After reading this textbook and working through the exercises, the student will have a basic understanding of the following topics:

- ▶ The System z Hardware concept and the history of the mainframe.
- ▶ HMC and SE components.
- ▶ The HMC and SE user interface and user management.
- ▶ System z First Failure Data Capture and Serviceability functions, including: Problem Analysis, Repair and Verify (R/V), Remote Support Facility (RSF), and Code Update.
- ▶ Monitoring the System z performance using the HMC System Activity Display (SAD).
- ▶ Implementing the networking capabilities of the HMC.
- ▶ Managing the System z Partitions (LPAR) and partition resource movement and sharing.
- ▶ Enabling and customizing the System z Customer Upgrade on Demand advanced functionality.

How each chapter is organized

Each chapter follows a common format:

- ▶ Objectives for the student.
- ▶ Topics that are relevant to the basics of z/VM computing.
- ▶ Questions or hands-on exercises to help students verify their understanding of the material.

The team who wrote this book

This book was produced by a team of specialists working in the hardware systems management organization in Endicott, NY.

Merwyn Jones is on an IBM® Fellowship assignment to Binghamton University, from the HMC organization, as a faculty member in both the Watson School of Engineering and the School of Management. He is also the Director of the Binghamton University Linux® Technology Center. Previously he was the Program Director for IBM eSystems hardware systems management at Endicott, where he had management responsibility for developing the HSM subsystem for the IBM largest enterprise servers, the System z and pSeries®. He has a Bachelor of Science in Electrical and Computer Engineering from Clarkson University and a Masters of Science in Computer Engineering from Syracuse University.

Steven Piersall is a Software Engineer with the System z SE/HMC Firmware Development Organization in Endicott, NY. He has a Bachelor's degree in Computer Science from Binghamton University. Steven has over 15 years of experience in SE/HMC Logical Partitioning development for zSeries®.

Thomas B. Mathias is a Senior Engineer in the IBM Systems Group. He received a Bachelor of Science degree in Electrical Engineering from Ohio State University and joined IBM in Endicott, New York, in 1984. He worked in System z hardware development and later in firmware development. Mr. Mathias was licensed as a Professional Engineer in the state of New York in 1992. He is co-inventor on four United States (U.S.) patents and one pending patent application.

Joe Gay is an HMC/SE Software Engineer with the System z SE/HMC Firmware Development Organization in Endicott, NY. He has more than 26 years of experience with the HMC, SE, System z, i, and p development. He has a Masters of Science degree in Electrical Engineering from Syracuse University. His area of expertise is System Serviceability.

Dawn Herrington is a Software Engineer with the System z SE/HMC Firmware Development Organization in Endicott, NY. She has been a Software Engineer with IBM for over 25 years. Her years of service gave her the opportunity to work in VM Development, Lotus® Development, the System z SE and the System z HMC. She has a Bachelor of Science in Computer Information Systems and a Masters degree in Software Development and Management from the Rochester Institute of Technology.

Forrest Schumacher is a Software Engineer with the System z SE/HMC Firmware Development Organization in Endicott, NY. He has 25 years of experience with HMC/SE development. He has a Masters of Science degree in Computer Science from SUNY-Binghamton. His area of expertise is channel advanced facilities.

John Eggleston is a Software Engineer with the System z SE/HMC Firmware Development Organization in Endicott, NY. He has over 27 years of experience as a code developer at IBM, including two years of experience with iSeries® and pSeries code update and over 10 years of experience with zSeries code update. John has a Bachelor of Science degree in Mathematics from Rensselaer Polytechnic Institute and a Master of Science degree in Computer and Information Science from Syracuse University. His area of expertise is Change Management.

Eve Berman is a Senior Software Engineer with the System z SE/HMC Firmware Development Organization in Endicott, NY. She has a Masters degree in Computer Science from the School of Advanced Technology in Binghamton University. Eve's areas of expertise include Remote Support technology, z/VM and z/OS development, and J2EE security. Eve currently designs and contributes to the development of the Remote Support Facility for the HMC.

Thomas Simkulet is a Software Developer with IBM System z Firmware Development in Endicott, New York. He has a Bachelor of Science degree in Mathematics from Pace University. He has 20 years of experience developing resources and applications for mainframes hardware support. He currently develops and supports the IBM Resource Link™ Web site, including its applications for Capacity on Demand for System z.

Richard Planutis is a Senior Software Engineer with the System z SE/HMC Firmware Development Organization in Endicott, NY. He has a Bachelors degree in Electrical Engineering from Pennsylvania State University and a Master of Science degree in Computer Engineering from Syracuse University. His area of expertise includes all aspects of hardware systems management and Common Criteria security evaluation of zSeries LPAR.

Brian Valentine is a Software Engineer with the System z SE/HMC Firmware Development Organization in Endicott, NY. Mr. Valentine graduated from Pennsylvania State University with a Bachelor of Science degree in Computer Science. He is a Senior Technical Staff Member with 26 years of experience with IBM, and he is currently the Lead Architect for the Hardware Management Console and Support Element Design and Development.

Francis Heaney is a Pre-Profession Programmer Co-op with the System z SE/HMC Firmware Development Organization in Endicott, NY. He has two years of experience with z/VM and System z hardware systems management. He has an Associate in Science (AS) in Computer Science from SUNY Jefferson and is currently pursuing his Bachelors at Binghamton University.

Eric Weinmann is a Senior Software Engineer with IBM in Endicott, New York, where he is a User Interface Developer for the IBM System z Hardware Management Console (HMC) and Support Element (SE). He has a Bachelors degree in Computer Science and Math from Clarkson University.

Justin Miller is a Software Engineer with IBM in Endicott, New York. He has eight years of experience as a Java™ Web Developer on IBM System z HMC and SE. He has a Bachelors of Science degree in Computer Science from Pennsylvania State University. He specializes in user interface development for the Hardware Management Console.

David Simpson is a Software Engineer with IBM in Endicott, New York. He has a Masters degree in Physics from Iowa State University. He has 25 years of experience with IBM, mostly as a hardware systems management Development Engineer. He developed expertise in many different System z HMC and SE areas over the years, most recently in hardware systems management using CIM.

Peter Kirkaldy is an Advisory Software Engineer with IBM in Endicott, New York with over 25 years of experience in the serviceability area. He has a Bachelors of Science and a Masters of Engineering degrees from Rensselaer Polytechnic Institute.

Brian Tolan is a Senior Software Engineer in Endicott, NY. He has a Bachelors of Science in Electrical Engineering from Columbia University. He has over 20 years of experience working in System z RAS. His area of expertise is System Serviceability.

Michael Clark is a Software Engineer in Endicott, NY. He has nine years of experience with System z Hardware Management Console development. He has a Bachelor of Science degree in Computer Science from the University of Pittsburgh. His areas of expertise are Java application development, Linux systems management, and Networking.

Brent Boisvert is a Senior Software Engineer with IBM in Endicott, New York. He has a Bachelors degree in Mathematics from the University of Vermont. He is responsible for the development of the System z Customer Initiated Upgrade tool, which manages and fulfills the IBM Capacity on Demand solutions for System z. He is also responsible for the IBM Resource Link site for System z.

Christine Smith is a Software Engineer with System z Firmware in Endicott, New York. Her expertise is in the information development area where she has more than 25 years of technical writing experience in both IBM software and hardware. For the past seven years she has developed technical information for the System z Hardware Management Console tasks and the corresponding Hardware Management Console Operations Guide. She has a Bachelor of Science degree in Information Processing from Mansfield University.

Bill Ogden provided the primary editing for this document. Bill is a retired Senior Technical Staff Member with the International Technical Support Organization in Poughkeepsie.

Acknowledgements

We acknowledge the good work of several interns from Binghamton University who helped prepare material for this document.

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks® to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review form at:

ibm.com/redbooks

- Send your comments in an E-mail to:

redbook@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400



Introduction to System z Hardware

In this chapter, we cover the following concepts:

- ▶ General introduction
- ▶ System z hardware architecture:
 - Consolidation of mainframes
- ▶ An overview of the early architectures:
 - Early system design
 - Current architecture
 - Hardware Management Console
- ▶ Frames and cages
- ▶ Processing units:
 - Multiprocessors
 - Processor types
- ▶ Memory hierarchy
- ▶ Networking the mainframe
- ▶ Disk devices:
 - Basic shared DASD
- ▶ I/O connectivity (channels)
- ▶ System control and partitioning:
 - Logically partitioning resources
- ▶ Exercises

This document deals with the IBM System z Hardware Management Console (HMC) and the System z Support Element (SE). As a new HMC/SE user, you must develop an understanding of the System z hardware platform. The HMC/SE is designed to install, manage, and initialize the mainframe hardware and its many sophisticated peripheral devices. Although much of the hardware that the HMC/SE manages has its roots in older

mainframe system designs, in this chapter, we introduce you to the concepts and current systems that are in production today.

Objectives

After completing this chapter, you will be able to:

- ▶ Discuss System z hardware
- ▶ Explain processing units and disk hardware
- ▶ Explain how mainframe hardware differs from personal computer system

1.1 General introduction: Mainframes in our midst

Mainframes tend to be hidden from the public eye. They do their jobs dependably and with almost total reliability. They are highly resistant to most forms of insidious abuse that afflict personal computers, such as e-mail-borne viruses, trojan horses, and the like. By performing stably, quietly, and with negligible downtime, mainframes set the standard by which all other computers can be judged. But at the same time, this lack of attention tends to allow them to fade into the background.

You would not know this from monitoring the general trends that computer professionals publicize, but mainframe computers play a central role in the daily operations of most of the world's largest corporations. While other forms of computing are also used in business in various capacities, the mainframe occupies a coveted place in today's environment, where e-business reigns. In banking, finance, health care, insurance, public utilities, government, and a multitude of other public and private enterprises, the mainframe computer continues to form the foundation of modern business.

Why has this one form of computing taken hold so strongly among the world's largest corporations? In this chapter, we look at some of the characteristics and design principles of mainframes that contributed to their success. In the remainder of this book, we then go into more detail about one of the areas involved in mainframe management.

Who uses mainframe computers

Just about everyone has used a mainframe computer at one point or another, whether they realize it or not, for example, if you ever used an automated teller machine (ATM) to interact with your bank account, you used a mainframe.

In fact, until the mid-1990s, mainframes provided the only acceptable way of handling the data processing requirements of a large business. These requirements were then (and are often now) based on large and complex batch jobs, such as payroll and general ledger processing.

The mainframe owes much of its popularity and longevity to its inherent reliability and stability, a result of careful and steady technological advances since IBM introduced System/360 in 1964. No other computer architecture can claim as much continuous improvement while maintaining compatibility with previous releases.

Because of these design strengths, the mainframe is often used by Information Technology (IT) organizations to host their important mission-critical applications. These applications typically include client order processing, financial transactions, production and inventory control, payroll, and many other types of work.

Many of today's busiest Web sites store their production databases on a mainframe host. New mainframe hardware and software are ideal for Web transactions because they are

designed to allow huge numbers of users and applications to rapidly and simultaneously access the same data without interfering with each other. This security, scalability, and reliability is critical to the efficient and secure operation of contemporary information processing.

Corporations use mainframes for applications that depend on scalability and reliability, for example, a banking institution could use a mainframe to host the database of its client accounts for which transactions can be submitted from any of thousands of ATM locations worldwide.

Businesses today rely on the mainframe to:

- ▶ Perform large-scale transaction processing (up to thousands of transactions per second)
- ▶ Support thousands of users and application programs concurrently accessing numerous resources
- ▶ Manage terabytes of information in databases
- ▶ Handle large-bandwidth communications

The roads of the information superhighway often lead to a mainframe.

1.2 System z hardware architecture

System z, as with all computing systems, is built on hardware components. Most of those components were introduced early in the mainframe era, and were developed over the years. As a user of the HMC/SE, you must interact with the hardware or speak in terms of the terminology that is prevalent in the mainframe world. In this chapter, we discuss the reasons why things are the way they are now and also supply you with the necessary understanding of the hardware to enable you to efficiently handle your day-to-day use of the HMC/SE.

1.2.1 Consolidation of mainframes

There are fewer mainframes in use today than there were 15 or 20 years ago. Why is this? In some cases, all of the applications were moved to other types of systems. However, in most cases, the reduced number is due to consolidation, where several lower-capacity mainframes are replaced with fewer higher-capacity systems.

There is a compelling reason for consolidation. Software (from many vendors) can be expensive and typically costs more than hardware. It is usually less expensive (and sometimes *much* less expensive) to replace multiple software licenses, for smaller machines, with one or two licenses for larger machines. Software license costs are often linked to the power of the system, but the pricing curves favor a small number of large machines.

Software license costs have become a dominant factor in the growth and direction of the mainframe industry. There are several nonlinear factors that make software pricing very difficult, such as the exponential growth of mainframe processing power, which has been problematic in recent years.

The power that is needed to run a traditional mainframe application (a batch job written in COBOL, for example) is unlike the needs of a modern mainframe running z/VM with many guest operating systems (which in turn might be executing complex code with graphical user interfaces or executing many Linux Virtual Servers running Java). The consolidation effect has produced very powerful mainframes.

Regardless of the speed of your mainframe, take the time to learn some of the internal hardware that is used in mainframes. It is vital that HMC/SE users and operators have an understanding of the hardware that they are running on. Some of the hardware is not even physically made anymore but still has relevance to the HMC/SE. In the remainder of this chapter, we illustrate the basic hardware concepts and, whenever possible, directly relate them to the use of the HMC/SE.

We provide a simplified overview of mainframe hardware systems, with emphasis on the processor, in this chapter. For more extensive information about mainframe hardware, there are numerous other resources that you can consult.

1.2.2 An overview of the early architectures

If you are wondering why a historical overview of early mainframe architecture is important, keep in mind that unlike personal computers—where technology is continuously made obsolete—mainframe computers (including the HMC) maintain backward compatibility. In this section, we examine the intricacies of interacting with mainframes in general.

The publication *Principles of Operation* provides detailed descriptions about the major facilities of z/Architecture®. You can find this and other IBM publications at the z/VM Internet Library Web site:

<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/zvmpdf/zvm53.html>

We start by explaining terminology that is associated with mainframe hardware. Being aware of various meanings of the terms systems, processors, CPUs, and so forth is important for your understanding of HMC/SE and mainframes in general.

In early mainframe computing, starting with the IBM S/360 (the ancestor of the current mainframes in production today), a system had a single processor, which was also known as the *central processing unit* (CPU). This is the same term that is used to describe the processor in your mobile computer or desktop personal computer.

In those days, the terms system, processor, and CPU were used interchangeably. Today, systems are available with more than one processor, and the terminology that is used to refer to such components has evolved. All current mainframes have more than one processor, and some of those processors are specialized for specific purposes, for example, I/O handling, relational databases, and Java execution. Figure 1-1 shows some of the terminology that is used today.

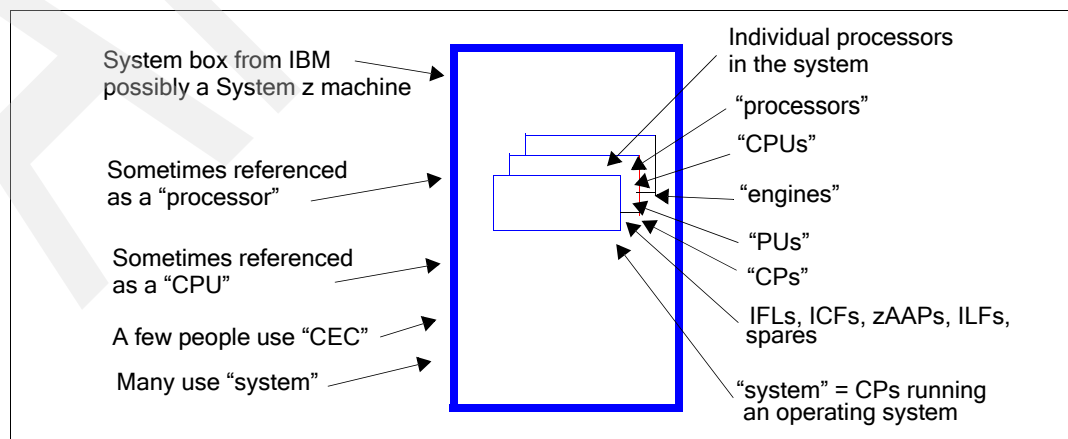


Figure 1-1 Terminology overlap

System Programmers use the IBM terms *Central Processor Complex* (CPC) or *Central Electronics Complex* (CEC) to refer to the physical mainframe.

In this text, we use the term CPC and CEC interchangeably to refer to the physical collection of hardware that includes main storage (memory), one or more central processors, timers, channels, and the numerous other hardware components that could be contained inside a single mainframe chassis.

Although the terms *processor* and *CPU* can refer to either the complete system or to one of the processors (CPUs) within the system, we recommend that you use the term CPU to refer to an actual processor unit inside of the system, and use the term CPC or CEC to discuss the entire physical machine. As illustrated in Figure 1-1 on page 4, there is much terminology in this area that can be confusing until you are more familiar with it. In most cases, the context of a discussion determines which of these overlapping terms are relevant for that context.

1.2.3 Early system design

The central processor contains the processors, memory, control circuits, and interfaces for channels. A *channel* provides a path between I/O devices and memory. Early systems had up to 16 channels. In contrast, modern mainframes can have many channels.

Channels connect to control units. A *control unit* contains the logic to work with a particular type of I/O device. A control unit for a printer, for example, has much different internal circuitry and logic than a control unit for a tape drive. Some control units can have multiple channel connections that provide multiple *paths* to the control unit and its devices.

Control units connect to *devices*, such as disk drives, tape drives, communication interfaces, and so on. The division of circuitry and logic between a control unit and its devices is not defined, but it is usually more economical to place most of the circuitry in the control unit.

Figure 1-2 on page 6 shows a conceptual diagram of a mainframe system. Current systems are not connected, as shown in Figure 1-2 on page 6; however, this diagram helps to explain the background terminology that permeates mainframe discussions. We provide this short history lesson to help you understand the origin to today's terminology.

Modern mainframes are descended from the architecture presented in Figure 1-2 on page 6 and retain some of the design shown. Because modern mainframes retain backward compatibility with their predecessors, this early design warrants further examination.

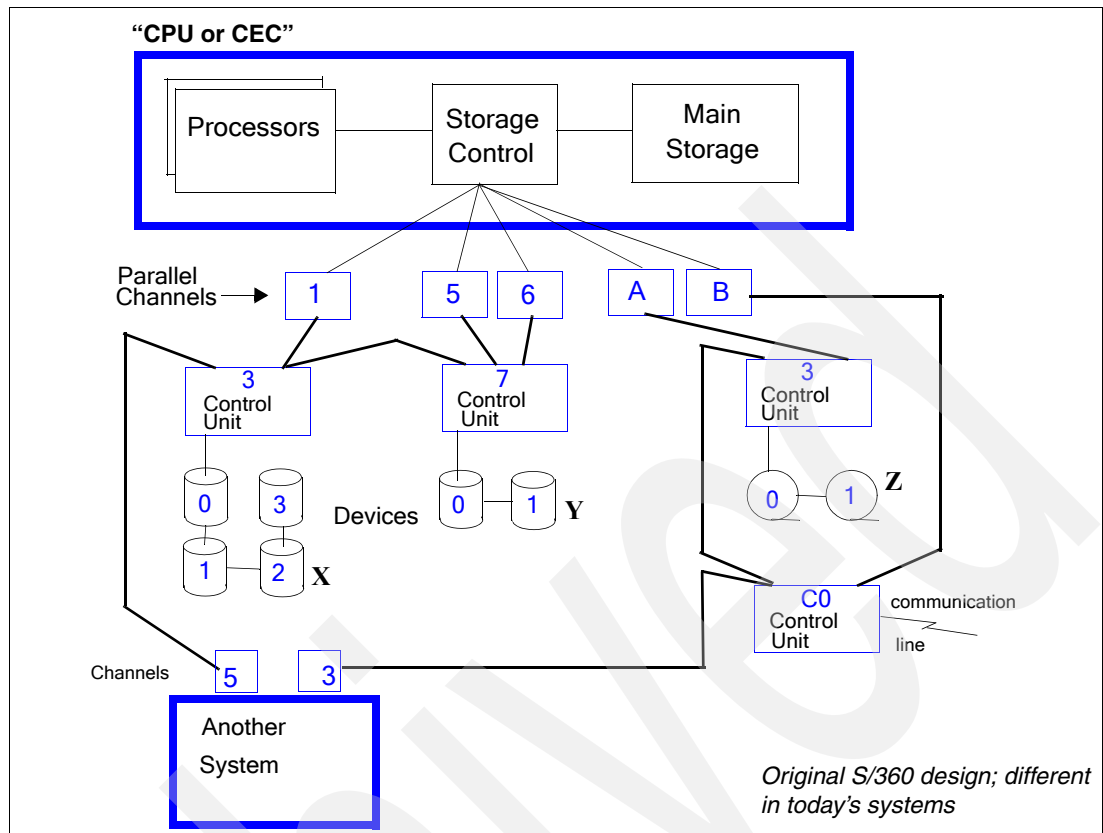


Figure 1-2 Conceptual older mainframe configuration

The control unit blocks in Figure 1-2 contain the logic circuits for controlling the physical tape and disk devices. Notice that a control unit can connect to multiple devices, and the maximum number of devices depends on the type of control unit.

Each channel, control unit, and device has a number. Used together these create device addresses, which are expressed as hexadecimal numbers. The disk drive that is marked with an X in Figure 1-2 has the address 132, as shown in Figure 1-3.

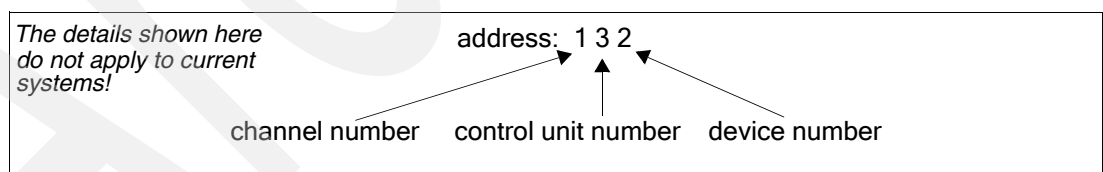


Figure 1-3 Device address

In Figure 1-2, the disk drive that is marked with a Y can be addressed as 171, 571, or 671 because it is connected through three channels. By convention, the device is known by its lowest address (171), but the operating system can use all three addresses to access the disk drive.

Multiple paths to a device are useful for performance and for availability. In the conceptual system, represented in Figure 1-2, when an application wants to access disk 171, the system first tries channel 1. If it is busy (or not available), the system tries channel 5 and so forth.

Figure 1-2 on page 6 also contains another S/360 system with two channels that are connected to control units used by the first system. This sharing of I/O devices is common in

all mainframe installations. Tape drive Z is address A31 for the first system but is address 331 for the second system.

Sharing devices, especially disk drives, is not a simple topic. There are hardware and software techniques that the operating system uses to control exposures, such as updating the same disk data at the same time from two independent systems.

Note: For more information about the development of IBM mainframes since 1964, refer to the following Web site:

http://www-03.ibm.com/history/exhibits/mainframe/mainframe_basinfo.html

Current mainframes are not used exactly as shown in Figure 1-2 on page 6. The differences include the following areas:

- ▶ The parallel channels in Figure 1-2 on page 6 are no longer available on the newest mainframes and are slowly being displaced on older systems. They are replaced with newer, more efficient types of channels called *Enterprise Systems CONnection* (ESCON®) and *Fiber CONnection* (FICON®) channels. We examine each of these technologies in later sections of this chapter.
- ▶ Current mainframes have more than 16 channels and use two hexadecimal digits as the channel portion of an address.
- ▶ Channels are generally known as *channel path identifiers* (CHPIDs) or *physical channel identifiers* (PCHIDs) on later systems, although the term channel is also correct. The channels are all integrated in the main processor box:
 - A CHPID is a value that is assigned to each channel path of the system that uniquely identifies that path.
 - A PCHID reflects the physical location of a channel-type interface. A PCHID number is based on the device location and the port number.

The device address that the software sees is more correctly known as a device number (although the term address is still widely used) and is only indirectly related to the control unit and device addresses.

1.2.4 Current architecture

Current CPC designs are considerably more complex than the early S/360 design. This complexity includes many areas, such as:

- ▶ I/O connectivity and configuration

Note: For more information about mainframe I/O connectivity and configuration, refer to *System z Connectivity Handbook*, SG24-5444.

- ▶ I/O operation
- ▶ Partitioning of the system

1.3 The raised floor

Early mainframes (even before the S/360) were physically large and sometimes spoken of as *big iron*. There were often many units, racks, and frames that are involved with large numbers

of large cables between the various units. These units consumed an abundance of electrical power and released considerable heat. These two factors (cables and heat) led to the use of *raised floors* to house the computers. The cables were routed under the floor, and cool air was forced under the floor and up through the system boxes through holes in the raised floor. Typically, the same holes were used for cables and air movement.

The raised floor was usually made of steel panels (often two feet square) set on posts or on a grid work. It was typically 12 to 18 inches above the “real” floor. These raised floors are strong because mainframe boxes tend to be heavy. Floor panels are frequently removed (with a puller) to reroute cables. Various types of chilled-air handling equipment were usually scattered around the room.¹ The machines (with lots of ventilation fans) and the air handling equipment tend to make a lot of noise, and a typical raised floor or machine room is not a good place for quiet desk work.

At various times in mainframe evolution there were more environmental aspects of the raised floor areas, which include motor generators, fire-suppression systems involving special gases, battery-operated power supplies, quick-start diesel generators to take over from the batteries, floor-wide emergency power shutdown controls, multi-phase power, and so on. Some of these more exotic items were typically not on the raised floor itself but were nearby.

The uniqueness of raised floor construction usually meant it was confined to specific rooms (perhaps large rooms), which made access control easier. General users, even application developers, were not allowed into these areas.

Complete mainframe systems have varied in size over the years, reaching a peak in the 1990-1995 era when a large system (with numerous disk drives) might occupy 10,000 square feet (1000 m²) or more. Some installations had multiple mainframes in these spaces that shared the I/O farm among all of the processors.

Modern mainframes are much smaller but are still normally placed in raised floor locations. There are fewer and smaller cables (mostly fiber optic now), but there are still lots of them. Less electrical power is used, which generates less heat, but raised-floor style air conditioning is still needed for the larger units.

This evolution contributed to the vocabulary of mainframes. You should expect to work with these terms:

- ▶ Raised floor: Machine room or the floor
- ▶ I/O farm or disk farm: Once a description of a physically large raised-floor area, but now more of a conceptual term
- ▶ Floor panel, tile or half-tile, or quarter-tile: Refers to the size of a hole in the tile
- ▶ To pull cables: Remove or rearrange a group of cables under the raised floor
- ▶ Puller or tile puller or floor puller: A hand-held tool used to lift raised-floor tiles
- ▶ Emergency Power Off (EPO) switches
- ▶ Halon: A fire suppression gas
- ▶ Floor access: Having the right badge to obtain access to the raised floor area

The hardware management console that is the focus of this book can actually be multiple consoles in multiple locations. At least one of these (often considered the primary HMC) is almost always located on the raised floor.

¹ Some larger mainframes also used chilled water for cooling, which made an even more complex raised floor environment.

1.4 Hardware management console

As a user of the HMC/SE, you might be called upon to alter the hardware configuration of your system or to answer questions from support personnel regarding the system configuration.

Regardless of what hardware is in your mainframe, the hardware can be managed by using either the Support Elements that are directly attached to the server or the Hardware Management Console. The HMC is an IBM System z feature that provides the end-user interface to control and monitor the status of the system.

Working from a Hardware Management Console, an Operator, System Programmer, or IBM technical personnel can perform basic operations on System z servers. Some of the common capabilities of the HMC are:

- ▶ Load the System z hardware configuration.
- ▶ Load or reset a system image.
- ▶ Add and change the hardware configuration (most of them dynamically).
- ▶ Access the hardware logs.

All of these functions can be executed by using a Web interface in a secure environment. If you are reading this book, probably you have not yet used a Hardware Management Console, but as a concept it is important to understand because the HMC is the centralized location from which hardware management for the entire mainframe(s) can be performed. Changes made on the HMC can alter only your operating system or the entire mainframe.

1.5 Frames and cages

The current System z hardware layout is the result of the continuous evolution of the mainframe hardware architecture. Over the years new components were added to the design but the basic requirement was not changed. The current design is highly available and has redundancy for most of the parts.

The IBM z10 machines are built on frames, in which the various components are fixed.² (We refer to both z9® and z10 systems in this document because they are very similar in many external details.) Depending on the model number that is ordered, the frames might not be fully populated. Each frame can contain several *cages*. There are two types of cages:

- ▶ CEC cage

The CEC cage is the basic cage. It contains the processor units (PU), the physical memory, and the connectors to the other cages. On each System z machine there is only one CEC cage.

- ▶ I/O cage

The I/O cage contains the hardware that is necessary for interacting with System z external I/O devices. Each System z configuration has at least one I/O cage with a maximum of three I/O cages.

Memory and processors reside in what are termed *books*. The book concept was first introduced on the z990 machine a few years ago. A book contains processors, memory, and connection to the I/O cages. Books are located in the CEC cage. Each System z configuration has one to four books.

² The frame method of housing a system is not new and has been used for mainframes for many years. The details of the frames have changed over the years.

Note: The current maximum hardware configuration allows 54 processor units to be available to the operating systems. Physically there are up to 64 PUs installed on the machine. The 10 additional PUs are used for I/O workloads and as hot spares in case another PU malfunctions.

Figure 1-4 shows a z9 processor with the covers removed.

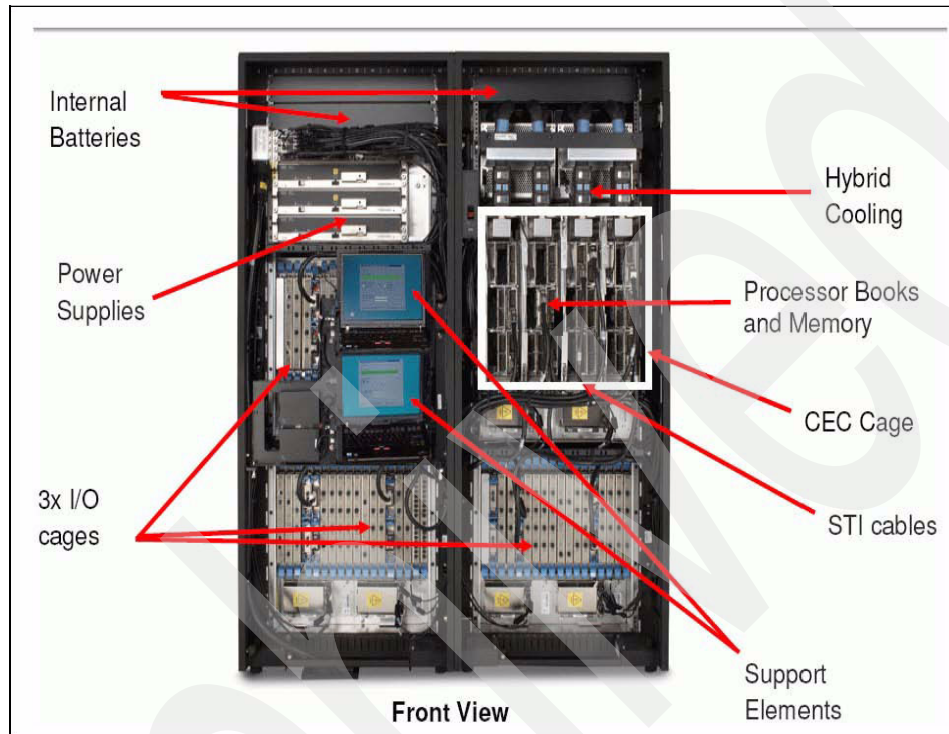


Figure 1-4 IBM System z9 mainframe

1.6 Processor units

In this section, we provide an overview of the different types of processors that are used on a mainframe.

1.6.1 Multiprocessors

Although it is possible to purchase a current mainframe with a single processor (CP), it would not be a typical system.³ The term *multiprocessor* means several processors (CP processors), and it implies that several processors are used by an instance of the System z operating systems, such as z/OS or z/VM.

The earliest operating systems were used to sequentially control single-user computer systems. In contrast, current computer systems are capable of *multiprogramming*, which means that they execute many programs concurrently. When a job cannot use the processor,

³ All current IBM mainframes also require at least one System Assistance Processor (SAP), so the minimum system has two processors: one CP and one SAP. However, the use of the term processor in the text usually means a CP processor that is usable for applications. Whenever we discuss a processor other than a CP, we always make this clear.

perhaps because it needs to wait for an asynchronous I/O operation to complete, through multiprogramming the system can suspend the job, thus freeing the processor to work on another job. When the I/O operation completes, the currently executing piece of work is interrupted and the suspended job is scheduled to run.

Most modern operating systems today, from mainframes to personal computers, can function in a multiprocessor environment. However, the degree of integration of the multiple processors varies considerably. With the mainframe, for example, pending interrupts in a system can be accepted by any processor in the system. Any processor can initiate and manage I/O operations to any channel or device that is available to the system. Channels, I/O devices, interrupts, and memory are owned by the system and not by any specific processor.

This multiprocessor integration appears simple on the surface, but its implementation is complex. It is also important for maximum performance because the ability of any processor to accept any interrupt sent to the system is especially important.

Operating systems make multiprogramming possible by capturing and saving status information about the interrupted program before allowing another program to execute. When the interrupted program is ready to begin executing again, it can resume execution just where it left off. Multiprogramming enables the operating system to run thousands of programs simultaneously.

When a computer system has multiple processors, the operating system supports multiprocessing, where each of the multiple processors can simultaneously process separate instruction streams from the various programs. The multiple processors share the various hardware resources, such as memory and external disk storage devices. Multiprocessing enables multiple programs to be executed simultaneously and allows a single program to use multiple processes in the same program to do parallel work.

The System z environment has supported multiprogramming and multiprocessing for its users for many years.

1.6.2 Processor types

In any given mainframe there could be multiple processors with each one processing different kinds of software. Figure 1-1 on page 4 lists several different classifications of processors that are tailored to various kinds of work. Although each processor is of the same System z architecture, they are to be used for slightly different purposes.⁴ Several of these purposes are related to software cost control, while others are more fundamental.

All of these processors start as equivalent processor units⁵ (PUs) or engines. A PU is a processor that has not been characterized for use. Characterized in this context means restricting the type of code that can be executed on a given processor.

Each of the processors begin as a general purpose processor and is characterized by the manufacturer during installation or at some later time. The potential characterizations are:

- **Central Processor (CP)**

This is a processor that is available to normal operating system and application software.

⁴ Do not confuse these with the controller microprocessors. The processors that we discuss in this section are full, standard mainframe processors.

⁵ This discussion applies to the current System z and System z machines at the time of writing. Earlier systems had fewer processor characterizations and even earlier systems did not use these techniques.

- ▶ System Assistance Processor (SAP)

Every modern mainframe has at least one SAP; larger systems might have several. The SAPs execute internal code⁶ to provide the I/O subsystem.

An SAP, for example, translates device numbers and real addresses of CHPIDs, control unit addresses, and device numbers. It manages multiple paths to control units and performs error recovery for temporary errors. Operating systems and applications cannot detect SAPs, and SAPs do not use any “normal” memory. See 1.10, “I/O connectivity (channels)” on page 18 for more information about SAPs.

- ▶ Integrated Facility for Linux (IFL)

An IFL is almost exactly the same as a normal central processor. The only difference is that the IFL lacks two instructions that the CP has, which are used only by z/OS. Linux and z/VM do not use these instructions.

The difference in using an IFL with Linux and z/VM from z/OS is that an IFL is not counted when specifying the model number⁷ of the system and thus does not contribute to the performance of the machine when it comes time to license certain software packages, which can make a substantial difference in software costs; therefore, many users opt for IFLs to run z/VM and Linux.

- ▶ Spare

An uncharacterized PU functions as a spare. If the system controllers detect a failing CP or SAP, it can be replaced with a spare PU. In most cases, this occurs without any system interruption, even for the application that is running on the failing processor.

Various forms of *Capacity on Demand* (CuOD) and similar arrangements exist whereby a customer can enable additional CPs at certain times (for unexpected peak loads, for example).

- ▶ Integrated Coupling Facility (ICF)

An ICF runs special code that is used to *couple* together multiple z/OS systems into a cooperative environment.

Note: Other important mainframe processors exist, but they are of less importance than the CPs that are the fundamental application processors, for example, the System z Integrated Information Processor (zIIP) is a specialized engine for processing eligible database workloads. The System z Application Assist Processor (zAAP) is a processor with a number of functions that are disabled (interrupt handling, some instructions) such that no full operating system can be executed on the processor. However, z/OS can detect the presence of zAAP processors and uses them to execute Java code (and possibly other similar code in the future). These processor types exist only to control software costs. There are also processors specialized for cryptography.

In addition to these characterizations of processors, some mainframes have models or versions that are configured to operate slower than the potential speed of their CPs to decrease purchase and operations cost, which is widely known as *throttling* or *capacity setting*. IFLs, SAPs, zAAPs, and ICFs always function at the full speed of the processor because these processors do not count in software pricing calculations.⁸

⁶ IBM refers to this as Licensed Internal Code (LIC). It is often known as microcode (which is not technically correct) or as firmware. It is definitely not user code.

⁷ Some systems do not have different models, but in this case, a *capacity model number* is used.

⁸ This is true for IBM software but might not be true for all software vendors.

1.7 Memory hierarchy

Getting data to the processor quickly and efficiently is a major task for all systems, and all major modern computing architectures have some level of hierarchy that is used for getting and storing data for execution on a processor. The mainframe is no different. The memory in a mainframe is typically referred to as storage or more technically, real storage. The term memory is used as the equivalent in personal computer computing. Some people in the mainframe community refer to hard disk units as storage too, but we recommend that you use the term storage only for the equivalent of personal computer memory.

Figure 1-5 illustrates the hierarchy on the mainframe.

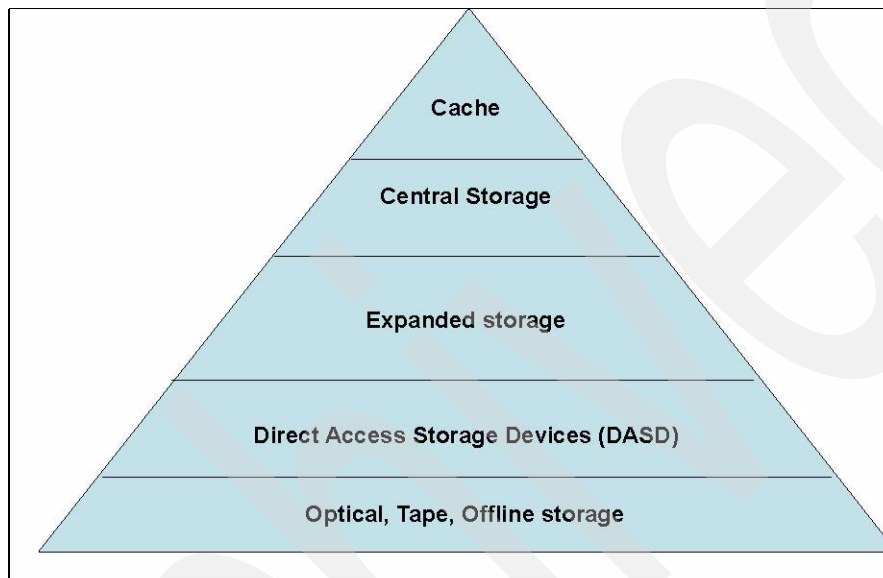


Figure 1-5 Mainframe memory/storage hierarchy

In this list, we explain the terms in Figure 1-5:

- **L1/L2 Cache**

The cache stores frequently accessed instructions for faster execution by the processor.

- **Central Storage**

Central storage contains the current running operating system and any processes or programs and data that the operating system uses. The amount of main storage that is supported depends on the addressability of the processor architecture. z9 supports 512 GB today, but the architecture supports up to 16 Exabytes (EB).

The amount of main storage that can be addressed is constrained by the operating system implementation, as well, for example, S/390® processors represented memory addresses as a 32-bit number but the later generation of zSeries, and the most current iteration known as System z, can address memory as a 64-bit number.

- **Expanded storage**

Expanded storage is needed to exploit certain special software facilities, and it is also used as a faster paging device.

Note: In the past, when memory was expensive, another type of memory was built on top of physical memory, called Expanded storage. This memory was built by using physical memory that was slower and less costly than the physical memory that was used for the central memory. Expanded storage was addressable only at the page level; therefore, instructions could not be executed from Expanded storage.

Unlike z/OS, which no longer uses Expanded storage, the z/VM operating system uses expanded memory as a high-speed paging device. Expanded storage is now implemented using the same physical memory as the central memory.

- DASD storage

With cached control units, DASD are the fast devices that are external to the processor hardware.

- Peripheral storage

Peripheral storage is mainly used for long-term persistent storage and is less expensive in comparison to the previous types. Peripheral storage includes Tape, optical storage devices, storage area networks (SAN), and SCSI I/O devices for Linux on System z.

1.8 Networking the mainframe

The various components that are required to perform networking with the mainframe are explained throughout this text. Here, we only mention that the interface on the mainframes for networking is known as the *Open System Adapter (OSA)*,⁹ which is the LAN adapter for the mainframe. The Integrated Console Controller (ICC) is also configured through the OSA card, which eliminates the need for a separate control unit for the system consoles.

For detailed information about networking a mainframe, refer to *Introduction to the New Mainframe: Networking*, SG24-6772.

1.9 Disk devices

In the mainframe environment, disks are usually referred to as DASD, which stands for *Direct Access Storage Device*. Although similar in concept to a hard disk in a personal computer or a mobile computer, DASD typically comprises many drives in a far more sophisticated arrangement.

Another key difference between mainframe DASD and a personal computer type of hard disk is that the physical disks are external to the mainframe. The device that houses the physical disks can be as large as, or larger than, the mainframe.

IBM 3390 disk architecture is commonly used on current mainframes. Originally the 3390 was a hardware model that was sold with earlier mainframes. Conceptually, the 3390 system is a simple arrangement, as shown in Figure 1-6 on page 15.

⁹ There are more exact names involved, such as OSA-Express3, but the name OSA is often in a generic sense as we use it in this paragraph.

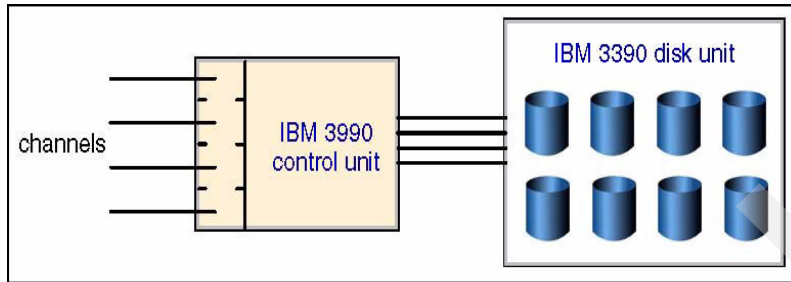


Figure 1-6 Early IBM 3390 disk implementation

Figure 1-6 shows 3990 and 3390 units, and it also represents the concept or architecture of current devices. Historically, each 3390 control unit (3990) could have up to eight channels connected to one or more processors, and the 3390 disk unit typically had eight or more disk drives. The concept of the control unit is similar to thinking about the printed circuit board that is found on the bottom of a mobile computer and personal computer hard disks, although the current control units are far more complex. The control unit in those cases is the circuit board. The purpose of a control unit is to control access to the device while managing data reads and writes.

Modern mainframe DASD units are very sophisticated devices. They emulate a large number of control units and associated 3390 disk drives. The Host Adapters appear as control unit interfaces and can connect up to 32 channels (ESCON or FICON).

The physical disk drives are a serial interface known as *Serial Storage Architecture* (SSA), which provides faster and redundant access to the disks.¹⁰ A number of internal arrangements are possible, but the most common involves many Redundant Array of Independent Disks 5 (RAID 5) arrays with hot spares.

Note: RAID refers to a family of methods for ensuring higher performance or reliability by using commodity hard drive technology. The benefits of RAID are handled for you on modern DASD units, so users can enjoy the benefits without the effort of configuration.

The current equivalent devices are the IBM 2105 Enterprise Storage Server® (ESS) and the IBM System Storage DS8000 family of disk subsystems. Practically everything in the unit has a spare or fallback unit. The internal processing (to emulate 3990 control units and 3390 disks) is provided by four high-end RISC processors in two processor complexes.¹¹ Each complex can operate the total system. Internal batteries preserve transient data during brief power failures. A separate console is used to configure and manage the unit.

Figure 1-7 on page 16 shows a simplified illustration of modern DASD units.

¹⁰ The DS8000® family uses Fibre Channel disks in a Fibre Channel arbitrated loop (FC-AL) configuration.

¹¹ The DS8000 family uses a cluster of p5 systems with many advanced features, such as storage partitions and so forth.

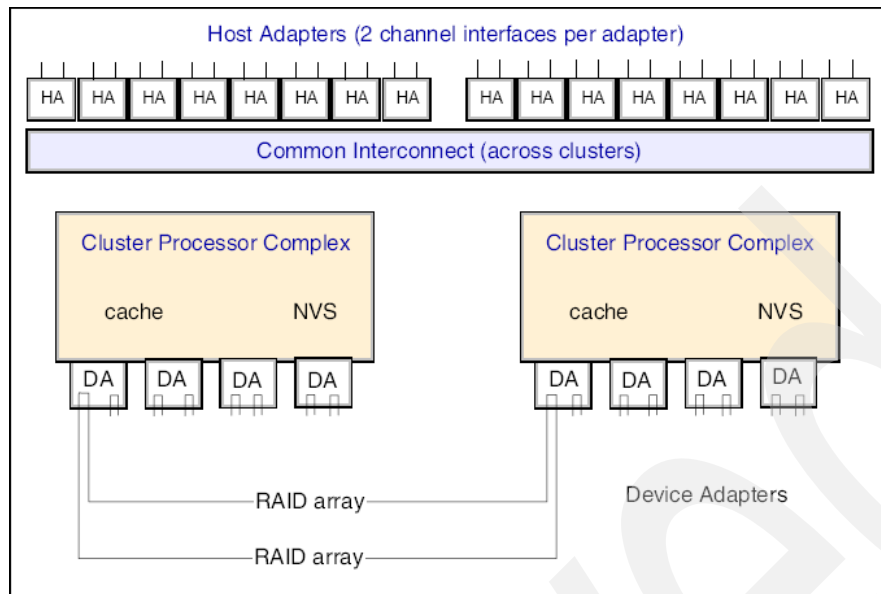


Figure 1-7 Current IBM 3390 implementation

The IBM 2105 and IBM DS8000 offers many functions not available in real 3390 units, including FlashCopy®, Extended Remote Copy, Concurrent Copy, Parallel Access Volumes (PAV), Multiple Allegiance, a larger cache, and so forth.

Clearly the simplistic 3390 disk drive and associated single control unit has much different underlying technology from the modern DASD units that we previously discussed. However, in keeping with the backward compatibility that permeates the mainframe world, the basic architectural appearance to software is the same, which allows applications and system software that is written for very old 3390 disk drives to use the newer technology with no revisions.¹²

There were several stages of new technology implementing 3390 disk drives, and the DS8000 is the most recent of these. The process of implementing an architectural standard (in this case the 3390 disk drive and associated control unit) with newer and different technology while maintaining software compatibility is characteristic of mainframe development.

1.9.1 Types of DASD

In this section, we briefly describe the types of DASDs that are used on the mainframe systems.

Extended count key data

Extended Count Key Data (ECKD™), developed from the earlier Count Key Data (CKD) DASD, organizes its data in *tracks*, which equate to the path that a single read/write head on a disk drive makes in one revolution. The disks had multiple read/write heads that read data from multiple disk surfaces, so there were multiple tracks available whenever the read/write heads were in a particular position, which is known as a *cylinder*, which is a term that you will come across frequently when working with DASD and System z operating systems. A Model-3 3390 DASD has 3339 cylinders available, for example.

¹² Some software enhancements are needed to use some of the new functions, but these are compatible extensions at the operating system level and do not affect application programs.

Fixed block architecture

Fixed block architecture (FBA) addresses the DASD data differently in that the DASD is formatted into 512-byte blocks and FBA are addressed sequentially starting at the first block and numbered up to the end of the DASD.

SCSI

Small Computer System Interface (SCSI) drives are a recent addition to z/VM. They are supported by z/VM itself like the fixed block architecture 9336-020 drives with up to a maximum of 2147483640 blocks. If not used by z/VM, they can be attached to operating systems that have the relevant support enabled.

1.9.2 Basic shared DASD

Within a mainframe environment it is usual to share data across systems and applications within those systems. Within a single z/VM system¹³ it is the z/VM Control Program that controls access to devices. However, if the DASDs are shared among different systems, the operating systems must have a mechanism to not allow concurrent updates to the same data, which is done using channel commands RESERVE and RELEASE.

Figure 1-8 illustrates a basic shared DASD environment. The figure shows z/VM images, but these could be any earlier version of the operating system, which could be two operating systems running on the same system or two separate systems. There is absolutely no difference in the concept or operation.

The capabilities of a basic shared DASD system are limited. The RESERVE command limits access to the entire DASD to the system that is issuing the command, and this lasts until a RELEASE command is issued. These commands work best when used to reserve DASD for a limited period of time.

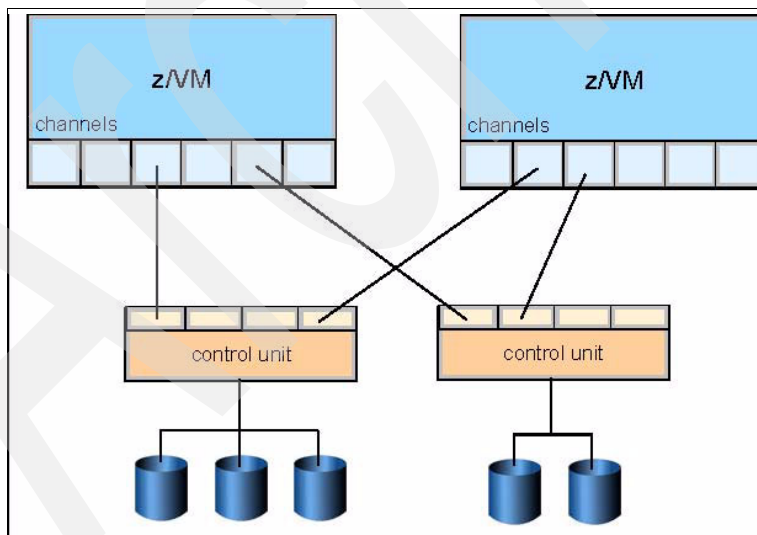


Figure 1-8 Basic shared DASD

A real system has many more control units and devices than shown in Figure 1-8. Other types of devices or control units can be attached to both systems, for example, a tape control unit

¹³ This discussion uses the z/VM operating system for examples, but the same concepts apply to z/OS and other System z operating systems.

with multiple tape drives can be attached to both systems. In this configuration, the operators can then allocate individual tape drives to the systems as needed.

1.10 I/O connectivity (channels)

As we mentioned earlier, devices are connected to the mainframe CEC through channels. Unlike the old parallel channels used on the original mainframe architecture diagram (see Figure 1-2), modern mainframe channels connect to only one control unit or, more likely, are connected to a *director* that handles the multiple paths to devices. Another difference is that modern channels are connected using optical fibers instead of the traditional copper wires. In the following sections, we discuss channel operations in more detail.

1.10.1 Channel subsystem

One of the main strengths of the mainframe computers is the ability to deal with a large number of simultaneous I/O operations. The *channel subsystem* (CSS) has a major role in providing this strength. The CSS manages the flow of information between I/O devices and central memory. By doing so, it relieves CPUs of the task of communicating directly with I/O devices and permits data processing to proceed concurrently with I/O processing.

The channel subsystem is built on the concept of a System Assist Processor (SAP) and the concept of channels. The SAP is one of the System z processor types. The SAP uses the I/O configuration loaded in the Hardware Storage Area (HSA) and knows which device is connected to each channel and that device's protocol. The SAP manages the queue of I/O operations that the operating system passes the channel subsystem.

Physical channel types

The channel subsystem can contain more than one type of channel path:

- ▶ Enterprise Systems Connection (ESCON)
Since 1990, ESCON replaced the S/370 parallel channel as the main channel protocol for I/O connectivity using fiber optic cables and a new switched technology for data traffic.
- ▶ FICON
Based on the Fibre Channel Protocol (FCP), FICON was introduced in 1998. Because it is much faster than ESCON, it is becoming the most popular connection type.
- ▶ OSA-2 Open Systems Adapter-2
A networking type of connector, OSA-2 Open Systems Adapter-2 can be configured to support various network protocols, such as Ethernet, Fast Ethernet, token-ring, and Fiber Distributed Data Interface (FDDI).
- ▶ OSA Express
A faster networking connector, OSA Express supports fast Ethernet, Gigabit Ethernet, and 10 Gigabit Ethernet.
- ▶ OSA Express2
OSA Express2 adds support for IBM Communication Controller for Linux, which provides a migration path for systems with dependency on SNA networking.

IOCDs

The I/O control layer uses a control file known as an I/O Configuration Data Set (IOCDs) that translates physical I/O addresses (composed of CHPID numbers, switch port numbers,

control unit addresses, and unit addresses) into *device numbers* that are used by the operating system software to access devices, which is loaded into the HSA at power on (power-on Reset, or POR) and can be modified dynamically.

A device number looks like the addresses that we described for early S/360 machines except that it can contain three or four hexadecimal digits. Without an IOCDS, no mainframe operating system can access I/O devices.

A subchannel provides the logical appearance of a device to the program and contains the information that is required for performing a single I/O operation. One subchannel is provided for each I/O device addressable by the channel subsystem when the system is activated.

ESCON and FICON

Recall our earlier discussion of the generic channel concepts born from the System 360 architecture. Current channels, such as ESCON and FICON, are logically similar to parallel channels but they use fiber connections and operate much faster. A modern system might have 100 to 200 channels or CHPIDs.¹⁴ The key concepts are:

- ▶ ESCON and FICON channels connect to only one device or one port on a switch. When connected to a switch (port), many devices can be accessed simultaneously.
- ▶ Most modern mainframes use switches between the channels and the control units. The switches can be connected to several systems, sharing the control units and some or all of its I/O devices across all of the systems. The main advantage of using switches is that we can share a single I/O channel to connect to many I/O devices.
- ▶ CHPID addresses are two hexadecimal digits.
- ▶ Multiple partitions can sometimes share CHPIDs. Whether this is possible depends on the nature of the control units that are used through the CHPIDs. In general, CHPIDs that are used for disks can be shared.
- ▶ An I/O subsystem layer exists between the operating systems in partitions (or in the basic machine, if partitions are not used) and the CHPIDs.

Switches and directors

An ESCON director, FICON director, or switch is a sophisticated device that can sustain high data rates through many connections. A large director might have 200 connections, for example, and all of these can be passing data at the same time.

The difference between a director and a switch is that a director is somewhat more sophisticated and contains extra functionality, such as built-in redundancy for maximum fault tolerance. The director or switch must keep track of which CHPID (and partition) initiated which I/O operation so that data and status information is returned to the right place. Multiple I/O requests, from multiple CHPIDs attached to multiple partitions on multiple systems, can be in progress through a single control unit.

Modern control units, especially for disks, often have multiple channel (or switch) connections and multiple connections to their devices, as shown in Figure 1-9 on page 20. They can handle multiple data transfers at the same time on the multiple channels.

¹⁴ The more recent mainframe machines can have more than 256 channels, but an additional setup is needed for this. The channels are assigned in a way that only two hexadecimal digits are needed for CHPID addresses.

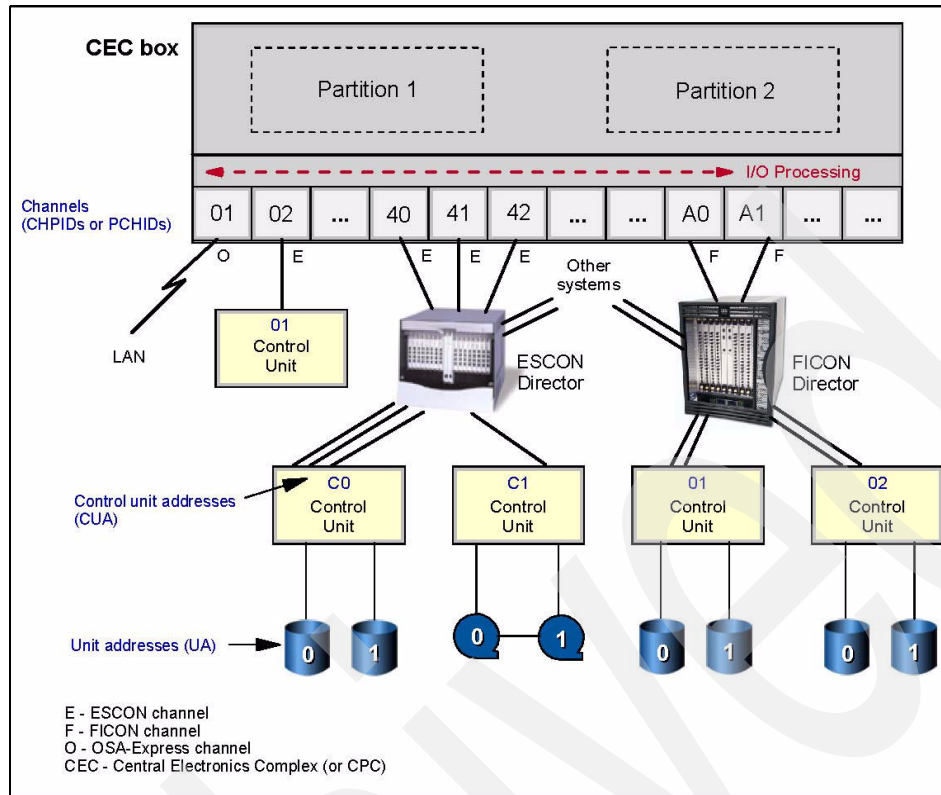


Figure 1-9 Recent system configuration

Logical channel subsystem

The Logical Channel Subsystem (LCSS) concept was introduced with the more recent mainframes. The LCSS was created to enable the System z environment to handle more channel paths and devices available to the server. Each LCSS can have from 1 to 256 channels.

1.11 System control and partitioning

At this point, we presented almost all of the types of hardware that are needed for a basic understanding of the modern mainframe architecture. Having said that, we still have not described how to control the mainframe or how to partition the vast resources that they contain.

1.11.1 Controlling the mainframe

There are many ways to illustrate a mainframe's internal structure, depending on the point of emphasis. Figure 1-10 on page 21, while highly conceptual, shows several of the functions of the internal system controls on current mainframes. The internal controllers are microprocessors, but they use a much simpler organization and instruction set than System z processors. They are usually known as controllers to avoid confusion with System z processors.

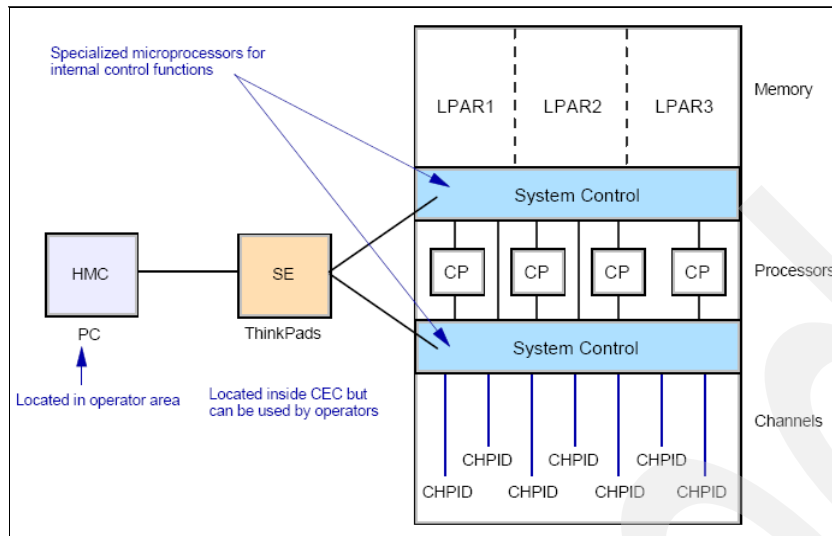


Figure 1-10 System control and partitioning

Tasks, such as partitioning, allocating resources to partitions, and so forth, on a mainframe are performed by using the HMC and SE, as discussed throughout this document.

1.11.2 Logically partitioning resources

The Processor Resource/Systems Manager™ (PR/SM™) is a feature of IBM mainframes that enables logical partitioning of the CEC. A logical partition (LPAR) is a virtual machine at the hardware level. It is the PR/SM functions that allow LPARs to be created and used. Using the IBM System z mainframes you can divide your physical machine into one or more LPARS (virtual machines), each of which contains a subset of your real machine's processors, memory, and input/output devices, which enables users to consolidate workloads and operating environments, that are currently running on separate systems, into one physical system while using resource sharing to improve resource utilization and to maintain performance.

Each LPAR operates as an independent server running its own operating environment. On the latest System z models, you can define up to 60 LPARs running z/VM, z/OS, Linux on IBM System z, and others. PR/SM enables each LPAR to have dedicated or shared processors, I/O channels, and dedicated memory (most of the resources can be dynamically reconfigured as needed). In other words, PR/SM transforms physical resources into virtual resources so that several logical partitions can share the same physical resources.

LPARs are generally equivalent to separate mainframes. Each LPAR runs its own operating system, which can be any mainframe operating system. The installation planners might elect to share I/O devices across several LPARs, but this is a local decision.

The HMC system administrator can assign one or more system processors for the exclusive use of an LPAR. Alternatively, the administrator can allow all processors to be used on some or all LPARs. Here, the system control functions (often known as *microcode* or *firmware*) provide a dispatcher to share the processors among the selected LPARs. The HMC administrator can specify a maximum number of concurrent processors executing in each LPAR. The administrator can also provide weightage for different LPARs (specifying, for example, that LPAR1 must receive twice as much processor time as LPAR2).

Using the HMC, System administrators assign portions of memory to each LPAR, and memory cannot be shared among LPARs. The administrators can assign processors (noted as CPs in Figure 1-10 on page 21) to specific LPARs or they can allow the system controllers to control any or all of the processors to all of the LPARs using an internal load-balancing algorithm.

For most practical purposes, there is no difference between, for example, three separate mainframes that are running z/VM (and sharing most of their I/O configuration) and three LPARs on the same mainframe doing the same thing. With minor exceptions, z/VM, the operators, and the applications cannot detect the difference.

Now that you have a firm grasp of the underlying mainframe hardware, “Systems management overview” on page 23 further explains the concepts of managing your mainframe.

1.12 Exercises and discussions

To help test your understanding of the material in this chapter, answer the following questions:

1. *Multiprocessor* means several processors that are used by the operating system *and* applications. What does *multiprogramming* mean?
2. Usually, each read or write operation on a non-shared DASD device is one I/O operation. How many I/O operations are involved when the DASD is in a shared mode?
3. What is the role of PR/SM?
4. What changes are needed for z/VM applications to work in an LPAR?

Systems management overview

Before presenting the HMC and SE, we must introduce the topic of systems management. As you saw in Chapter 1, “Introduction to System z Hardware” on page 1, the System z is a very large, scalable, and robust hardware platform. Controlling, monitoring, installing, configuring, and handling error conditions are what we mean when we use the term systems management. As an HMC System Programmer, Systems Administrator, or Systems Operator you must understand both the concepts of systems management, System z Hardware, and how the HMC/SE manages and monitors System z hardware.

After completing this chapter, you will:

- ▶ Understand System Management disciplines.
- ▶ Understand how the HMC and SE accomplish System z System Management.
- ▶ Understand the capabilities of the HMC.
- ▶ Understand the capabilities of the SE.
- ▶ Understand how the HMC and SE provide a single point of control and single system image of the System z.

2.1 Introduction to systems management

Systems management is a general term that is widely used and whose meaning is dependent on the context. In some cases, systems management means the operator interface, while in others, it means provisioning capacity. In a large-scale commercial system, systems management usually is considered to be “A collection of disciplines that monitor and control a system’s behavior.” The disciplines that are included in System z management are:

- ▶ System z control disciplines:
 - Business management: The techniques, resources, and tools that are used to ensure that the system resources are controlled and auditable to the customer’s defined IT processes.
 - Configuration management: The techniques, resources, and tools that are used to initialize, customize, and maintain an accurate inventory of the hardware.

- Performance management: Techniques, resources, and tools that are used to define, monitor, and eventually act upon the overall system performance.
- Operations management: Techniques, resources, and tools that are used to have more effective and error-free operator communications to ensure that system operations are efficient and effective.
- ▶ System z serviceability disciplines:
 - Change management: Techniques, resources, and tools that track and manage firmware changes (usually upgrades) and their dependencies and implications.
 - Problem management: Techniques, resources, and tools that detect, open, follow, resolve, and close any problem (hardware, software, or application), which includes directing the customer or engineer through the repair process.

Many businesses concentrate more on running their day-by-day operations than on planning for or anticipating critical situations; therefore, the implementation of their systems management disciplines is frequently driven by critical situations, such as unacceptable response time, client complaints, problems that are not being solved in a timely manner, changes in software or hardware that create problems, changes in the business environment that require more computational resources, and so on.

However, most of the information about how a system is performing, which is needed for systems management, is designed into the System z hardware, firmware, and software. The HMC/SE monitors, collects, logs, and registers a great deal of valuable system information. The HMC user interface can then order this information in a way that is readable and significant to people, so that immediate decisions can be made or the appropriate planning can occur to accommodate or provide new resources.

2.1.1 System data

In a System z environment, the HMC/SE is responsible for collecting and organizing the hardware data, even if the data is generated by different hardware components (the hardware itself, HMC/SE, PR/SM, Network, Service components, and so on).

The usual flow of information is that the system, or subsystem, gets the data and puts it in a common repository. This data is usually raw data that must be interpreted by the HMC. Therefore, every component that writes data into the HMC has an HMC function to read it and report about it. In addition, there are also tools that allow this data to be retrieved from the HMC through programmable interfaces for remote monitoring and consolidation. The HMC and SE data collection components for accounting, reporting, and managing a system include:

▶ Data collection in the HMC

Given the size and complexity of a System z, system data is characterized into different contextual areas. Understanding the pervasive system data that is available gives you a reference point as we describe the HMC functions throughout this book.

▶ Activation Profiles

The Activate function controls the starting up of the system, which includes power-on reset, partition activation, and initial program load of your operating system software. Activate is the primary function for CPC or CPC image start up. Activate senses the status of the object, and then performs only those actions that are necessary to get the object to an operational state, for example, if the CPC is already in the power-on reset complete state, Activate skips the power-on reset step and proceeds with loading the operating system. If the CPC allows activation to include a power-on and the CPC is powered-off, Activate will power it. The Activate function uses profiles that contain the parameters that

are needed to put the system in an operational state. The profiles are stored in each CPC Support Element.

Activation Profiles are required for CPC and CPC image activation. They tailor the operation of a CPC and are stored in the Support Element that is associated with the CPC. There are four types of activation profiles:

- | | |
|--------------|---|
| Reset | Every CPC in the processor cluster needs a reset profile to determine the mode in which the CPC Licensed Internal Code is loaded and how much main storage and expanded storage is used. The maximum number of reset profiles that are allowed for each CPC is 26. |
| Image | If logically partitioned (LPAR) mode is selected in the reset profile, each partition has an image profile. The image profile determines the number of CPs that the image uses and whether these CPs are dedicated to the partition or shared. It also allows you to assign the amounts of main storage and expanded storage to be used by each partition. Depending on the Support Element model and machine type, the maximum number of image profiles that are allowed for each CPC can be between 64 and 255. |
| Load | A load profile is needed to define the channel address of the device from which the operating system is loaded. Depending on the Support Element model and machine type, the maximum number of load profiles allowed for each CPC can be between 64 and 255. |
| Group | A group profile defines the group capacity value that can be customized in determining the allocation and management of the processor resources that are assigned to the logical partition in a group. This profile will not contain the name(s) of the LPAR images that make up the group. |

► **Security Logs**

The security logs contain the security events that are logged for the Hardware Management Console or a server (CPC). A security event occurs when an object's operational state, status, or settings change or involves user access to tasks, actions, and objects.

► **Traces**

One important tool for debugging firmware problems is the tracing of program steps. On the SE, this occurs through a central wraparound buffer in memory. There is a common routine that any program can call to create a trace entry. The SE coding conventions require tracing of the entry and exit of every major function. Additionally, a trace can be taken at any critical point in the code that the developer deems is helpful in debugging a code problem.

► **Memory dump**

If the system encounters problems that do not allow further execution of the CEC firmware, a dump of the entire hardware system area (HSA) memory is written to the SE disk. This happens only in rare cases in which recovery cannot handle the failure. It holds all system state information at the moment the CECDUMP was written.

► **Event logs**

The event logs track all system events and are useful for reconstructing a sequence of actions.

► Task logs

The task log tracks the last 1000 tasks performed including the object(s) of the task. Task Log tracks how many times each task is used and the date and time of the last use.

► System error logs

The system error log is the cornerstone of the First Error Data Capture (FEDC) infrastructure. The three main purposes of the system error log are to:

- Journal all significant system events
- Be a problem data repository for automatic problem analysis (auto PA)
- FEDC storage area. A key enhancement to the system error log was the creation of a table-driven approach that enabled attaching the necessary files whenever needed. Therefore for certain groups of log events, files can be collected and added to the FEDC data.

► HMC console logs

The Hardware Management Console log file (IQYYLOG.LOG) permits scrolling through previous HMC console messages and commands.

► Support element data

Support element data consists of Support Element's upgrade data and the Support Element's Input/Output Configuration data. The Support Element data is used only when a system is being upgraded.

► Service data

The service data is a set of system information, such as program and event traces, that the CPC's Support Element collects. Service data assists IBM in servicing the System z machines.

► HOM configuration data

To initialize the hardware, POR needs knowledge of the hardware topology. Also the exact hardware level and defective information is needed. HOM provides this information.

► Engineering data

The engineering data is needed to initialize all chips in the system during POR.

► CEC configuration records

The CEC configuration record contains machine configuration data, such as model number. The record is built on the SE and transferred to the CEC.

► IO configuration records

This record is built out of SE HOM information and contains records about which I/O cards are physically plugged in the system.

► I/O Configuration Data Set (IOCDS)

IOCDS is a (binary) file used by the I/O subsystem in z/Architecture systems. It is created from I/O Configuration Program (IOCP) source control statements. It defines the paths that are involved in accessing each I/O device and the device number (or software address) that the operating system sees for all I/O devices. It also defines the name and number for each LPAR and how various I/O resources are assigned to each LPAR.

► I/O topology information

The I/O firmware code senses the connections between the CEC cage and the I/O cages.

- ▶ Zone group files
This file contains persistent configuration information for the LPAR hypervisor, which is provided by the SE LPAR code.
- ▶ Licensed Internal Code Controlled Configuration (LICCC)
The LICCC contains the information about customer purchased capacity that is transferred from the SE to the CEC.
- ▶ Console User Settings Data
This function allows the user to define what is the acceptable status for various CEC elements. The user can also define the colors or patterns to be used for exception status displays.

2.2 Hardware Management Console introduction

You can use a Hardware Management Console (HMC) to manage System z machines, including partitions, I/O channels, and other resources in the System z hardware. It includes features for problem analysis, automatic real time notification of system events, and automatic reconfiguration and repair. The HMC also provides views of system resources and provides capabilities for system administration, including real-time monitoring, backup, and recovery.

The HMC is a management solution that helps you to manage your server resources in an efficient and effective way. It was initially a feature of S/390 servers starting in the early 1990s. This feature provided much more than just a user interface for the operation and management of the servers. Users of the HMC who were not familiar with the product sometimes thought of it as *just a hardware console*; however, the HMC is much more than *just a console*, for example as the hardware platform interface, an HMC provides:

- ▶ An expert system that performs analysis of system-level failures—besides server level failures—determines the root cause, and automatically notifies the customer and IBM Service provider of the problem, the impact of the problem, the fix, and the impact of the fix.
- ▶ Automatic backup and restore of configuration and customization data for the system's hardware components.
- ▶ Automatic and autonomic firmware change management for the servers, which includes automatic monitoring of firmware levels, updates, backups, retrieval and concurrent application of updates without any customer effort or involvement.
- ▶ A highly reliable *call home* server with automatic failover.
- ▶ Weekly information about the performance and use of systems in the field. This data is used, in part, to support On-Demand software billing.

You can use a single HMC to manage all System z and zServers that any customer has, while simultaneously allowing complete redundancy for up to 32 HMCs without additional customer management requirements. The HMC communicates with each server through the server's Support Element. When tasks are performed at the HMC, the commands are sent to one or more SEs that in turn issue commands to their servers. Servers can be grouped at the HMC so that a single command can be passed along to as many as all of the servers that are defined to the HMC. One HMC can control up to 100 SEs, and one SE can be controlled by up to 32 HMCs.

Figure 2-1 on page 28 illustrates the HMC.

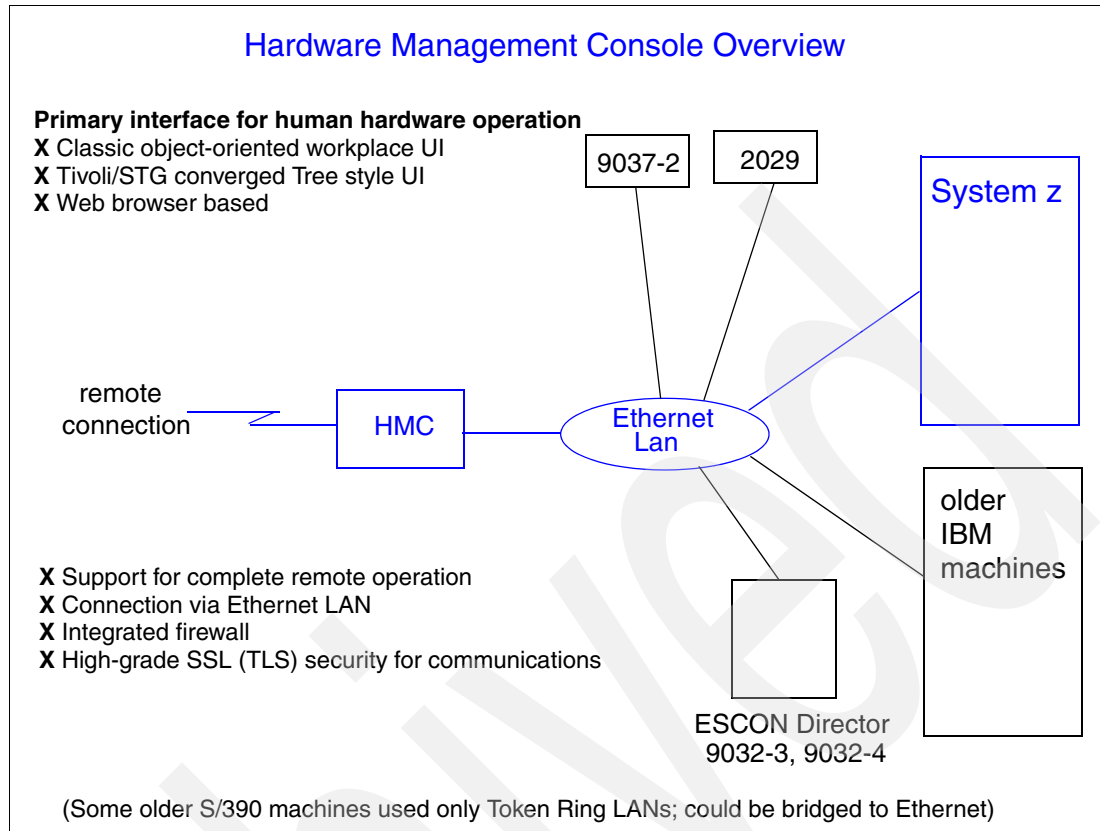


Figure 2-1 Hardware Management Console overview

In Figure 2-1, the key requirements and design points of the HMC and SE subsystem are:

- ▶ Single point-of-control and single system image for managed systems.
- ▶ Replicated single point-of-control (customer can tailor).
- ▶ Continuous monitoring for managed objects, providing instant (consolidated) status.
- ▶ Operations, change, service, and problem management convergence.
- ▶ Backwards compatibility.
- ▶ Support for complete and secure remote operation.
- ▶ Zero configuration for OS access and GDPS® support.
- ▶ Consolidated message processing for hardware and operating system messages.
- ▶ Single-point for creating, updating, replicating, and viewing profiles that define IML and IPL parameters for CPCs, CFs, and LPAR partitions.
- ▶ Automatic discovery and configuration of added managed systems.
- ▶ Serviceability functionality through Hot plug support for additional hardware, “Out-of-Band” concurrent firmware updates, automatic problem analysis and FFDC, automatic “call-home” with automatic redundant path support, and Out-of-Band concurrent repair.

The HMC user interface provides the functions that you need through an object-oriented user interface. Through this design, you can directly manipulate the objects that are defined to the HMC and be made aware of changes to hardware status as they are detected.

The main functions are:

- ▶ Single system image and single point-of-control for multiple systems. One single HMC can manage all System z systems.
- ▶ Exception driven with color/pattern support to illustrate problems to end-user.
- ▶ Up to 32 HMCs can simultaneously manage any given system.
- ▶ Full replication support among HMCs, which allows any HMC to control any System z.
- ▶ Support for complete remote operation.
- ▶ TCP/IP SNMP agent and Application Programming Interface (API) that provides the ability to build automation routines or collect and forward information using management APIs in REXX, Java, or C/C++ running on z/OS, Windows®, OS/2, and Linux.
- ▶ Single object operations to the System z SEs, which eliminates the need for working near the machine.
- ▶ Interface for console integration.
- ▶ Primary Interface for Human for Hardware Operation:
 - Drag-and-drop for functions, such as IPL, RESET, and LOAD
 - Classic object-oriented workplace UI
 - Tivoli® / STG Converged “Tree” style UI
 - Web-browser based

The HMC provides the platform and user interface that controls and monitors the status of the System z system using the two redundant Support Elements that are installed in each System z, as shown in Figure 2-2. The System z implements two fully redundant interfaces, known as the Power Service Control Network (PSCN), between the two Support Elements and the CPC. Error detection and automatic switch-over between the two redundant Support Elements provides enhanced reliability and availability.

When tasks are performed at the HMC, the commands are sent to one or more Support Elements, which then issues commands to their CPCs. The CPCs can be grouped at the Hardware Management Console so that a single command can be passed along to as many as all of the CPCs that are defined to the Hardware Management Console.

As shown in Figure 2-2, the CPC is contained in what is referred to as an A frame, while the Support Elements are held in a Z frame. The HMC is physically outside of either frame. A local area network (LAN) connection is necessary for the HMC to communicate with the CPC and its Support Elements.

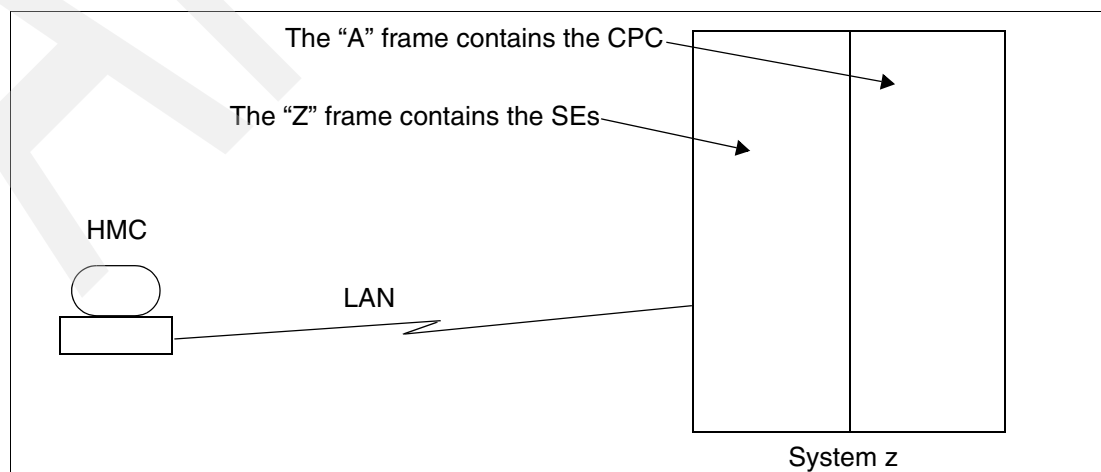


Figure 2-2 HMC and Support Element configuration: Single CPC Environment

CPCs can be grouped so that a single command that the HMC issues is passed along to multiple CPCs that the HMC defines. One HMC can control up to 100 Support Elements, and one Support Element can be controlled by up to 32 HMCs, as illustrated in Figure 2-3.

While an HMC is a physical PC machine, this course generally uses the term HMC to represent the underlying technology of the HMC. So, when we reference HMC Version 2.10.0, we are talking about the HMC system as a whole, including the current version of its user interface.

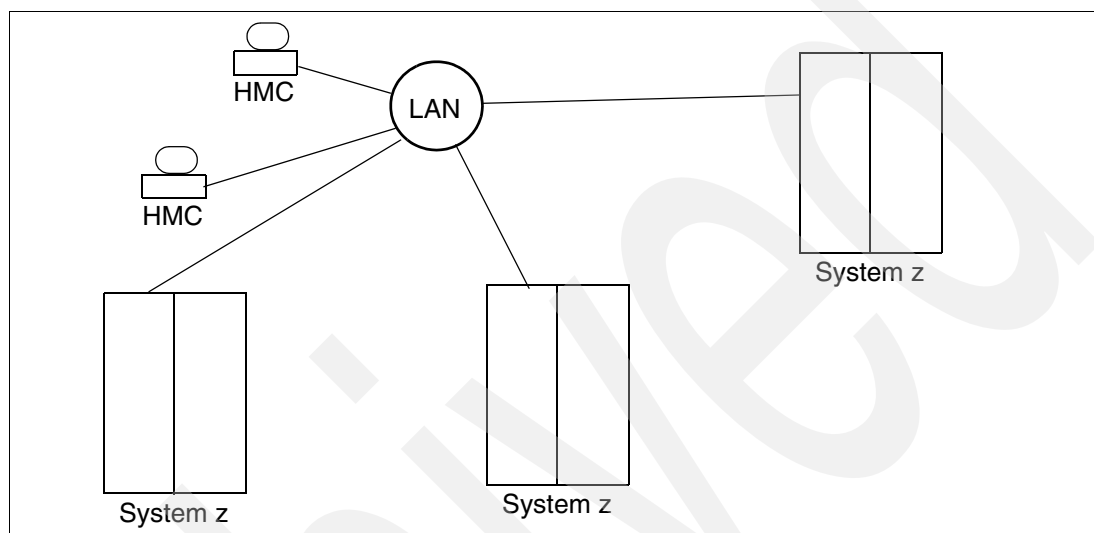


Figure 2-3 HMC and Support Element configuration: multiple CPC environment

To ensure the integrity and security of the environment, the HMC and SE operate on a closed platform, which means that the customer is not provided access to the underlying operating platform and is not permitted to install or run additional applications on the underlying operating platform that the HMC or SE are using. The sole purpose of the HMC and SE is to provide a platform for the execution of the HMC and SE application.

2.3 Support element introduction

The IBM System z is a large-scale server that is designed to meet the needs of customers at the high end of the marketplace. Such servers can run multiple operating systems simultaneously. The maintenance and management of these servers is done concurrently. With the IBM System z, an out-of-band system control structure was introduced to manage these complex systems. Such management tasks include testing the hardware before the operating system is loaded, loading the operating systems, concurrent repair, concurrent upgrade, reporting of and recovering from errors, and so on. To accomplish these tasks, a distributed service and control subsystem that consists of redundant Support Elements, cage controllers (CC), and communication links was implemented, as shown in Figure 2-4 on page 31.

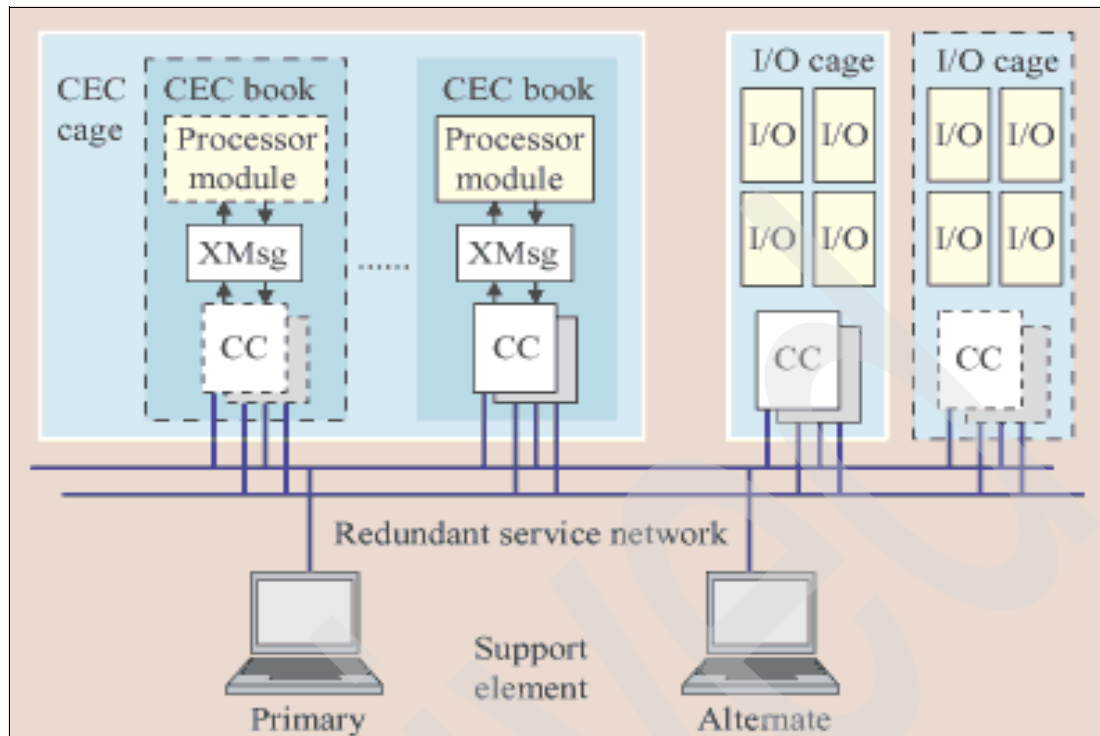


Figure 2-4 Support elements and redundant network

Support Element

The HMC communicates to each CEC through the Support Element (SE). A Support Element is a dedicated Lenovo ThinkPad that is used to monitor and operate a system. If you have experience using other systems, you might have used a processor console, support processor, or a similarly named workstation to monitor and operate them. The IBM System z has an *integrated support element*, that is, the Support Element is located inside of the same frame that the central processor complex is located. An alternate Support Element is also provided for the option to automatically failover and switch from the primary Support Element to the alternate Support Element if a hardware problem occurs.

The SE, plus an alternate SE, are supplied with each server to provide a *local* console to monitor and operate the system. The SE is usually used by service personnel to perform maintenance operations on the system. Using the communication path through the SE, the HMC can perform numerous operations and tasks to assist in the maintenance and operations of the Central Processor Complex, logical partitions, I/O, and other physical and logical entities on the system.

The firmware that is responsible for executing system management tasks runs on different system components: the processor module itself, the cage controller, and the Support Element. Certain system management tasks require the cooperation of *all* firmware components on the CEC, the cage controller, and the Support Element, for example, to replace an I/O card, concurrently with system operation:

- ▶ It must be removed from the SE view of the system configuration.
- ▶ The firmware that is running on the processors must be informed that it can no longer access this hardware, and when it can be powered off safely.
- ▶ The CC in the I/O cage where the card resides must be instructed to power off the card.

This is just one simple example of a management task, but it shows the need for communication between the involved firmware components. The Support Element is connected to the cage controllers through a redundant service network, which is an Ethernet network. The cage controllers in the CEC cage are connected to the processor modules through the XMsg-engine hardware in the clock chip. When firmware components that reside on the Support Element must communicate with firmware components that run on the processors, they communicate to the firmware on the cage controller.

2.4 HMC and SE system management disciplines

Now that we introduced systems management (the HMC and the SE), we can provide an overview of how the HMC and SE provide System z customers with advanced control and monitoring of their systems. Our purpose in this section is to provide an overview of the functions and capabilities of the HMC and SE as they relate to the System Management disciplines. In subsequent chapters, we describe how to use the HMC and SE, the detailed design behind the components, and the technologies that are used to provide System z Hardware hardware systems management.

2.4.1 HMC and SE business management functions

In this section, we discuss HMC and SE business management functions.

User ID and password

In the HMC users are identified by their user ID and password. The HMC Administrator role is responsible for setting up user accounts and determining the resources that each user has access to. The user authenticates to the HMC by logging in with their assigned user ID and password. User authority for a resource or task on the HMC is determined by the user role. Each user role is a list of tasks that the user is authorized to perform or a list of resources that the user is allowed to manage. The HMC-provided predefined task roles are Operator, Advanced Operator, System Programmer, Access Administrator, and Service Representative. Access to all objects and tasks on HMC are controlled using the user ID and password.

Evaluation Assurance Level 5 security logs

Evaluation Assurance Level 5 (EAL5) is the System z grade (EAL1 –EAL7) that the System z achieves after completing ongoing assessments, by an independent authority, against an international standard call *Common Criteria*. The purpose of this standard is to ensure that systems-security features are reliably implemented. To meet these standards, the HMC and SE log all:

- ▶ Local and remote accesses by user ID and if a remote access, then by IP address also
- ▶ Disruptive actions by user ID
- ▶ All profile and configuration changes (including what was changed) by user ID

Event log

The event log function tracks all system events, which are individual activities that indicate when processes occur, begin, end, succeed, or fail. When an event occurs, the date and time that it occurs and a brief description of the event is recorded in the event log. Through the Monitor System Events task, you can create and manage event monitors. Event monitors listen for events from objects that the Hardware Management Console manages. There are three types of events: Hardware messages, State changes, and Operating system messages. The Event log contains all of the data that is captured for an event.

Tasks log

The task log function tracks the last performed 1000 tasks, which includes the objects of the task. It also tracks how many times each task was used and the date and time of the last use, for example, the HMC has a workplace that supports functions, facilities, and controls that allow you to monitor and operate different entities, such as servers, logical partitions, groups of servers, and logical partitions. The HMC workplace represents tasks and their targets as icons. Using the workplace to get information and start tasks is a matter of monitoring and manipulating icons. An example of a task is the Activate task, which is found in the Daily group of tasks. An example of an object is the Central Processor Complex of the server or a logical partition. Activating a CPC performs the actions that are necessary to get the CPC into an operational state, such as a Power On, and then Power On Reset of the CPC, as defined in the CPC's Activation profile.

Integrated firewall

The network is an integral part of the HMC and its management of the associated System z or S/390 processors and related devices. The network is also one of the primary areas of concern related to the security and integrity of the HMC. To help address this area of concern, the HMC and the SE have integrated firewalls that are built into the underlying operating platform. The HMC and SE are configured with a firewall to limit network access in and out of the HMC and SE. By default, no external connections are allowed through the firewall. There are several ways that the firewall configuration is altered, which we discuss in a later chapter.

The firewall on the HMC and SE is designed to prevent access to network services that are running on the HMC and SE from unauthorized network hosts. This firewall operates at the network layer, which means that it filters packets at the IP level. The firewall operates based on a table of rules that the HMC/SE code manages. The initial state of this rule table is to prevent access to all services from all hosts. The HMC/SE then dynamically adds rules as needed to allow communication with authorized hosts. When communication with a particular host is no longer needed the rule that allows access from that host is removed.

All modifications to the firewall are handled automatically by the HMC/SE without any end-user interaction. The end-user can only affect the firewall as a side-effect of enabling other HMC services, such as Remote Operation, SNMP APIs, and Remote PE Access. If these services are enabled, the HMC/SE adds rules to the firewall that allow access to them from any host. These services have their own security measures built in to prevent unauthorized access.

The HMC and SE firewalls do not restrict any outgoing network connections, and the firewall does not process packets that originate from the HMC/SE; instead, the packets are sent immediately to their destination. When the firewall encounters an incoming packet that is not allowed by any of its rules, it simply drops the packet, which prevents it from reaching the destined service on the HMC/SE. No response or acknowledgement is sent to the originator of the dropped packet. In most situations, the originator of the dropped packet simply retries until some timeout is reached.

2.4.2 HMC and SE configuration management functions

Many functions are involved with HMC and SE configuration management. In this section, we describe the key elements.

Vital product data

The purpose of vital product data (VPD) is to provide an accurate hardware product and component tracking process for service to debug the right level of hardware and for upgrades to hardware to be handled automatically. VPD is non-volatile data that uniquely identifies

each System z hardware component (configuration data, serial number, EC level, and so forth). Vital product data consists of hardware information, such as part numbers, serial numbers, and engineering change levels. Although not all devices contain VPD data, the HMC provides a user interface. Most of the hardware components VPD is automatically sensed. For hardware that cannot be sensed, such as frames, cables, and so forth, the HMC provides a graphical editor to associate a unique identifier with them. The machine VPD file is a listing of all configuration and hardware components in the System z.

Whenever any component changes, such as a processor is spared or a part is removed, added, or replaced, the VPD is updated. The Machine VPD is scanned daily for spared CPs and FRU exchanges. This data is then used to call home if preventative service is required.

Concurrent management of I/O configuration

All cards in the I/O subsystem, including the cage power and cooling structure, allow for concurrent upgrade. The upgrade includes adding additional channels that are concurrent with system operation. Concurrent LIC upgrades are also supported for all of the System z I/O cards.¹

Concurrent book add

System z provides the capability to concurrently upgrade the server model by adding the second, third, or fourth processor book, which increases physical processors, memory, and enhanced self-timed interface (eSTI) high-speed I/O links. The concurrent book add (CBA) function allows new physical hardware to be integrated into the system without affecting ongoing customer workload. This capability is an extension of the concurrent permanent upgrade function that was introduced in previous-generation servers to add the new processors, memory resources, and high-speed I/O links seamlessly to the system.

Concurrent driver upgrade

The concurrent driver upgrade (CDU) feature was developed so that customers can have the ability to upgrade System z firmware and add new System z “firmware only” features without the need for planned downtime. With CDU functionality the System z firmware engineering change (EC) driver can be upgraded to the next EC level without any impact to the System z workload during the upgrade.

LIC controlled number and type of processors, and memory size

Current processors, memory, and input/output (I/O) cards are produced with very dense physical packaging. The granularity of this dense hardware packaging is provided under the control of System z Licensed Internal Code (LIC). Dormant hardware entities are reserved in this very dense physical packaging for capacity upgrades or self-healing. Concurrent capacity upgrade functions enable dormant hardware, such as memory or processors, to be added to the System z, which creates a new upgraded configuration. Alternatively, in case of hardware failure, healthy dormant hardware can be enabled, without disruption, to replace failing hardware.

I/O configuration data set support

An I/O configuration data set support (IOCDs) is used during a power-on reset to define the System z input/output configuration to the channel subsystem of the CPC. The I/O configuration is the set of all I/O devices, control units, and channel paths that are available to the CPC. By assigning control units, devices, and I/O paths to LPARs, the IOCDs defines access rights for these entities. System z firmware polices these access rights based on the

¹ See IBM Journal of Research Volume 46 *Flexible configuration and concurrent upgrade for the IBM eServer™ z900* by J. Probst, B. D. Valentine, C. Axnix, and K. Kuehl.

definitions in the IOCDS, which ensures that no program can access a particular I/O device unless the IOCDS grants access to the LPAR in which the program is running.

Miscellaneous equipment specification support

The customer can order System z system or capacity upgrades, which are called miscellaneous equipment specifications (MES) to the System z. The MES can be an upgrade to hardware, such as Processor Units (PUs), memory, or I/O, or the MES can be Licensed Internal Code Control Code. The MES can be delivered physically by installing hardware or by LICCC delivered by CIU. The HMC/SE is used to direct the Customer Engineer or Service Engineer through the process of the upgrade to the System z.

The Support Element provides step-by-step instructions for all PU and memory hardware upgrades through HTML presentation of text, graphics, and video. These instructions are consistent with the hardware repair instructions. Figure 2-5 is an example of an SE's MES panel.

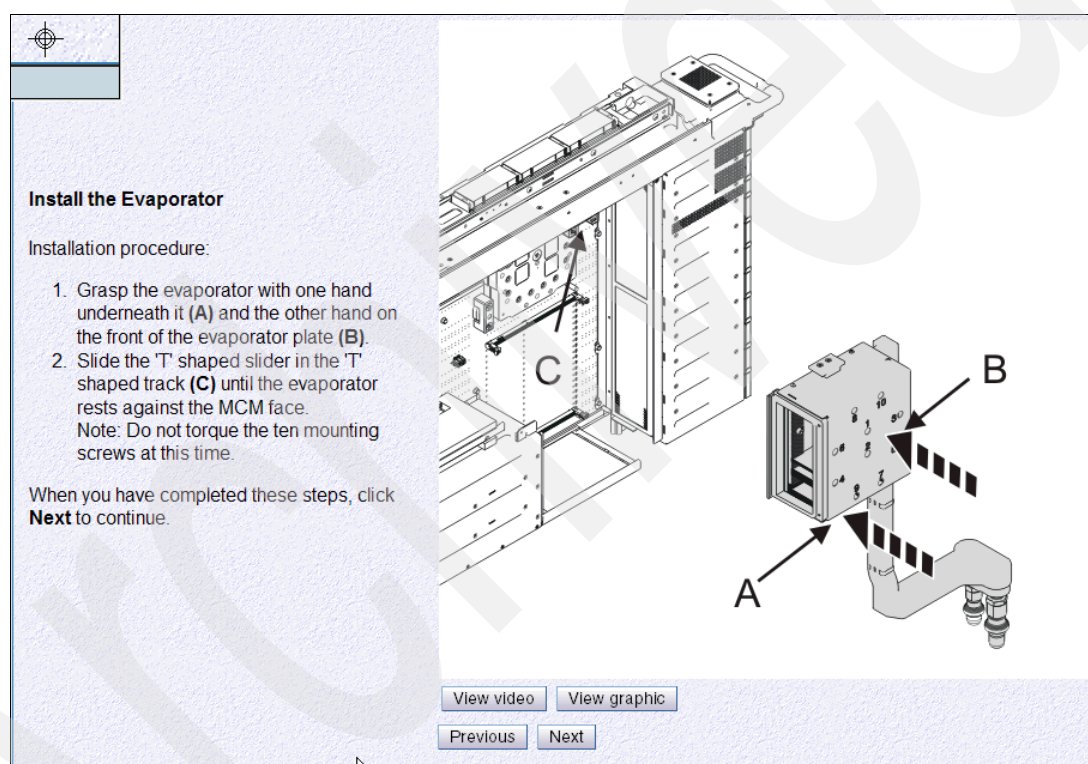


Figure 2-5 MES example

There are two key capabilities that the MES' guided procedures provide:

- ▶ The SE determines the maximum amount of power that can be consumed by the new configuration and compares it to the existing configuration. Instructions are provided to add additional power components that are required to support the new configuration. The appropriate procedure is launched to add all of the necessary hardware.
- ▶ When a PU or memory MES requires resources to be removed from the current configuration to complete the upgrade, the SE attempts to perform the upgrade concurrently by determining the workload that is being performed by the resource that must be removed and locating spare or unused PUs and memory to move the workload to.

2.4.3 HMC and SE performance management functions

Performance management is a common day-to-day factor in HMC usage. In this section, we discuss the performance management functions of the HMC/SE.

System activity display

The system activity display (SAD) shows the system activity for CPCs or a group of CPCs. System activity includes the channel activity and physical processing activity that were defined in the system activity profiles that are stored in the selected CPC. With SAD you can create a list of channels that you want to view and monitor.

The first display, shown in Figure 2-6, shows a summary of channel activity and physical processing activity for the selected CPC based on the SAD profile that the System Operator set up.

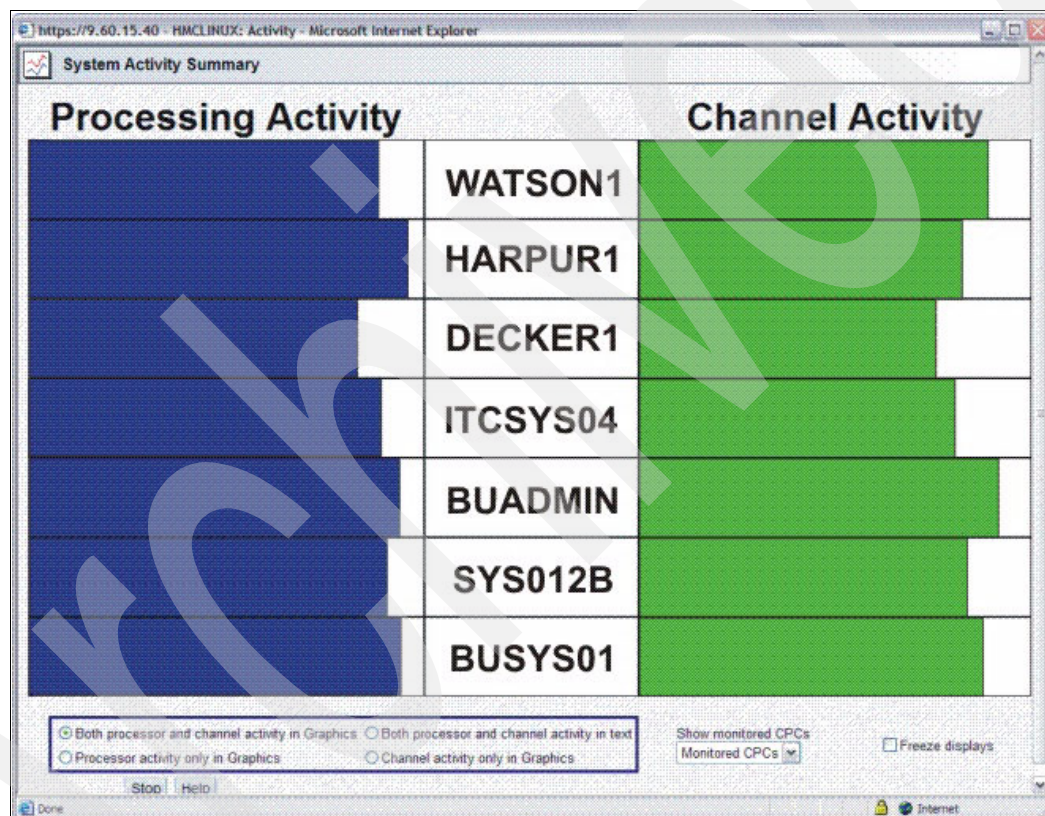


Figure 2-6 Summary Panel: System Activity Display

The processing activity and channel activity that are shown in the detailed panel of Figure 2-7 on page 37 was summarized by SAD to produce the overall processing utilization summary panel.

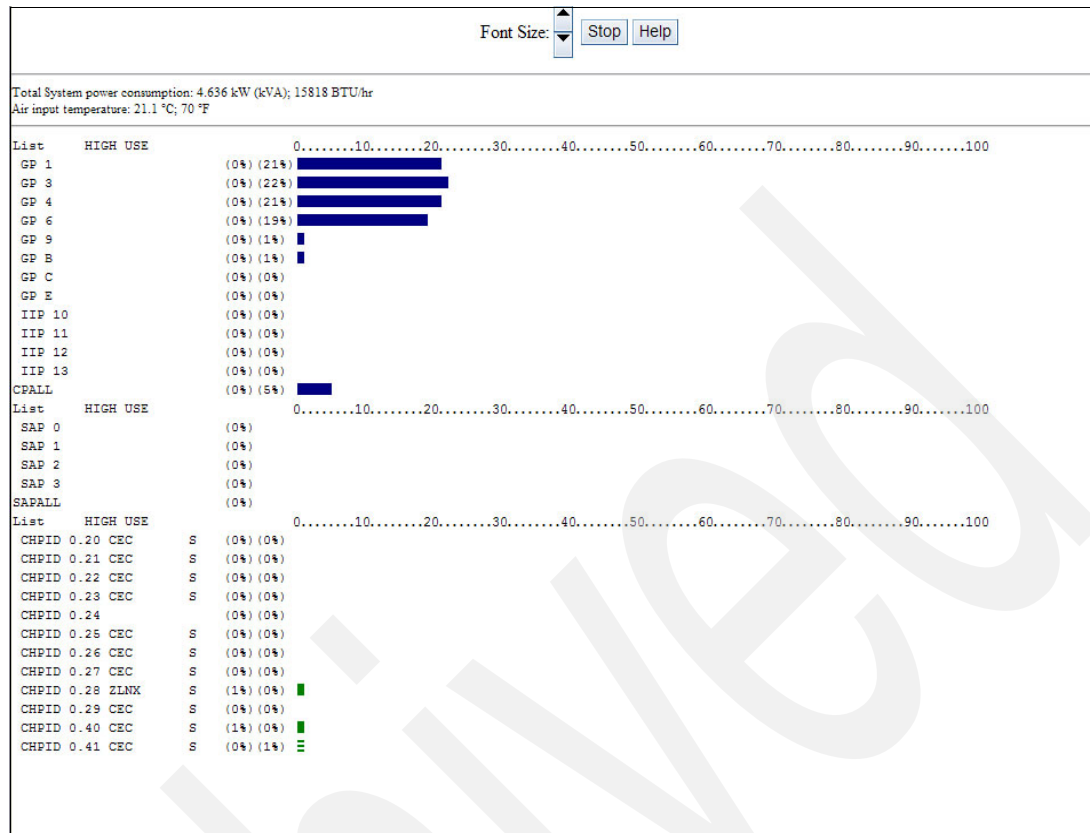


Figure 2-7 Detailed performance data for a selected CPC.

In Figure 2-7, the power consumption and input temperature are shown at the top. Below the power/temperature information is where up to 50 lines of detailed information can be displayed. In the DEFAULT profile shown, a list of the 10 most-used processors (or blank lines if the system has fewer than ten processors), followed by one line (CPALL) where the activity for all processors are averaged together. Then there is a list of the four most-used SAPs, followed by the SAPALL line, which averages the activity for all SAPs together. Finally, the profile uses the rest of the 50 available lines to show the most-used channels.

System Activity Profiles can be defined by the System Operator to save performance displays that are important to maintaining System z throughput. SAD supports multiple profiles, so the administrator can monitor resource activity customized for their systems metrics. SAD Profiles can monitor: CP utilization (absolute or by LPAR partition, all or by PSW key), I/O Channel utilization (absolute or by LPAR partition), and ICF/IFL/zIIP/zAAP/SAP utilization.

Dynamically change LPAR weights

An IBM System z can be divided into up to 60 logical partitions. Each LPAR has its own allocation of system resources (memory, processor, and channels). Resources can be dedicated to an LPAR or shared among LPARs. An LPAR is granted control of processors based on time slices. Each LPAR gets time slices based on the weighting factor for the LPAR. If only one LPAR needs resources and the other two LPARs are idle, the weights will not be factored. However, if one of the idle LPARs becomes active and starts competing for resources, then the processing power is divided between the two LPARs based on the relative weights.

The Customize Image Profile panel on the HMC is used to customize a profile (establish the configuration) for one (or all) of your LPARs. Changing the profile does not affect an LPAR

that is already running. The new profile is used as the configuration at the next IPL of the LPAR.

Emergency backup processing capacity

Capacity backup (CBU) provides reserved emergency backup processor capacity for unplanned situations where customers lost capacity in another part of their establishment and want to recover by adding the reserved capacity on a designated z990 server and moving the workload to another System z. CBU is the quick, temporary activation of Central Processors (CPs) in the face of a loss of customer processing capacity due to an emergency or disaster/recovery situation.

Send home performance and availability data

The HMC also provides a mechanism for the System Operator to transmit the performance and availability data of the System z to the IBM Product Support Center (PSC). This data is analyzed by Service Support Specialists and is used to ensure a high level of availability, for example, the System z is designed with all hardware resources error checked and redundant so that a single hardware error does not bring the system down. When the system detects a problem and dynamically reconfigures to avoid it, the customer's workload continues to execute. The IBM PSC reviews the system's performance and availability data to identify the use of redundant hardware and dispatch the appropriate level of service to bring the system back to full redundancy, as appropriate.

Capacity on demand

Capacity on demand (CIU, CUoD, OOCOD, eBoD, and so on) offers the flexibility to rapidly increase or decrease your computing capability as workload or business requirements change, which can mean a permanent capacity increase for planned growth or a temporary capacity increase for seasonal or unpredictable peak periods. Capacity on demand functions provide the capability to quickly and non-disruptively activate extra processor, memory, or I/O capacity that is built directly into the System z products. Available are Capacity Upgrade on Demand (CUoD) for a permanent, nondisruptive increase of processing capability, On/Off Capacity on Demand (On/Off CoD) for a temporary capacity increase that lets you react to fluctuating workloads whenever you want, and capacity backup for continued operation using interim capacity in a short term emergency situation. Other similar features and programs that are offered are Concurrent Memory Upgrade, Plan Ahead, and Customer Initiated Upgrade (CIU).

2.4.4 HMC and SE operations management functions

The HMC provides IT professionals with the tools that they need to better coordinate and manage all of their virtual and physical resources in the data center. The cost of operating the IT infrastructure has become the largest and fastest-growing component of overall IT spending for many organizations. Virtualization helps to address this cost through the consolidation of physical resources; however, virtualization also adds complexity to the system by creating a sharp increase in the system's capacity and the number of managed virtual resources. The HMC and SE provide the advanced capabilities and tools for managing both the physical and virtual systems across multiple architectures and environments. The HMC and SE provide a unified approach to platform management, which is designed to lower IT operational costs and increase operator and IT staff productivity.

With the HMC and SE, a System Operator can perform logical partitioning functions, service functions, and various system management functions using either the Classic HMC user interface or the new Tree based user interface. In addition, the HMC provides a Web-browser

based user interface for remote operations and a comprehensive set of APIs for advanced automation of data center systems.

The following HMC and SE components provide the key day-to-day operational capabilities that allow the System z IT staff to efficiently and productively manage a large enterprise data center. We review many of these functions in more detail later in this book and use them in the laboratory exercises that each student must complete through out this course:

Activate/deactivate This task controls starting up the system including power-on reset, partition activation, and initial program load of your operating system software. Activate is your primary function for CPC or CPC image start up. Activate senses the status of the object, then performs only those actions necessary to get the object to an operational state. The system hardware can be powered on, and activated using activate/deactivate found in the Daily tasks and the Recovery tasks.

Activation profiles The operating settings for the activation are defined in the activation profile. The activation profile is a record, on the HMC and SE that specifies a possible configuration for a system or logical partition. It indicates the required and maximum amounts of specified system resources that are used by the system or logical partition.

POR During Power On Reset (POR) the entire system is initialized. POR incorporates all functions that are required to initialize and load all participating system components. It provides a defined state of the system, known as POR complete, which is the ability to load a System Control Program using the architected IPL process. POR can be initiated after the system was logically powered on. The top level control of all necessary tasks within POR is located on the SE. To accomplish each task, most system components, such as the FSP, CEC firmware, I/O firmware, hardware itself, are involved. The main tasks of POR are:

- Test and initialize all hardware elements in the system

- Transfer the CEC firmware from the SE to the CEC memory, and start its operation

- Transfer the channel firmware for each I/O card

- Trigger a system reset for the CEC and I/O firmware

- Transfer and start the LPAR hypervisor

- When POR completes, all configured LPARs are initialized and a System Control Program can be loaded into each LPAR partition

Load support Load causes a program to be read from a designated device and initiates running that program. If the CPC is operating in LPAR mode, the logical partition is the target of the load. If the CPC is operating in basic mode, the CPC is the target of the load. A load profile is needed to define the channel address of the device from which the operating system gets loaded, for example, a Hardware Management Console's CD/DVD device or an FTP server can be used as a load device for system software.

Set TOD The Set TOD option allows the user to manually change the HMC's TOD. In a typical installation, the HMC's time is automatically established to the CPC's Support Element. The specific CPC to use as a time source is set during HMC initial set up. Alternatively, the HMC can be set up to get its time from an external NTP server. You can set the HMC to use an external NTP server by using the Customize

Console Date/Time task, under Console Actions to set this up. Regardless of the time source (SE or NTP server), you must set the HMC's time zone using the Customize Console Date/Time task.

PSW restart

PSW restart performs a restart operation on the first available central processor(s) of the selected CPC images. A restart interruption stores the current program status word (PSW) at real address 8 and fetches a new PSW from real address 0 in main storage. PSW Restart can be used when the status of the selected object is Operating or Stopped.

Store status

Store status is a processor operation that stores the contents of a processor's registers, excluding the time-of-day (TOD) clock, in assigned storage locations. The contents of the following registers are stored by the store status operation: CPU timer, Clock comparator, Current program status word (PSW), Access registers 0-15, Prefix Register,; Floating point registers 0-6, General registers 0-15, and Control registers 0-15.

External interrupt

External interrupt is a processor operation that you can use to present an external interruption to a processor. If you have experience using other systems, you might have used an IRPT command or an IRPT key to interrupt a processor. You can use the Support Element workplace to interrupt any eligible processor. Eligible processors are: Physical processors that support the image of a central processor complex (CPC) and Logical processors that support the images of logical partitions that are activated in operating modes other than coupling facility mode.

System resets

System resets prepare a system or logical partition to be loaded with an operating system. On the Support Element workplace, images support operating systems, and images are your targets for resets, for example, a reset normal is one of several recovery tasks that you can use to attempt to recover from hardware or software errors. A reset normal is often effective but less disruptive than other tasks, which typically makes it the first task that is attempted to recover from errors when they occur.

Power controls

Power subsystem firmware is part of the Licensed Internal Code that enables the power subsystem hardware. The Power subsystem consists of Bulk Power Controller, DCAs, Regulators, Air Moving Devices (AMD), and other granular devices. The bulk power controller (BPC) has its own service processor. The power firmware not only has the code load for the BPC service processor itself, but it also has the code for the distributed converter assemblies (DCA), bulk power regulators (BPRs), and fans. The BPC service processor code load also has the firmware for the cluster switches that might be installed in the frame. In the same way that the Central Electronics Complex (CEC) has dual service processors, the power subsystem has dual BPCs. Both are updated when firmware changes are made using the Licensed Internal Code Updates section of the HMC. You must use an HMC to communicate, monitor, manage, control, and update the power subsystem.

Automatic activation

When automatic activation is enabled and a utility power failure occurs, the CPC is activated automatically when the power is restored. The CPC is activated using the same reset profile used most recently to activate the CPC before the power outage. When automatic activation is disabled (default setting) and a utility power failure occurs, the CPC power remains off when the power is restored. You can

activate the CPC manually at any time after the utility power is restored.

Start/Stop

Start/Stop are processor operations that you can use to control whether processors can process instructions. If you have experience using other systems, you might have used START and STOP commands or Start and Stop keys to start and stop processors. On the Support Element workplace, images are supported by physical processors or logical processors. You can use the Support Element workplace to start and stop any eligible processor. Eligible processors include: Physical processors that support the image of a central processor complex. Logical processors support logical partitions that are activated in operating modes other than coupling facility mode.

Stop on condition(s) The processing and input/output (I/O) activity of central processors (CPs) is reflected in the contents of main storage, the status of I/O devices, and the contents of program status word (PSW). CP activity is indicated by the conditions of main storage, I/O devices, and the PSW. Monitoring these conditions provides another means for monitoring and controlling CP activity. By setting an address match or event that identifies the specific condition to watch for, all CPs are automatically stopped when the actual condition of main storage, I/O devices, or the PSW matches the condition you set.

Operating system

Console Integration, through the Operation System Messages task, gives the operating system the ability to write messages to an operator through the SE's and HMC's console panel and accept commands from the operator. This communication path can be used during system initialization, prior to achieving network communications capability, and it can be used after the network communications are established as an alternate path from the normal network path. Also, the operating system can use it at any time.

The Integrated 3270 Console is a task that provides a 3270 console to be used with a host operating system without the need for any special connectivity or additional hardware, such as control units or network connections. This task is beneficial to customers of z/VM because the Integrated 3270 console can do the job of the 2074 console. Similar to the Integrated 3270 Console, the Integrated ASCII Console is a task that provides an ASCII console to be used with a host operating system without the need for any special connectivity or additional hardware, such as control units or network connections. This task is beneficial to customers with Linux running in a logical partition or on a z System. The Integrated 3270 Console and the Integrated ASCII Console use the already existing network connection between the HMC and the SE and the SCLP interface between the SE and the CPC to connect to the host operating system.

In addition, the HMC also provides the System Operator with the ability to access the ESCON Director Console, the Sysplex Timer® Console, the 3270 console support, the ESCON Director (9032-3, 9033-4), and the Sysplex Timer (9037-2) Consoles from the HMC.

Trace

One important tool for debugging firmware problems is the tracing of program steps. On the HMC & Support Element, this is done through a central wrap around buffer in memory. There is a common routine that any application can call to create a trace entry. The HMC/SE coding conventions require tracing of the entry and exit of every major function. Additionally, a trace can be taken at any critical point in the

code that the developer deems is helpful in debugging a code problem. Component tracing is introduced, which defines unique trace areas for registered components and assures that no trace entries are *wrapped out* by other components.

Display/alter

Display/alter provides a wide-range of display and alter facilities that are used to access memory locations, registers, and many kinds of special data areas that reflect the current state of the System z processor and attached I/O hardware adapters. Data that is being displayed can be dynamically overwritten to correct an undesired state and to continue the debug process with the current code load, for example:

i390 Storage

Provides the display/alter capability for the entire XMP storage range (including customer absolute, i390, and millicode storage). Addressing begins with an absolute address zero.

i390 PSW

Provides the display/alter capability for the i390 PSW.

i390 general registers

Provides the display/alter capability for the i390 general purpose registers (GPRs).

i390 control spaces

Provides the display/alter capability for the architected and implementation-dependent i390 control spaces as they are defined in the i390 kernel functions and services and some selected i390 storage areas that are pointed to by it.

Display i390 common data areas

Displays the i390 common data areas. The i390 common data areas are important i390 storage control structures. Pointers and the corresponding description to those areas are displayed in an initial panel.

Display i390 module load list

Shows the list of all i390 modules (name and starting address) that were loaded into i390 storage by the TOC loader during IML.

I/O support

Essential to the ever increasing processing capacity of the System z is the corresponding need for significant increases in total I/O scalability and connectivity. The System z I/O subsystem provides unparalleled I/O capacity which gives the System z a key competitive advantage in I/O bandwidth. Increased System z I/O capacity is provided by increasing the number of physical I/O channels that can be configured to the system and by restructuring the physical channel subsystem (CSS) into logically distinct channel subsystems. This restructuring is commonly called the multiple-channel subsystem (MCSS) facility. The HMC and SE provide the System Operator with the controls necessary for I/O management and control. The HMC I/O System Operator functions include:

Advanced facilities

Displays a list of actions that you can take for the selected channel path. The list of tasks are:

View code level	Display or alter MAC address
Card Trace/Log/ Dump Facilities	Enable/disable ports

Display or alter trace mask	Run port diagnostics
Read trace buffer	Set card mode
Read log buffer	Display client connections
Read MIB buffer	Display active sessions configuration
OSA LAN Analyzer...	Display active server configuration
Read OSA LAN Analyzer buffer...	Panel configuration options
OSA-Express Host Network Traffic Analyzer Authorization...	Manual configuration options
Card specific advanced facilities	Activate configuration
Query port status	Display activate configuration errors
View port parameters	Debug utilities
View code level	Reset to default

Config on/off

Configure on and configure off are CHPID, OSA, Crypto, and so on, operations that you can use to control whether the channel paths are online or on standby in the active input/output (I/O) configuration.

A channel path, OSA, crypto, and so on is online while configured on. It is in the active I/O configuration and it can be used.

A channel path, OSA, Crypto, and so on is on standby while configured off. It is in the active I/O configuration but it cannot be used until it is configured on.

Service on/off

Channel operations that you can use to control whether channels that are identified with physical channel identifiers (PCHIDs) are on standby in or reserved from, the active input/output (I/O) configuration.

A channel is on standby while service is set off. It is in the active I/O configuration but it cannot be used until it is configured on. It remains in the active I/O configuration until service is set on.

A channel is reserved while service is on. It is not in the active I/O configuration and cannot be used. It remains out of the active I/O configuration until service is set off.

Setting service on for a channel removes it from the active I/O configuration, which allows running of diagnostic tests on the channel without disturbing other channels being used by the system. Setting service on for a channel can be used also to remove failing channels from the I/O configuration so subsequent power-on resets do not attempt to initialize the failing channels.

Scheduled operations

Scheduled operations enable the System Operator to schedule an operation to run at a later time, define operations to repeat at regular intervals, delete a previously scheduled operation, view details for a currently scheduled operation, view scheduled operations within a specified time range, and sort scheduled operations by date, operation, or console.

The System Operator can schedule the times and dates for automatic Licensed Internal Code updates and backup of critical hard disk data for the Hardware Management Console. Using the Customize Scheduled Operations task displays the following information for each operation: The processor that is the object of the operation, the scheduled date and time, the operation, and the number of remaining repetitions.

An operation can be scheduled to occur one time or it can be scheduled to be repeated. The System Operator enters the time and date that you want the operation to occur. If the operation is scheduled to be repeated, you are asked to select: The day or days of the week that you want the operation to occur, the interval or time between each occurrence, and the total number of repetitions.

STP

Server time protocol (STP) is a time synchronization architecture that is designed to provide the capability for multiple servers (CPCs) to maintain time synchronization with each other and to form a Coordinated Timing Network (CTN). STP is designed for servers (CPCs) that are configured to be in a Parallel Sysplex® or a sysplex (without a Coupling Facility), and servers (CPCs) that are not in a sysplex, but that need to be time synchronized. STP is designed as a message-based protocol that allows time keeping information to be sent between servers (CPCs) and Coupling Facilities (CFs) over InterSystem Channel-3 (ISC-3) links configured in peer mode, Integrated Cluster Bus-3 (ICB-3) links, or Integrated Cluster Bus-4 (ICB-4) links.

ETR

The IBM External Time Reference (ETR) architecture facilitates the synchronization of server time-of-day (TOD) clocks to ensure consistent time stamp data across multiple servers and operating systems. The ETR architecture provides a means of synchronizing TOD clocks in different servers with a centralized time reference, which in turn can be set accurately on the basis of an international time standard (External Time Source). The architecture defines a time-signal protocol and a distribution network (called the ETR network) that permits accurate setting, maintenance, and consistency of TOD clocks. The Sysplex Timer Model 2 (9037-2) has been a key element in the Parallel Sysplex environment since its introduction. It ensures that multiple z/OS and OS/390® systems can appear as a single system image, synchronizing the Time-of-Day (TOD) clocks of all of the attached systems to ensure a consistent time stamp. When multiple systems update the same database, all updates are time-stamped in sequence. The ETR function is now standard on System z. There are two ETR cards located in the CEC cage, each with a fiber optic connection to provide the capability to attach to the Sysplex Timers.

Industry standard SNMP operations APIs

As an alternative to manual operations, the HMC supports a programmable interface, or API. The automated interface allows a program to monitor and control the hardware components of the system in the same way that a human can monitor and control the system. The HMC APIs provide monitoring and control functions through TCP/IP SNMP to an HMC. These APIs provide the ability to get and set a managed object's attributes, issue commands, receive asynchronous notifications, and generate SNMP traps. The automated interfaces are used by various automation products, including Tivoli System Automation for z/OS - Processor Operations.

Grouping controls

Provides a mechanism for System Operators to group system resources together in a single view. Groups can be nested to create custom topologies of system resources. Custom Groups Objects include the predefined groups *All Images* and *All Objects* and any user-defined groups that are created using the Grouping task under the Daily category in the tasks pad. The status of a group of objects is summarized and displayed as a background color of the group icon.

2.4.5 System z serviceability functions

The System z technologies have evolved over time, and the integrated hardware, firmware, operating system and middleware elements are designed to tightly work together to provide an application environment with world-class levels of high availability and disaster recovery. The reliability, availability, and serviceability goal for an IBM System z is to provide 24-hour-per-day, 365-day-per-year service with reduced system downtime. The continuous reliable operation rate is improved with every new System z release by using error prevention, error detection, error recovery, and other methods that contribute to avoiding unplanned interruptions. To accomplish this goal, the System z product line offers layer upon layer of fault tolerance and error checking. These *self-management* technologies enable the server to protect itself, to detect and recover from errors, to change and configure itself, and to optimize itself, in the presence of problems and changes, for maximum performance with minimal outside intervention. Even if a failure does occur, the built-in redundancy on System z servers can reroute work from failing components to operational ones in an effort to prevent the end-user service from being interrupted. The failed components can be removed and replaced while the application is still active, which allows service to continue. MTBF of unscheduled uninterrupted repair actions (UIRA) for System z servers is now measured in decades.

The IBM Customer Engineers, also known as IBM Service Representatives, use the HMC and SE to install and service the System z hardware. They use the SE for MES Upgrades, to perform a repair action, and for CE Education.

Problem management

The HMC and SE offer tools and automatic mechanisms for servicing the system with minimal disruption to the customer's operation. Problem management functions and tasks detect failures within the system and repair them as quickly as possible. These functions include automatic problem analysis in conjunction with First Failure Data Capture (FFDC), automatic problem reporting to IBM, followed by a dispatch of an IBM Customer Engineer (CE) to the customer's system with replacement parts in hand.

Automatic redundant backup support allows for a concurrent, transparent repair. Firmware updates can be scheduled as an automatic operation.

Problem management has wide-ranging aspects that include these elements:

- ▶ Automatic Problem Analysis: This function is Expert System based. It correlates multiple events and multiple systems, and then generates an automatic *call home* and dispatch of the Customer Engineer. It also notifies of the System Operator using an alert, e-mail, or pager.
- ▶ Redundant backups: Most System z hardware subsystems have redundant backups that allow for concurrent, transparent repair, for example, CP sparing, Hot CEC Plug, N+1 power, memory ECC, spare chips, ALT SE, peer HMC design, CHPID mapping, and alternate channel paths.
- ▶ CE Education: With the MTBF of the System z being so long, the HMC provides *just in time education* for the Customer Engineer or System Operator to refresh their knowledge of the repair process when a problem does occur. This education is delivered on-site using the Web.
- ▶ Service History: Maintains a detailed chronology of all problems and fixes that were applied, which gives the customer an accurate history for problem post-mortem analysis.
- ▶ I/O Problem determination tools.
- ▶ Traces: Such as I/O trace and program trace allow accurate problem debug and analysis.
- ▶ Weekly transmission of RAS data to IBM for expert review of system events to ensure that the system is running at peak performance.
- ▶ Remote IBM Product Engineer access to guide the System Operator through tough situations.
- ▶ PE only tools: Such as internal traces, dumps and logs, scan ring facility access, array access, and special diagnostic programs.

Change management

Each HMC and each SE has Licensed Internal Code that is subject to periodic updates from IBM. On systems with multiple HMCs, one of the HMCs is configured as a LIC Change Management focal point. The HMC that is configured as the LIC Change Management focal point monitors all firmware levels, updates, backups, retrieval, and concurrent application of updates without any customer effort or involvement. The HMC then automatically retrieves and distributes Licensed Internal Code updates for the Hardware Management Consoles remotely from IBM and retrieves and distributes SE LIC updates to all of the SEs of all of the CPCs that are configured to the Hardware Management.

Concurrent Driver Upgrade (CDU): The continuous reliable operation rate is improved with every new System z release by using error prevention, error detection, error recovery, and other methods that contribute to avoiding unplanned interruptions. Planned downtime—the time that is needed to upgrade a System z to the next firmware driver for new functions—required a system reset to take effect. The concurrent driver upgrade feature is provided so that customers can add new functions without downtime. It is now possible to upgrade the System z firmware engineering change (EC) driver to the next EC level without any system availability impact during the upgrade.

Repair and verify (R/V)

The repair and verify (R/V) directs the Service Representative through the exchange of the failing parts and verifies that the system is operational again. To minimize the impact of the repair to the customer approximately 95% of the FRUs in the system can be exchanged concurrent with System z R/V. R/V is a Directed Maintenance Package. The SE provides specific instructions to perform every procedure that is needed to service the System z hardware. Directed Repair leaves minimal decisions for the servicer to make and the CE is expected to perform each and every step as directed by the SE Maintenance Package. The

reason for this degree of control is that in a large System z enterprise, replacing a part that is concurrent with system operation is very complex. The R/V procedures are tested and proven to manage the complexity without interrupting the System z workload. A typical processor requires about 400 to 500 Removal/Replacement panels. About 75 to 100 panels are needed for the isolation procedures. Concurrent MES uses many of the Removal/Replacement panels plus about 150 to 175 additional, unique panels.

Emergency backup processing capacity

Capacity backup (CBU) provides reserved emergency backup processor capacity for unplanned situations where customers lose capacity in another part of their establishment and want to recover by adding the reserved capacity on a designated z990 server and moving the workload to another System z. CBU is the quick, temporary activation of Central Processors (CPs) in the face of a loss of customer processing capacity that is due to an emergency or disaster/recovery situation.

Send home performance and availability data

The HMC also provides a mechanism for the System Operator to transmit the performance and availability data of the System z to the IBM Product Support Center (PSC). This data is analyzed by Service Support Specialists and is used to ensure a high level of availability, for example, the System z is designed with all of the hardware resources error checked and redundant so that a single hardware error does not bring the system down. When the system detects a problem and dynamically reconfigures to avoid it, the customers' workload continues to execute. The IBM PSC reviews the system's performance and availability data to identify the use of redundant hardware and to dispatch the appropriate level of service to bring the system back to full redundancy, as appropriate.

2.5 Questions and discussions

1. How does the HMC participate in controlling operating systems, such as z/OS, z/VM, or Linux for System z?
2. Who should have access to the HMC?
3. What HMC functions are important to Application Programmers?
4. What is an HMC, in hardware terms? What programs or operating systems are involved?
5. What are the key differences between an HMC and a SE?

Archived

User experience

In this chapter, we provide an overview of how a user interacts with an HMC. We discuss the following topics:

- ▶ Overview and background
- ▶ User interface (UI) design:
 - Requirements
 - Personas/user stories and stakeholders
 - Feedback and iteratively incorporating it into the product
- ▶ Customer investment (user interface transition and migration):
 - Classic style
 - Tree style
 - The user interface framework
- ▶ Consistent user experience:
 - Common UI widgets
 - Panel and task consistency
- ▶ Technologies:
 - Extensibility through XML based UI binding
 - Plug-ins
 - UI technology change tolerance
 - Server push (real-time status)
 - Disconnect (roaming users)
- ▶ User defined resource relationships
- ▶ Systems management launch in context:
 - Upward management APIs
 - UI task integration: launch in context
 - Motivation

Objectives

After reading this chapter, you will be able to:

- ▶ Describe classic style and tree style user interface
- ▶ Navigate and use both interfaces
- ▶ Describe the technology that powers the user interface
- ▶ Understand the framework that was used to develop the new interface

3.1 HMC user interface overview and background

Since the 1980s mainframe hardware was managed and controlled by what is now known as the classic style user interface. This style was designed based on the popular operating system of the time, OS/2, which used a folder/group-based user interface. The newer user interface, known as *tree style* or *tree UI*, was developed with a look and feel that is consistent with modern UI paradigms. The classic style is still maintained and supported to protect customer investment and to provide backward compatibility, which is a key mainframe differentiator.

In this chapter, we present the requirements for a new user interface and specifically what was required of the tree UI. We explore how these requirements, along with Personas and User stories, influenced the design of the HMC's new user interface. We then see how customer feedback and iterative design was employed to help develop a modern user interface.

When introducing a new user interface, it is important to consider customer investment in the existing user interface. That investment can include time spent training employees to use the existing user interface and creating documentation that details procedures in previous implementations of a product. To protect this customer investment, the HMC allows the use of both classic style and tree style user interfaces. We look into the functionality and components of both interfaces and explore the framework that powers both user interfaces.

Providing a consistent user experience is a necessary feature for any user interface. Maintaining a similar look and feel throughout the interface makes the customer feel more comfortable operating the product. All IBM management platforms follow a set internal user interface guidelines to maintain consistency across IBM products, allowing customers of one management platform to more easily operate another. The HMC follows these same guidelines and further maintains a consistent look and feel by using a common set of graphic widgets between the new user interface and the HMC's tasks. These widgets can be launched from either user interface.

We also explore the technology behind the tree style user interface by discussing how and why the new user interface is bound to XML. We also explain *server push* implementation, which allows the new user interface to refresh only the display portions that need updating and are visible to the user. We cover how *Disconnect*, sometimes referred to as roaming users, allows operators to leave their session running and reconnect to it from another location.

User-defined resource relationships (custom groups) allow operators to create their own groups of resources on the HMC, which allows users to group desired systems together and operate on the entire group, rather than one system at a time.

Lastly, we discuss the use of *upward management* APIs, which allows the HMC to be managed from other platforms. Performing HMC tasks from another platform is known as *Launch in Context*. This design capability reduces redundant development because it allows

one management platform to use the functionality of an already-written platform. There are also other design benefits that we discuss.

3.2 HMC user interface design

There were many goals when designing the new user interface. Automating complex functions, single point-of-control, single system image (for example, *activate* 20 different systems with one action), common look and feel across platforms, high percentage of day-to-day tasks on the front panel, and minimize operator and system programmer opportunities for mistakes were the requirements for the new user interface to be effective. The challenge was implementing all of these features in an effective way to minimize operator errors and provide an intuitive interface for the HMC/SE.

A modern user interface has various requirements to be useful and functional. Consistent user experience, task simplification, and interface usability are key design requirements. We must recognize that the time and money a customer spends on training employees and documenting procedures is a significant investment. Any new user interface should protect the customer's investment in employee skills with existing procedures and processes, while also providing new features that enhance the user's experience, for example, System z customers invested in Operator and System Programmer training for controlling System z hardware. They created specific documented procedures for controlling their System z hardware for their environment. These documents often reference the UI (for instance, "When the Views Area background changes to red, click the HMC icon in the Work Area..."). A new user interface design must protect this investment by providing a smooth transition to the new user interface. Consideration must also be given to allowing the customer to choose when they migrate to the new interface.

The design of the new interface leverages modern familiar user interface paradigms to increase the overall usability of the system. The new user interface was designed with a hierarchical layout and navigation and context-based tasks and menus. Providing real-time status was also kept in mind when creating the interface.

Simplification and usability were also taken into account for the new user interface design. Some goals of the new UI included improving task discovery, using consistent task placement and organization, displaying resource properties in main views, reducing task depth, and simplifying hard-to-use areas.

A consistent interface allows for less time spent learning different systems and increases productivity for customers. A major emphasis of the new user interface design is consistent user experience. The goal is to create an interface that is consistent across the HMC family (HMC, SE, TKE, and POWER® HMC) and other IBM Web-based user interfaces.

New technologies were required for the new interface because of its unique aspects. Because the HMC must provide real-time status of all systems that it monitors, the HMC UI must be refreshed to display information that is up-to-date. Because the HMC is a Web application, this requirement was met using Asynchronous JavaScript and XML (AJAX) and server push (dynamic updates driven independently of user action). Roaming user technology gives the customer the ability to perform tasks while disconnected from the HMC. The user can log back in at another console later and access the HMC as he left it in his previous session, which allows users to roam between Hardware Management Consoles.

3.2.1 Personas, user stories, and stakeholders

A well-designed interface can increase user productivity and decrease frustration. Studying how a user interacts with the interface helps to accomplish this goal. The HMC uses *personas* as contexts to organize feedback from users about their interactions with the system.

A persona is a fictitious character that is created to represent a user of a product, for example, an Operator persona describes a user who performs basic system functions, such as status monitoring. A System Programmer persona performs more advanced functions, such as configuring console services. Another persona might be an IBM Product Engineer who requires more in-depth diagnostic capabilities. Personas give each type of user a face or personality with defined responsibilities and skills, which makes it easier for the Developers to communicate the target user's capabilities.

A *user story* is a short description of a user requirement for the system. An example might be an Operator user story that describes monitoring and reacting to status changes. A Product Engineer user story might describe viewing console logs as a result of a problem.

3.2.2 Initial design

Figure 3-1 illustrates the general development path of the new user interface. The GA term means General Availability, that is, the GA version is available to customers.

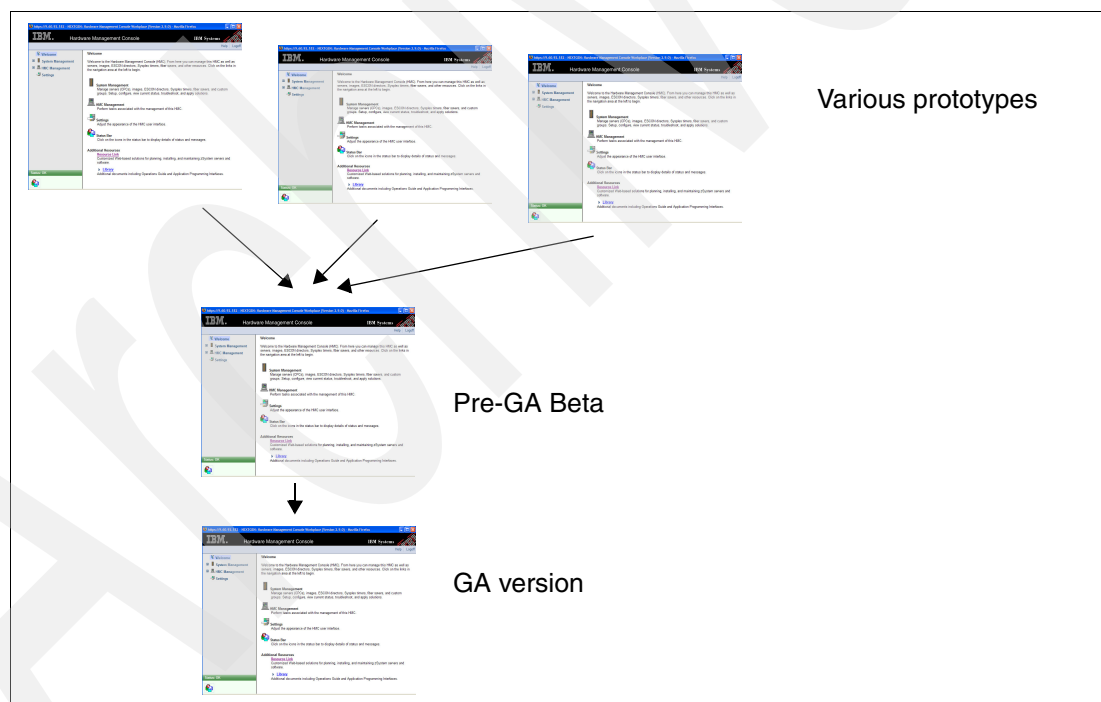


Figure 3-1 Development path

The tree style UI and other HMC features are designed based on an iterative, customer-focused approach. Customers include IBM internal users and external customers, from Operators to System Programmers and service personal. The HMC development process entails the following steps:

1. Create an initial design, and review it with customers.
2. Create a prototype, and present it to internal IBM customers for feedback.

3. Update the prototype, based on initial feedback, to create a beta version, which is released to selected external customers for testing and feedback.
4. Complete the design, perform final tests, and release the new functions for general availability.

This cycle continues for each new release. Customer trials of prototypes permit early hands-on experience while providing valuable feedback to guide the development process. Obtaining feedback and the iterative nature of the process helps to ensure that the final product matches customer needs.

This customer-focused design process continues throughout the life of the product. This process gives customers a voice in the development of the HMC through the HMC Partners program. The HMC Partners program provides opportunities for ongoing feedback. Site visits to customer data centers, user-centered design interviews, individual or group conference call presentations, and direct test drives of prototypes are all offered through the program, which increases customer satisfaction and helps the Developers to build a better product for the customer.

3.3 Customer investment: User interface transition/migration

Since the 1980s mainframe hardware was managed and controlled by what is now called the classic style user. The newer user interface, called tree style, was developed with a look and feel that younger employees were more used to. The classic style, shown in Figure 3-2 on page 54, was still kept to protect customer investment and to maintain backward compatibility.

3.3.1 Classic style

The Classic Style user interface, shown in Figure 3-2 on page 54, was the original interface style for the HMC. It is an object-orientated user interface where each component is a resource that has tasks that can be performed. Because many customers developed *run books* and other documentation based on the classic style, it is still the preferred choice for many long-time users of the HMC. Classic style supports a drag-and-drop technique, where the user can drag an object and drop it on a task to perform. An object does not have to be a single object; indeed, it can be a collection of managed resources or a group, which can contain one or more managed resources and other groups. The classic style interface has three major visual components:

- ▶ Views area
- ▶ Task area
- ▶ Work area

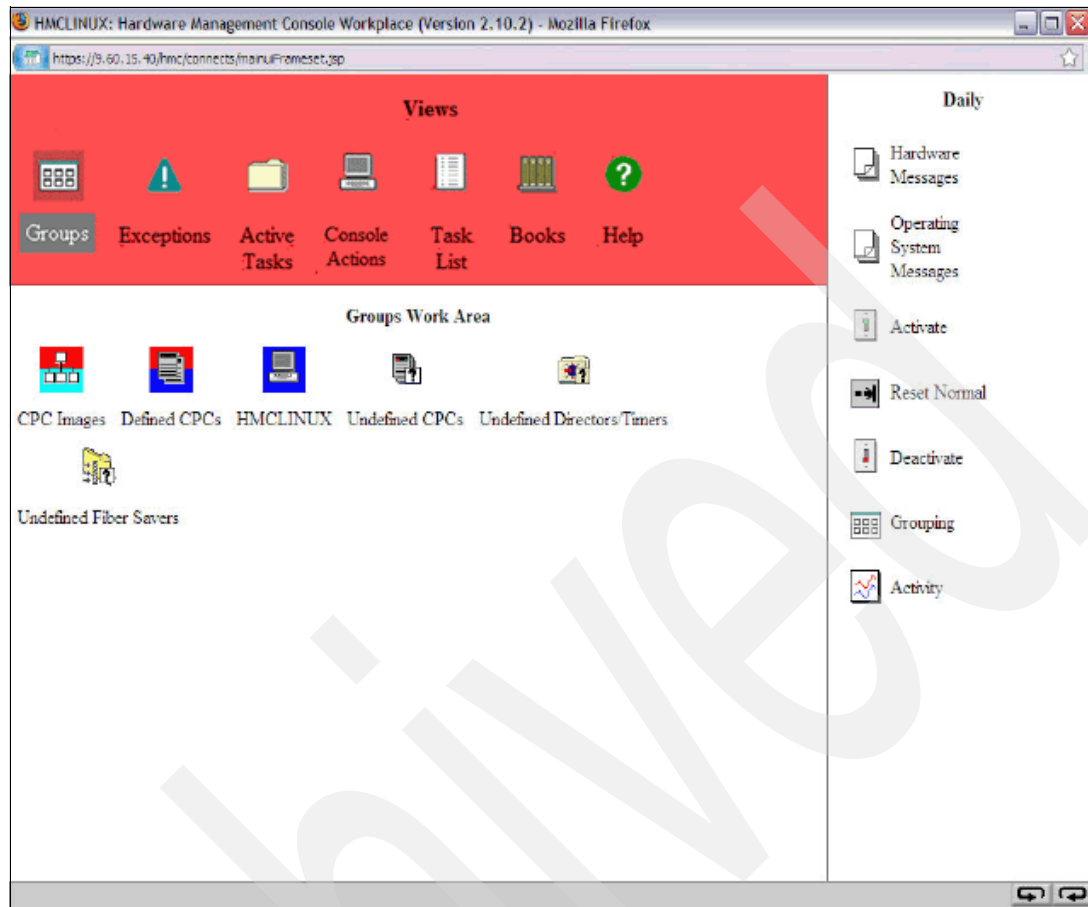


Figure 3-2 Classic style panel

The views area

The views area is located in the upper left portion of the HMC window. The views area provides a way for the user to monitor and manipulate their HMC from a single menu that is always shown. The views area is the main panel of the classic style, and it is from here that managed resources and tasks can be accessed on the HMC. By selecting groups or objects within them, the entire hierarchy of managed objects can then be accessed in the work area. Exceptions can be viewed in a similar way. Tasks can be found through the *console actions* or *task list* icons. The active tasks icon lets the user monitor their currently running tasks. The books and help selections are available for information and guidance.

The background color of the views area reflects the current status of all of the managed systems. When all managed objects are in an acceptable state, the background of the views menu is green. When one or more managed resources enter an unacceptable state, the background turns red. This color coding allows users to monitor the health of their systems from a distance while using the classic style.

The task area

The next major component of the classic style is the task area, which is located along the right side of the HMC window. It is from here that users perform tasks on an object or a group of objects. Users can select different tasks to be displayed in this area. The task area gives users a way to start tasks on selected objects.

The work area

The last area of the classic UI is the work area. The work area is located in the lower-left portion of the HMC window and is where the main manipulation of objects is performed. The contents of a views area *selection* are displayed in the work area. Resources can also be navigated through the work area, such as opening a group or object (such as a server), which displays the group's or object's children (partitions of the server) in the work area. Objects that are selected in the work area can be dragged and dropped onto a task in the task area to invoke the task on the selection. Tasks can also be launched against the selected objects using a context menu that is available when right-clicking the selection.

All three components come together to make up the classic interface. The classic UI continues to be a popular choice for System z customers.

3.3.2 Tree style

The newest user interface style is called tree style, as shown in Figure 3-3. This style provides a hierarchical view of system resources and tasks using drill-down and launch-in-context techniques to enable direct access to hardware resources and tasks management capabilities. The tree style provides customers with quick access to tasks and to details that are not available in the classic style UI. The major components of the tree style are:

- ▶ Banner
- ▶ Task bar
- ▶ Navigation panel
- ▶ Work pane (includes the Tasks Pad)
- ▶ Status bar



Figure 3-3 Tree style panel

The Banner and Task bar

The Banner and Task bar, shown in Figure 3-4, are located on top of the tree UI window. The banner identifies the product and logo. Below the banner is the task bar, which provides a single point to monitor the user's active tasks. When a task is launched, a button that represents the task is added to the task bar. Clicking the Task bar button brings the task window into focus. When a task ends, its task bar button is removed. The task bar also provides quick links to launch the user settings, logoff or disconnect tasks, and the tree UI help function.

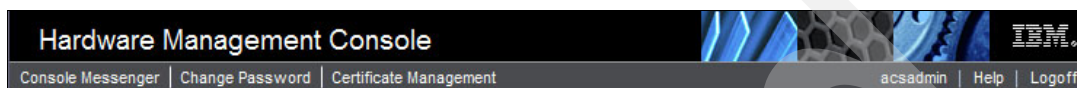


Figure 3-4 Banner and Task bar

The Navigation panel

The Navigation panel, shown in Figure 3-5, is located along the left side of the HMC window and includes the resource tree that manages the HMC and its managed systems. Each node in the navigation tree is a selectable object that can be expanded and has child nodes. The navigation tree displays the HMC's managed resources, unmanaged resources, and tasks in a hierarchical layout.

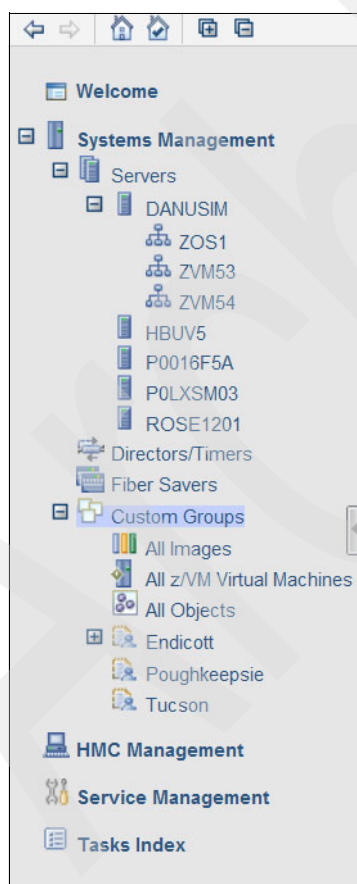


Figure 3-5 Navigation panel

The nodes of the Navigation panel

The systems management node is the root node for all managed and unmanaged resources. It contains the Servers, Directors/Timers, Fiber Savers, and Custom Groups containers. The nodes that are displayed are based on user access:

- ▶ The Servers node contains all servers that are managed by the HMC. Each managed server then contains its partitions, which might contain z/VM Virtual Machines.
- ▶ The Directors/Timers node contains all managed ESCON Director consoles and sysplex timer console.
- ▶ The Fiber Savers node holds the IBM 2029 Fiber Savers that the HMC manages.
- ▶ Custom Groups contain all predefined and user-defined object groups, All Images (which includes all partitions on the servers that the HMC manages), All Objects (which contains all managed objects), and All z/VM Virtual Machines (which contains all z/VM Virtual Machines running on managed partitions).
- ▶ Unmanaged Resources contains the Servers, Directors/Timers, and Fiber Savers that are not defined to the HMC, which an administrator can assign to the HMC.

The HMC Management node contains a list of tasks that are used for setting up the HMC, maintaining its internal code, and securing the HMC. The Service Management node contains a list of tasks that are used to service the HMC. Lastly, the Tasks Index node contains a table of all console actions and targeted tasks that the current user can perform.

The navigation toolbar, shown in the navigation tree of Figure 3-6 on page 58, provides many useful features for the navigation tree. The Back and Forward buttons allow the user to navigate through their navigation tree history. The Home button changes the navigation tree selection to the user's HMC homepage, which can be set using the Set Home button. The Expand All and Collapse All buttons let the user quickly expand or collapse all navigation tree nodes.

The Work pane

The Work pane contents are based on the current Navigation Panel or Status bar selection. The information that is displayed follows the tree style hierarchy, as each selection of an object displays any or all available children. The Work pane supports multiple resource views that display child objects in different displays, which allows for further customization. Managed resources can be displayed in table, tree, or topology views:

- ▶ Table view displays managed objects in separate homogenous tables. The tables can be sorted, filtered, and configured to provide customized data. Table view also allows users to customize the columns that display various details about each object.
- ▶ Tree view displays managed objects and their children in a tree table, where each object is represented by a row in the table, and a row can be expanded to show its children in the table.
- ▶ Topology view displays the selected object and its children in a topological map, where resources are represented as icons that can be expanded to show their children.

The Tasks Pad

Included in the Work pane is the Tasks Pad, shown in Figure 3-6 on page 58. The Tasks Pad is located at the bottom of the Work pane when a managed object is selected in the Navigation tree. The Tasks Pad shows tasks that can be performed on the resources that are selected in the Work Pane Table, Tree Table, or Topology view. If no resources are selected in the Work pane, tasks are listed for the current Navigation pane selection. The Tasks Pad gives a quick and easy way to manipulate multiple objects simultaneously and can be customized using its Settings menu.



Figure 3-6 Tasks panel (partial)

The Work pane also displays task lists for HMC and Service Management. Users can customize these tasks lists. Using the View icon in the upper-right corner of a task list, the current layout of the tasks can be changed. Tasks can be listed in alphabetical order or by category. The tasks can be displayed in three different styles: detail, icon, or tile:

- ▶ Detail displays the tasks in a table form with the task name on the right and a short description of the task on the left.
- ▶ Icon displays tasks with large icons and the tasks' names.
- ▶ Tile displays each task with a large icon, name, and a short description.

The Status bar

The Status bar, shown in Figure 3-7, is located in the lower-left portion of the HMC window. The Status bar provides visual indications of the status of the HMC and its managed resources. The color and title of the Status bar changes depending on the system status, for instance, when the HMC or one or more managed resources are in an unacceptable state the Status bar is red, and its title indicates an Error state. Like the classic UI view's area background coloring, with the Status bar users can monitor the health of their systems from a distance while logged in to the tree style UI.

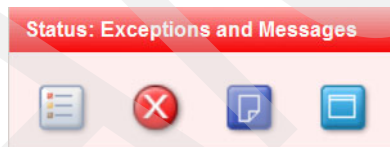


Figure 3-7 Status bar

Users can get additional detailed status information by clicking the Status bar buttons. Clicking the Status Overview button displays the Status Overview Page, which provides a detailed account of all managed resources with exceptions, hardware messages, and operating system messages. Users can then view all managed resources with exceptions, hardware messages, and operating system messages by clicking the respective button in the Status bar.

3.3.3 How: user interface framework

One of the features of the tree UI is the display of important managed objects properties in resource tables. However, properties that are defined and managed by the Managed Object Framework (MOFW) are often not human-readable or useful to the user interface. To provide a general and extensible interface for shuttling properties to user interfaces that are both displayable and useful, the UIProperties interface was created.

The UIProperties interface is defined by the User Interface Framework (UIFW) and is implemented by the Managed Object Framework. The reasoning behind this design is that the UIFW is placing the requirement on the MOFW that it provide any UI-useful properties in a displayable manner because these are object-specific and therefore beyond the scope of the UIFW, for instance, a server might have a property that contains its status. This status property can be stored in the server managed object as a integer value. The user interface cannot display this integer value in a Status column; instead, it relies on the server's UIProperties to convert that integer to a human-readable status message.

From this point forward, we distinguish between managed object (MO) properties and user interface (UI) properties. MO properties are those properties that are defined and supported in the MOFW by the managed object classes. UI properties are those properties that are defined and supported by the managed objects' UIProperties classes. Their relationship is not necessarily one-to-one, but simply that UI properties are generally built or converted from MO properties.

The UIProperties classes can be used by user interfaces for several things:

- ▶ To reference the UI properties that they want to display
- ▶ To retrieve values for objects and UI properties that they want to display
- ▶ To retrieve labels for UI properties that they want to display
- ▶ To determine what UI properties can change, and therefore what MO properties must be listened for changes of (because property change events contain the MO property names and values)
- ▶ To, given a MO property change event, determine which UI properties might have changed as a result and convert the MO property value to a UI property value

3.4 Consistent user experience

Keeping a common look and feel across all IBM products allows the customer to use multiple products without having to spend extra time learning to navigate a new interface.

3.4.1 Common UI widgets

UI widgets are controls and components that display information and allow interaction with a system. Widgets include tables, trees, buttons, and entry fields. Using a set of common UI widgets helps to maintain the same look and feel across IBM Web applications. The new user interface uses the same widgets that the other IBM management platforms use. The tree UI also uses the same widgets across the various HMC platforms (HMC, SE, TKE, and POWER HMC).

The HMC follows an IBM standard set of user interface guidelines to maintain consistency throughout multiple products that IBM creates. Multiple IBM products use consistent widgets to create a similar feel of the IBM products for users. When new panels are designed for the

HMC, they must follow the panel design guidelines and be reviewed by the panel design team to ensure that all guidelines are met.

3.4.2 Panel and task consistency

Another major component of a consistent user experience is keeping panels and tasks consistent. The goal is to give different panels and tasks the same look and feel so that the user intuitively knows how to navigate and accomplish the tasks that they need to do. Every panel has the same layout. Banner, buttons, and style are all the same throughout each panel. Figure 3-8 through Figure 3-11 on page 62 demonstrate consistent layout across the selections of objects for two different tasks.

Figure 3-8 provides an example of one panel and illustrates many of the features that are common to all panels.

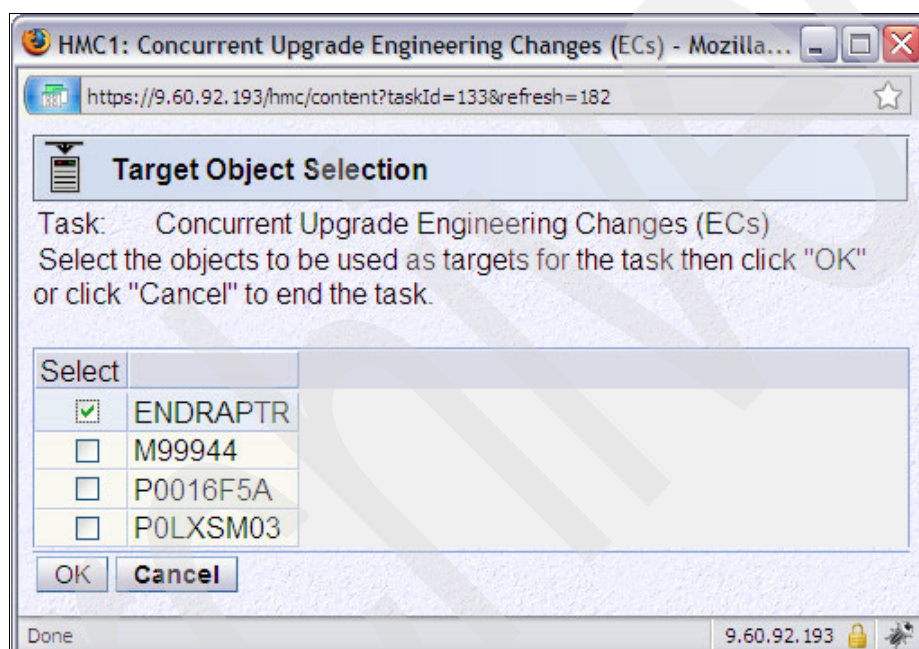


Figure 3-8 Panel consistency layout

Figure 3-8 and Figure 3-9 on page 61 display the target object selection panel that is shown when a task is launched without any objects selected. A task can be launched from selected objects or it can be launched with no objects selected in which case Figure 3-8 is launched. This method is consistent throughout the HMC.

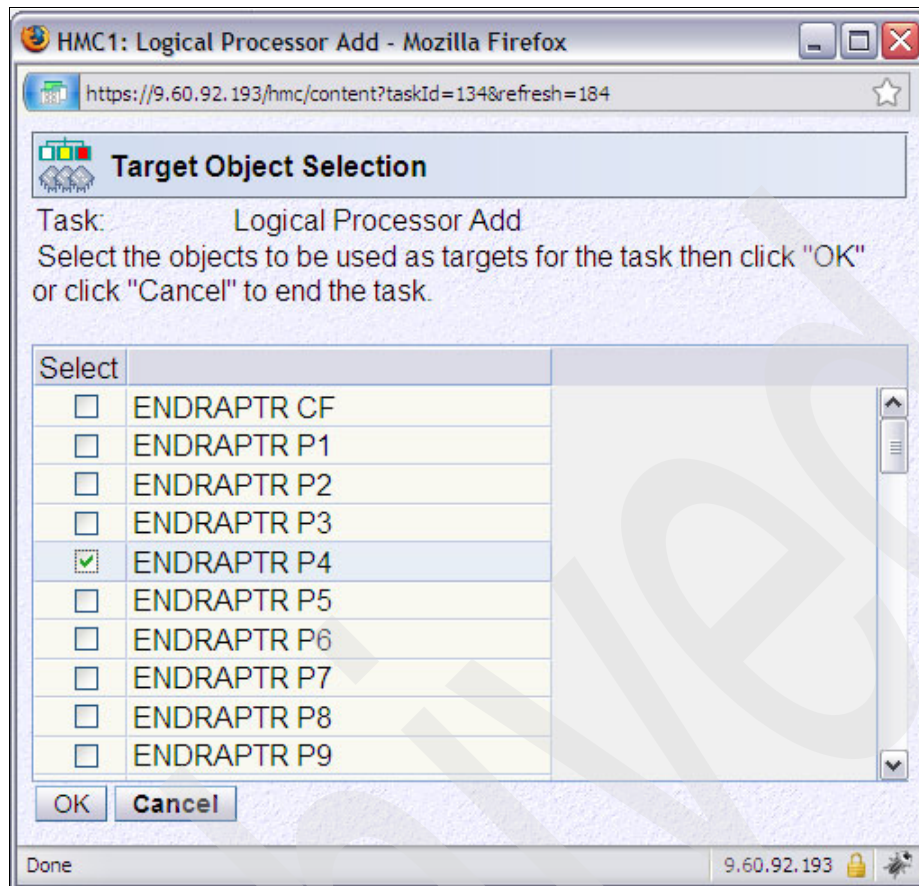


Figure 3-9 Target object selection example

Figure 3-10 on page 62 and Figure 3-11 on page 62 show the Certificate Management task and the View Security Logs task respectively. Notice how the tables and menus of the two tasks are consistent, keeping the same look and feel throughout.

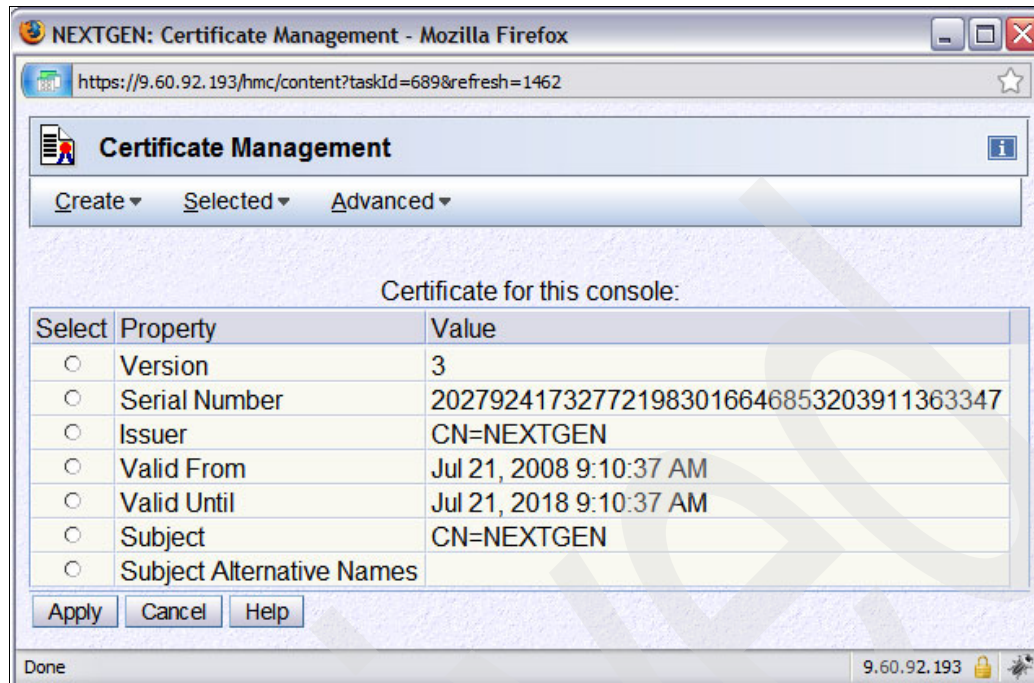


Figure 3-10 Consistent look and feel, target panel

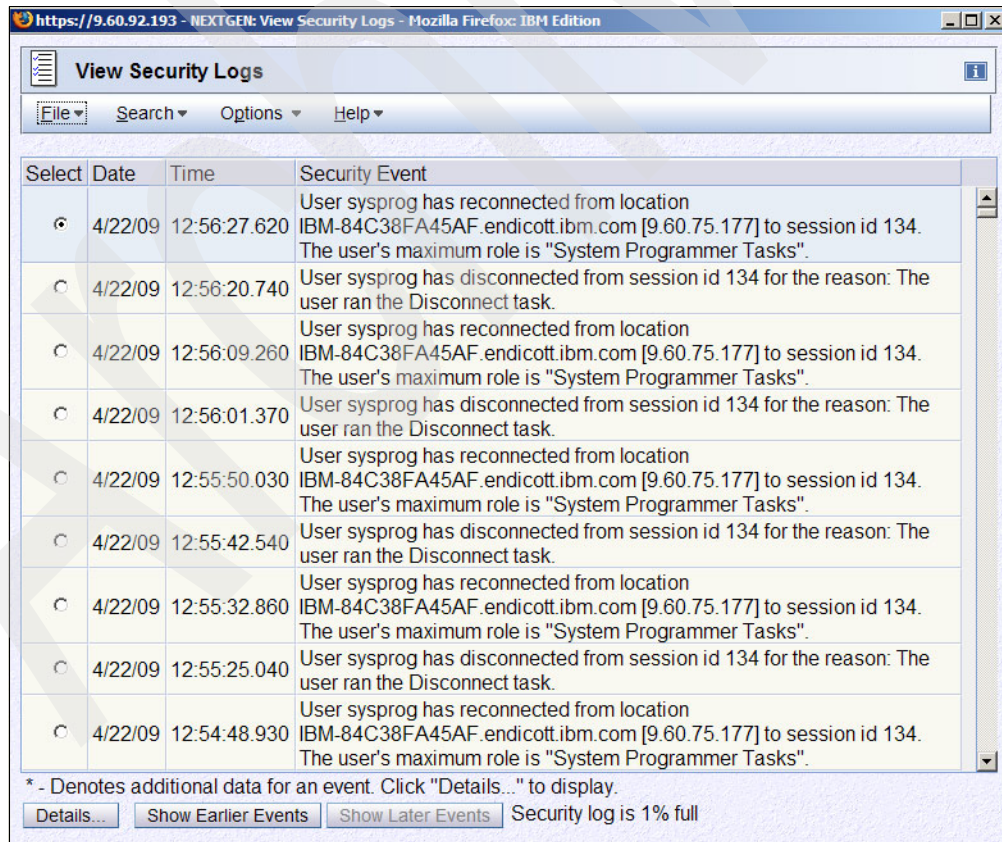


Figure 3-11 Another example of consistent panel design (partial panel)

3.5 Technologies

The HMC tree style UI was designed with XML-based UI binding. Each major component of the interface (Banner, Task bar, Navigation pane, Status bar, and all work panes that are illustrated in Figure 3-12) is defined by a section in an XML file. Because the HMC is a Web application, each section defines the Servlet, JSP, HTML, or Java class that renders its content (a Web page). All Navigation tree and Status bar nodes are defined in the XML too. Each Navigation tree and Status bar node has a corresponding work area definition, which defines the Servlet or JSP that displays its work pane.

By binding the tree style UI to XML, the tree style UI is portable. It can be moved from one console to another by defining a new XML, for instance, the tree style UI is implemented on zHMC, pHMC, and TKE consoles. Each console is unique, but uses the same framework to generate the UI. Each console defines its own Navigation tree content, Status bar, and toolbar through a console-specific XML file.

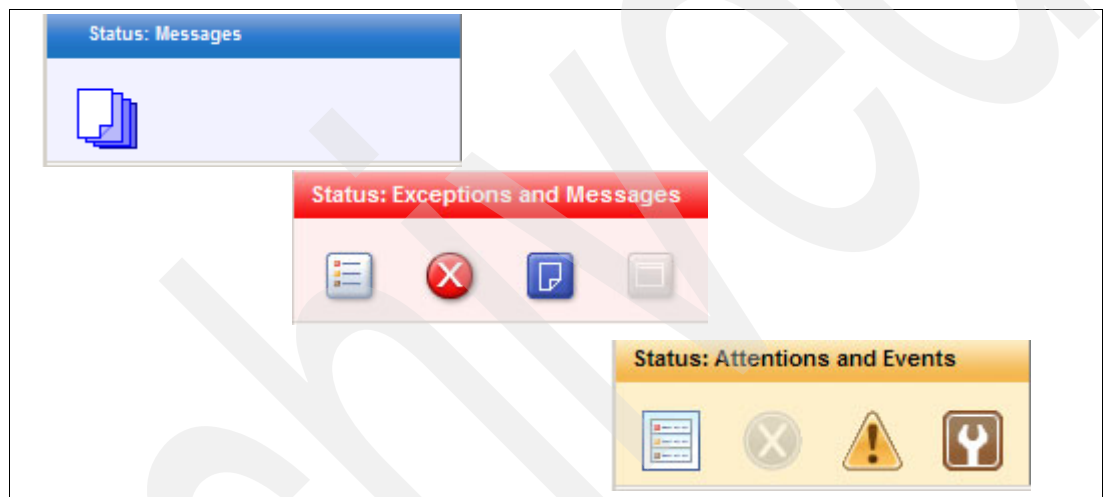


Figure 3-12 TKE, HMC, and Power HMC Status Bar implementations

Example 3-1 shows an XML fragment and description that provide an example of how the tree style UI defines its Banner and Task bar, as seen in Figure 3-13 on page 64.

Example 3-1 XML fragment

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE WUI SYSTEM "wui.dtd">
<WUI>
  <CodeLevel>6</CodeLevel>
  <Banner>
    <uri>zhmc-banner.jsp</uri>
  </Banner>
  <TaskBar>
    <class>TaskbarServlet</class>
    <params>
      <param name="help-uri">TreeStyleUI.html</param>
    </params>
  </TaskBar>
  ...
</WUI>
```



Figure 3-13 Tree style UI Banner and Task Bar generated by XML fragment

3.5.1 WUI elements

A WUI element defines the tree style Web-based user interface:

- ▶ Its `<CodeLevel>` tag indicates the level of the tree UI code for which the XML definition was created. If the tree UI code has changes that require an updated XML definition, the `CodeLevel` is raised, which prevents incompatible XML from being parsed and loaded.
- ▶ After `CodeLevel` the `Banner` element defines the contents of the banner frame of the tree style user interface. Its `<uri>` tag defines the class, JSP, or uri that is used to render the frame.
- ▶ The `<TaskBar>` tag represents the Task bar of the tree style. The `<TaskBar>` element defines the class that renders the Task bar content and any parameters that are supplied to that class.

Example 3-2 is an XML fragment and description of how the tree style UI's root navigation nodes, shown in Figure 3-14, are bound to XML.

Example 3-2 XML fragment of the root navigation node

```

<NavigationPane>
  <RootNode>welcome</RootNode>
  <RootNode>systemsmanagement</RootNode>
  <RootNode>hmcmanagement</RootNode>
  <RootNode>servicemanagement</RootNode>
  <RootNode>tasksindex</RootNode>
</NavigationPane>

```

Figure 3-14 shows the Navigation tree that the XML fragment generates.

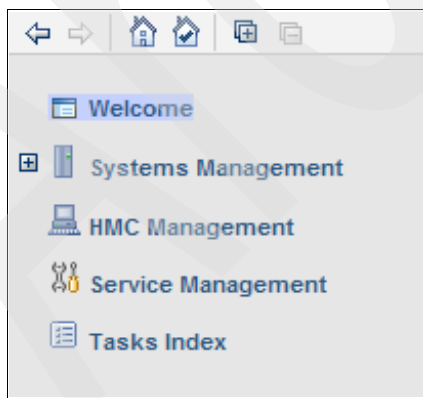


Figure 3-14 Navigation tree generated by XML fragment

The `<NavigationPane>` element defines the root nodes of the navigation tree. Each `<RootNode>` tag value is a reference to a corresponding `TreeNode` definition that is displayed as a root node of the Navigation Pane tree, for instance, `<RootNode>welcome</RootNode>` indicates that the first node in the tree should be the node `TreeNode` whose key is `welcome`. As you can see, the Welcome, systems management, HMC Management, Service

Management, and Tasks Index are all defined this way. A similar element is used to define the Status bar buttons.

Example 3-3 is an XML fragment and description of how each Navigation node item is defined in XML.

Example 3-3 XML definition of each navigation node item

```
<TreeNodes>
  <LinkNode>
    <key>welcome</key>
    <workarea>welcome-page</workarea>
    <uiinfo>
      <name>welcome.name</name>
      <caption>welcome.caption</caption>
      <image>/treestyleui/images/welcome-icon.gif</image>
    </uiinfo>
  </LinkNode>
  ...
</TreeNodes>
```

Figure 3-15 shows how the Welcome node appears as a result of the definition in Example 3-3.

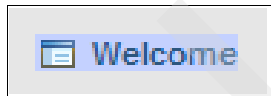


Figure 3-15 Navigation Welcome node

A `<TreeNodes>` tag begins the definition of tree nodes for the tree style user interface. A `<LinkNode>` represents a navigation tree node that is a simple link. The `<key>` tag defines the key for this node. This key value can be referenced by other nodes to include this node as a child or, as seen earlier, as a `RootNode` in the Navigation tree. The `<workarea>` tag identifies the key of this node's workarea-definition. The `<uiinfo>` defines various user interface information for this node, including its name, description, and icon.

Example 3-4 is an XML fragment and description of how a Work pane is defined for a navigation node.

Example 3-4 Definition of a Work pane for a navigation node

```
<WorkAreas>
  <workarea-def>
    <key>welcome-page</key>
    <uri>zhmc-welcome.jsp</uri>
  </workarea-def>
  ...
</WorkAreas>
```

Figure 3-16 on page 66 shows the Welcome node and its corresponding work pane.

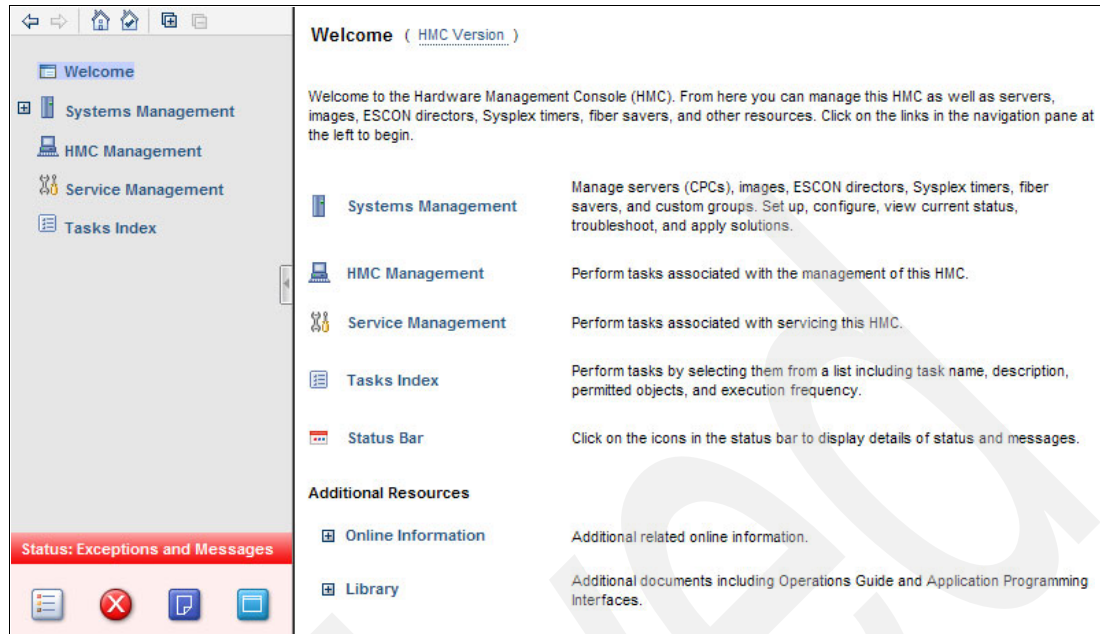


Figure 3-16 Navigation Welcome node and corresponding work pane (partial)

A `<workarea-def>` element defines the contents of the workarea that is referenced by its `<key>` value. When the Welcome node is selected, its contents that are linked through the `<workarea-def>` are displayed in the work area. The information that is displayed is defined with the `<uri>` tag. As seen in Figure 3-16, when the Welcome node is selected, the Welcome-page workarea is displayed in the Work pane, which means that in Example 3-4 on page 65, `zhmc-welcome.jsp` is loaded in the Work pane when Welcome is selected in the Navigation Pane.

The following XML fragment and description shows how navigation children are defined in the tree style UI's XML binding. Figure 3-17 on page 67 shows the resulting systems management navigation node.

Example 3-5 Navigation children defined in the tree style UI's XML binding

```
<ResourcesNode type="dynamic" display="always">
  <key>systemsmanagement</key>
  <workarea>navtree-sysmgt</workarea>
  <node-retriever>
    <class>RootNodeRetriever</class>
    <params>
      <param name="root_types">TYPE_MANAGED_OBJECT_GROUP</param>
    </params>
  </node-retriever>
  <uiinfo>
    <name>sysconfig.name</name>
    <caption>sysconfig.caption</caption>
    <image>/treestyleui/images/system_z_icon16.gif</image>
  </uiinfo>
  <children>
    <key>servers_group</key>
    <key>directortimers_group</key>
    <key>fibersavers_group</key>
    <key>customgroups_group</key>
  </children>
</ResourcesNode>
```



```

    <key>unmanagedresources_group</key>
  </children>
</ResourcesNode>

```

Figure 3-17 shows the Navigation Systems Management node and its children.

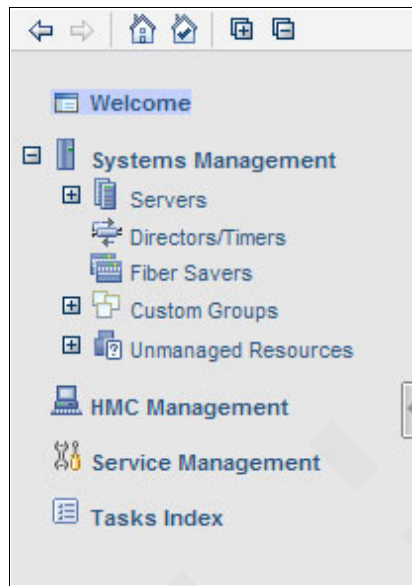


Figure 3-17 Navigation Systems Management node and its children

A `<ResourceNode>` element defines a Navigation tree node that represents a system resource, such as a server or partition. Resources can have properties, such as name, description, and status, corresponding tasks, and children, for example, a server contains partitions. The same tags for a `<LinkNode>` apply to the `<ResourceNode>` with some additions. The `<node-retriever>` element defines the `<class>` that retrieves this resource that is represented by this node along with any `<params>` that are required for the class. An optional `<children>` element defines the keys of each child node. The children of systems management (Servers, Director/Timers, Fiber Savers, Custom Groups, and Unmanaged Resources) are defined here. Figure 3-17 shows the navigation area with the Systems Management node expanded. A child defined by the `<children>` tag is displayed when the node is expanded and the user has authority to see the node.

Example 3-6 on page 68 is an XML fragment and description of how a managed resource's Work pane is defined.

The `workarea-def` of a `ResourceNode` defines the information that is needed to create and display the work area. The `<class>` tag names the associated class for this workarea. As usual, the parameters that the class needs are specified by the `<params>` tag. For `ResourceNodes`, these parameters can include the various views (Table, Tree, and Topology) that might be used to display the resources. In some cases, these parameters might reference nodes in other XML files, for instance, a table component's configuration is defined in a separate XML file and is referenced in the Web-based user interface XML as the table component parameter `table_config_key`.

Example 3-6 Definition of a managed resource's Work pane

```
<workarea-def>
  <key>navtree-sysmgt</key>
  <class>ResourcesWorkareaAction</class>
  <params>
    <param name="workarea_name">NavTreeDefaultWorkarea</param>
    <param name="components">table;treetable;topoviewer</param>
    <param name="table.class">ResourcesWorkareaTable</param>
    <param name="table.params">table_config_key=default</param>
    <param name="treetable.class">ResourcesWorkareaTreeTable</param>
    <param name="treetable.params">table_config_key=default-tree</param>
    <param name="topoviewer.class">ResourcesWorkareaTopology</param>
    <param name="topoviewer.params"></param>
  </params>
</workarea-def>
```

Figure 3-18 shows the resulting hardware systems management Work pane with its available display options (Table, Tree, and Topology) listed in its Views menu in the upper-right corner.

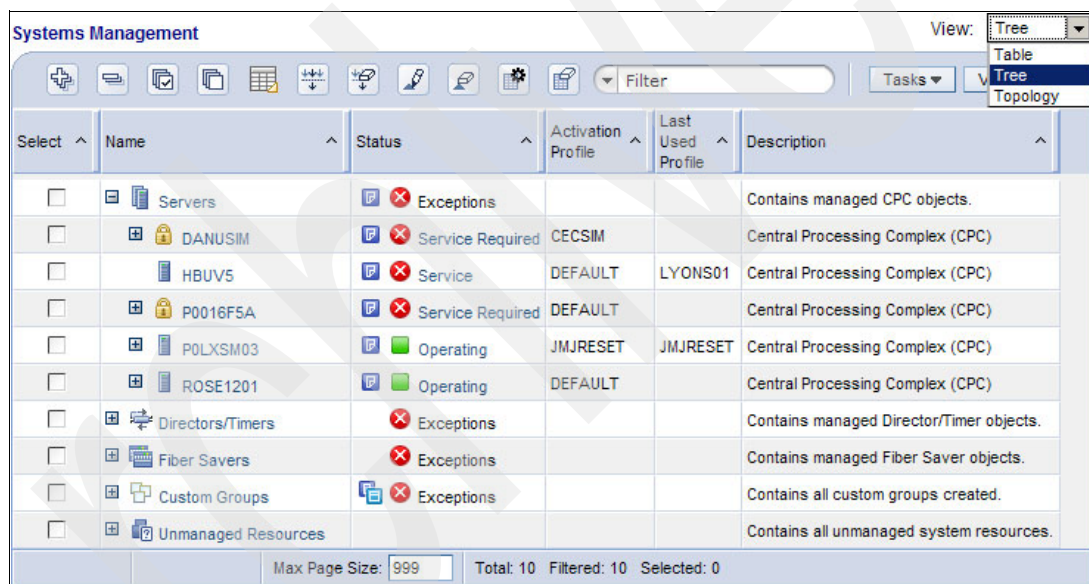


Figure 3-18 Systems management work pane views (partial view)

3.5.2 Plug-ins

The tree style UI allows extensibility through the use of plug-ins. OEM partners can create a plug-in XML file that defines additional content to add to the tree UI's Navigation pane. As the Web-based user interface, XML is the core of a tree UI implementation. A plug-in XML file is the core of a plug-in implementation within the tree UI that defines the sub-tree that is the plug-in to be attached to the main tree and defines what work areas and tasks display for each tree node.

The root element of a plug-in is the <Plugin> node. This element can contain content that is similar to a WUI element to define the OEM plug-in, as shown in the sample plug-in XML in Example 3-7 on page 69.

Example 3-7 Sample plug-in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Plugin SYSTEM "plugins.dtd">
<Plugin>
  <RootNode>plugin</RootNode>
  <TreeNodes>
    <LinkNode>
      <key>plugin</key>
      <workarea>plugin-workarea</workarea>
      ...
    </LinkNode>
  </TreeNodes>
  <WorkAreas>
    <workarea-def>
      <key>plugin-workarea</key>
      <uri>plugin.jsp</uri>
    </workarea-def>
    ...
  </WorkAreas>
  ...
</Plugin>
```

3.5.3 UI technology change tolerance

All panels for HMC tasks are defined in a platform-independent manner, which means that the interface is abstracted from the actual implementation of the system and its technology. This approach protects the panel from changes in rendering technology.

This approach occurs by using an abstract user-interface markup language in which each user-interface widget is defined in an XML file. The language can define buttons, layouts, tables, and so on. Example 3-8 shows such a panel definition.

Example 3-8 A user-interface widget defined in an XML file

```
<AXML>
<DATA-GROUP NAME="HWMessagesPanel">
  <CAPTION SRC="resource://HWMessagesPanel.TEXT"/>
  <TABLE NAME="MsgTable" SELECTION-POLICY="MULTIPLE">
    <ROW-STRUCTURE>
      <STRING NAME="DateCol" READ-ONLY="TRUE">
        <CAPTION SRC="resource://HWMessagesPanel.DateCol.TEXT"/> </STRING>
      <STRING NAME="TimeCol" READ-ONLY="TRUE">
        <CAPTION SRC="resource://HWMessagesPanel.TimeCol.TEXT"/>
      </STRING>
      <STRING NAME="TextCol" READ-ONLY="TRUE">
        <CAPTION SRC="resource://HWMessagesPanel.TextCol.TEXT"/>
      </STRING>
    </ROW-STRUCTURE>
  </TABLE>
  <GROUP NAME="Group1">
    <ACTION NAME="DetailsButton" DEFAULT="TRUE">
      <CAPTION SRC="resource://HWMessagesPanel.DetailsButton.TEXT"/>
    </ACTION>
```

```

<ACTION NAME="DeleteButton">
  <CAPTION SRC="resource://HWMessagesPanel.DeleteButton.TEXT"/>
</ACTION>

...
</GROUP>
</DATA-GROUP>
</AXML>

```

The AXML file is then parsed and rendered into the panel. The AXML is platform independent, so it can be rendered as Java Swing, HTML, or JSF renderers. As the AXML file defines the interface, it is independent of the platform and the code. If there is an updated rendering technology, the user interface does not need to be changed because everything is defined in AXML. The renderer can be updated to display the data in the desired way without having to change each and every interface component. Figure 3-19, Figure 3-20, and Figure 3-21 on page 71 provide views of the panel, in Example 3-8 on page 69, rendered on the HMC using a Java Swing renderer, and rendered with a default Web renderer.

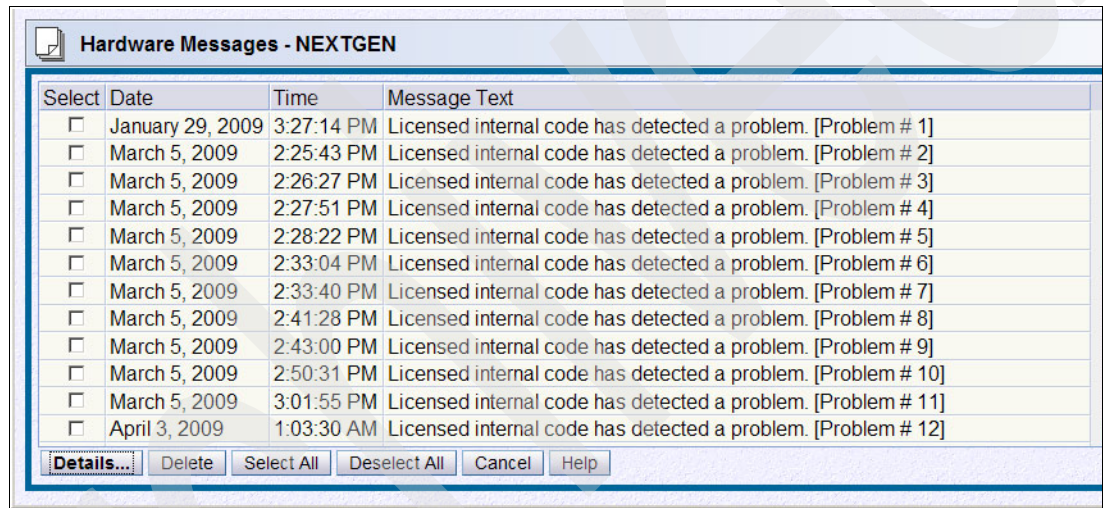


Figure 3-19 AXML-HMC Rendered (partial panel)

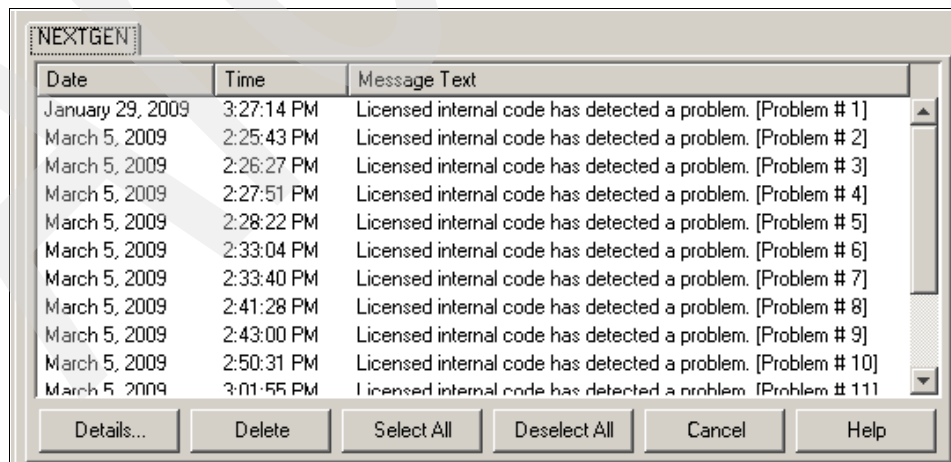


Figure 3-20 AXML-Swing Renderer

NEXTGEN	Select	Date	Time	Message Text
	<input type="radio"/>	January 29, 2009	3:27:14 PM	Licensed internal code has detected a problem. [Problem # 1]
	<input type="radio"/>	March 5, 2009	2:25:43 PM	Licensed internal code has detected a problem. [Problem # 2]
	<input type="radio"/>	March 5, 2009	2:26:27 PM	Licensed internal code has detected a problem. [Problem # 3]
	<input type="radio"/>	March 5, 2009	2:27:51 PM	Licensed internal code has detected a problem. [Problem # 4]
	<input type="radio"/>	March 5, 2009	2:28:22 PM	Licensed internal code has detected a problem. [Problem # 5]
	<input type="radio"/>	March 5, 2009	2:33:04 PM	Licensed internal code has detected a problem. [Problem # 6]
	<input type="radio"/>	March 5, 2009	2:33:40 PM	Licensed internal code has detected a problem. [Problem # 7]
	<input type="radio"/>	March 5, 2009	2:41:28 PM	Licensed internal code has detected a problem. [Problem # 8]
	<input type="radio"/>	March 5, 2009	2:43:00 PM	Licensed internal code has detected a problem. [Problem # 9]
	<input type="radio"/>	March 5, 2009	2:50:31 PM	Licensed internal code has detected a problem. [Problem # 10]
	<input type="radio"/>	March 5, 2009	3:01:55 PM	Licensed internal code has detected a problem. [Problem # 11]
	<input type="radio"/>	April 3, 2009	1:03:30 AM	Licensed internal code has detected a problem. [Problem # 12]
Page 1 of 2 <input type="button" value="1"/> <input type="button" value="Go"/> Total: 15 Displayed: 12				
<input type="button" value="Details..."/> <input type="button" value="Delete"/> <input type="button" value="Select All"/> <input type="button" value="Deselect All"/> <input type="button" value="Cancel"/> <input type="button" value="Help"/>				

Figure 3-21 AXML Web Renderer (partial panel)

3.5.4 Server push (real-time status)

The HMC deploys Push methodology instead of the traditional Pull methodology. The Push method traditionally involves an update being triggered by a user action, such as the use of a Refresh button. Push technology is a type of Internet-based communication where the request for updated data comes from the server-side of an application instead of the client-side. So instead of having to push a Refresh button to receive updated content, the Web page is refreshed when new content is available. An instant messaging client is an example of *server push*. Any updates to the chat window are immediately seen. Users do not have to refresh their window to get the most updated data.

The HMC operates on a similar concept. When there is a change on the server-side of the HMC, it is sent to the client to provide the most up-to-date data. The HMC is a Web application, yet it runs like a desktop application. It is asynchronous with the data. AJAX is used to keep everything up-to-date, which allows the user to perform tasks and operations in real time.

A task controller process begins for every user as they log into the console. The process consists of servlet code and browser javascript code. An initial HTTP request from the browser is kept alive/open for the servlet code to communicate with the browser javascript code. The task controller process consists of the following steps:

1. When the user clicks a link on the console, it causes an HTTP request to come into the console Web server.
2. The Web server routes the request to the appropriate task code to handle. If the task needs to update the console displays as a result of this action and the task sends javascript code to the task controller servlet code.
3. The task controller servlet code receives the javascript and queues it to be sent to the browser.

4. The script is eventually dequeued and sent to the browser and the browser task controller javascript code executes the script to update the console display, as needed.
5. The session listens to see if any updates for the console are on the page that the user is viewing. If updates for the page that is being viewed exist, the session refreshes the page and displays the updates.

The task controller process also maintains a *heart beat* timer between the servlet and browser code to be sure that the browser and servlet are still maintaining communication. With each script that the servlet sends to the browser, it sends an indication that they are still connected. If a script is not sent within 30 seconds, a simple script to indicate that they are still connected is sent to the browser. The browser JavaScript code continually turns off the indication that they are still connected then waits 45 seconds to see if it gets turned back on. If the browser JavaScript code finds it is disconnected, it sends a request to the server to force disconnect the user and the main UI panel is closed. The user receives a panel with a message saying that connectivity was lost.

The task controller process works basically the same for a user regardless of whether they are logged on locally or remotely on most browsers. However, because the HMC is a Web application that can be accessed remotely, multiple Web browsers must be supported. Due to a limitation in certain browsers, an applet had to be used to establish the communication between the browser and the servlet and to execute the scripts.

3.5.5 Disconnect (roaming users)

Most Web applications allow the user to log on and perform various tasks. The HMC even allows the user to start several tasks at the same time. Each running task displays its output in a separate browser window. Some HMC tasks take hours to run, such as the one that causes the mainframe to IML. Logging off of a session causes all running tasks to be terminated. As an alternative, users can disconnect the session. When a session is disconnected, all tasks continue to run. Any output that the tasks generate is not displayed anywhere, but it is not discarded either. A user can reconnect to a disconnected session at a later time, possibly from a different browser or location. When a session is reconnected, the browser windows for all running tasks are re-opened to display the current output for the task. An example of this application is an Operator who starts a long-running operation from home before disconnecting the session and driving in to work. When he gets to work, he can reconnect to the session and wait for the task to end. Similarly, he could start a long-running task from work and check on it later from home, or disconnect a session that he started from his office and reconnect it on the HMC itself after walking to the raised floor to investigate a problem.

3.6 User defined resource relationships

Using the tree style UI users can create custom groups and hierarchies for managed objects. User-defined groups can be accessed in the Custom Groups node, shown in Figure 3-22 on page 73, under Systems Management in the Navigation pane. The Custom Group node includes three predefined groups:

- All Images includes all of the partitions defined to all servers managed by the HMC.
- All Objects is a collection of all resources managed by the HMC.
- All z/VM Virtual Machines appears when virtual machines are being managed for any server images that are running z/VM 5.3 or later.

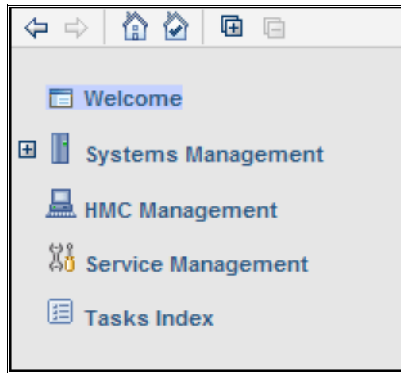


Figure 3-22 Custom groups

Users can create their own groups of managed resources. The user can also create groups of custom groups, which allows for hierarchical customization. Custom groups can be created in two ways by launching the Grouping task. If a managed resource is selected, the grouping task can be launched from that resource to create a custom group that includes the currently selected managed object. Groups can also be created by pattern matching, which is done by launching the Grouping task without a target selected and entering a pattern (a regular expression). All managed objects whose name matches the pattern are automatically added to the group.

3.7 Systems management launch in context

Upward management allows higher level consoles and business partners to manage the HMC and execute its tasks remotely. The HMC allows upward management through three different APIs:

- ▶ Simple Network Management Protocol (SNMP)

SNMP is the most widely used management interface. It provides a framework for management along with a protocol for the exchange of that information. SNMP assumes the manager/agent relationship:

- An agent is a part of the software that manages local operation and carries that information up to a manager.
- A manager handles agents and provides single point-of-entry amount multiple agents.

- ▶ Common Information Model (CIM)

CIM is a system management architecture that the Distributed Management Task Force controls. CIM employs an object-oriented representation of manageable objects and inter-object associations, which is expressed in Managed Object Format (MOF), which is a C++ like class representation with equivalent XML representation. Vendors can extend this framework for platform-specific functionality. The API allows a management client to manipulate CIM objects.

- ▶ Command Line Interface (CLI)

A Command Line Interface is also implemented for use. POWER HMC has this implemented through ssh, while the System z HMC employs a CLI through SNMP.

See “Automation” on page 169 for more information about CIM and SNMP.

3.7.1 UI task integration: Launch-in-context

Launch-in-context, partly illustrated in Figure 3-23, provides a way for a higher-level manager to control lower-level element managers seamlessly. In terms of use, the user never knows that the higher-level manager, such as an external Web application is controlling lower-level element managers (HMC). Everything is *launched in context* of the current manager, which provides a seamless flow from the user interface. Launch-in-context not only simplifies usage, but it also provides various advantages for development.

The scope of some management consoles can include managing Hardware Management Consoles. Users of those consoles perform various HMC tasks and functions within the console. As far as the user is concerned, they are performing tasks on HMC-related tasks that are on that console. In essence, the logic is done through the HMC, and the user interface is updated on the higher-level console, which provides the smooth execution of tasks from the higher-level manager while using the underlying element manager's logic.

Launch-in-context also provides a way to separate development of the higher-level manager and the element managers. Because tasks are performed by executing the element manager's logic, the higher-level manager does not have to be updated each time new functionality is introduced to an element manager, and there are several advantages to this approach. Any new features that are added to the element manager are automatically available to higher-level managers, which provides benefits in both that any new content is automatically provided to the higher-level manager, and it reduces the cost of developing by avoiding the need to provide duplicate functions. This approach also breaks the dependency of the two managers because the higher-level manager does not need upgrading when a new element manager is released; therefore, separate release dates can be maintained. Lastly, multiple versions of the element manager can work with the same version of the higher-level manager.

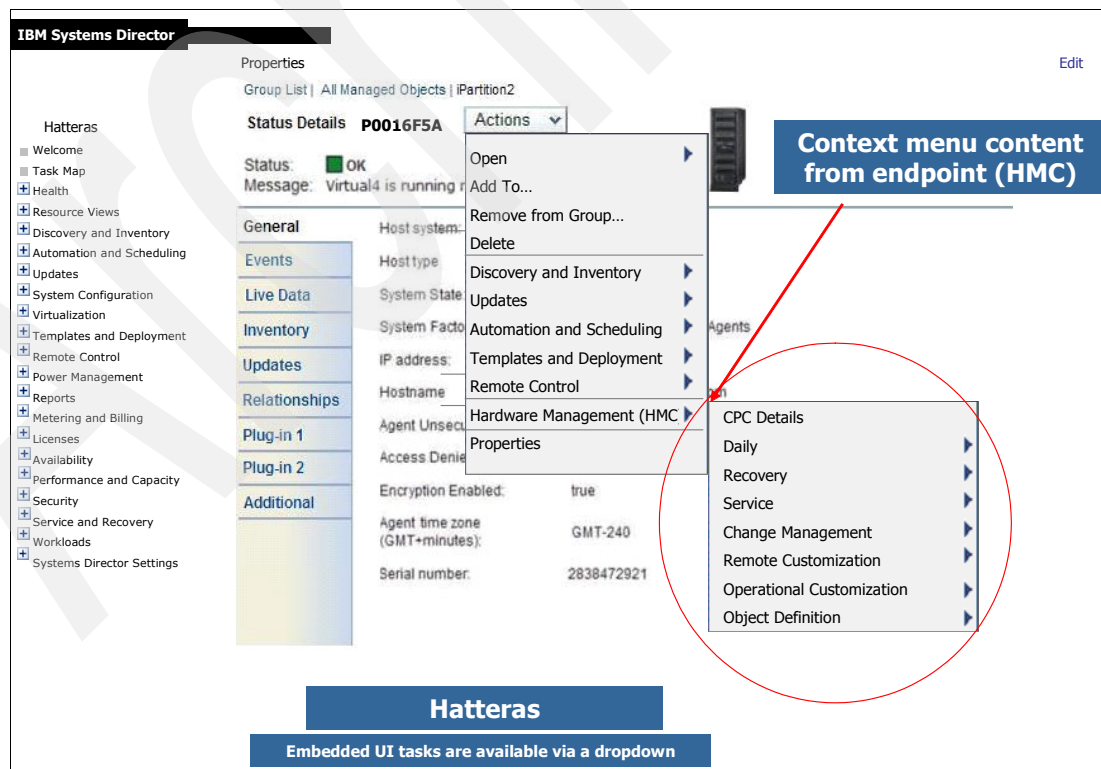


Figure 3-23 Launch-in-context

3.8 Questions and discussions

1. What significant control functions are exclusive to the tree style? The classic style?
2. What high-level markup language is highly involved in HMC implementation?
3. How difficult would it be to expand an area of HMC panels?
4. How much XML skill does an HMC user need? Why did we provide this overview?

Archived

User management

In this chapter, we discuss the following topics, as they apply to the HMC:

- ▶ User management:
 - Typical management scenarios
- ▶ Managing user profiles and access
- ▶ Adding a new user with an operator role
- ▶ Changing user passwords:
 - Password profiles
- ▶ Creating, copying, modifying, or deleting managed resource roles:
 - Creating a new managed resource role
- ▶ Creating, copying, modifying, or deleting task roles:
 - To create a new user task role
- ▶ Manage users with data replication

After completing this chapter, the student should understand the nature of different HMC user roles (types of users) and how they are created and managed.

4.1 HMC User Management overview

A user profile is a combination of a user ID, authentication mode, permissions, and a text description. Permissions represent the authority levels that are assigned to the user profile for the objects that the user has permission to access.

An HMC Administrator defines the user name, password, and a set of permissions that determine the set of objects and tasks that a user can perform. The HMC User Profiles task allows an Administrator to add, remove, or edit user definitions. When creating a new user using the User Profiles task, the Administrator selects one or more User Roles from a list of all predefined and customized user roles.

Permissions are defined as containers of objects and tasks, known as User Roles. A User is only allowed to view and manipulate the set of objects that are defined by the union of all its User Roles. Similarly, a User can only see the tasks that are defined in its User Roles. The Administrator can create customized User Roles containing any set of objects or tasks by using the Customize User Controls task, for instance, the Base tower defines a Base Operator Task User Role that contains all of the tasks that it believes are appropriate for an Operator User.

4.1.1 Typical scenario

A customer might be leasing partitions 1, 2, and 3 on a particular CEC. The System Administrator might want to make a User Role for the customer to have authority to only see partition 1, partition 2, and partition 3 on the particular CEC. The Administrator would use the Customize User Roles panel to create a Customized Managed Resource Role. Then he would use the User Profiles panel to create user IDs and assign this customized managed resource role to these IDs, which would ensure that the leasing customer would only have access to the objects that are specified in the customized managed resource role.

4.2 HMC user management defaults and definitions

The general operational paradigm uses managed objects, groups of objects, and tasks that can be performed on one or more objects. A user logs on the HMC and thus is authorized for some predefined level of control of a defined set of objects and associated tasks. In general, the user selects a task to be performed on one or more objects and initiates the task. The user interface provides event-driven status updates and task status. Objects can be grouped with other objects in tower-defined or user-defined groups.

There are five predefined default user IDs, user roles, and passwords that are established as part of a basic HMC. Different rights can be assigned to the users so that they have different capabilities to control the HMC and the attached machines. A list of predefined users is provided with the HMC.

Table 4-1 lists some basic actions and user management definitions.

Table 4-1 User management terminology

Action or term	Meaning in HMC User Management contexts
Copy	Copies the selected object. Use this option to create an exact copy of the selected object.
Group	An object that is defined to the HMC that represents a predefined (defined CPCs for example) or a user-defined (the user built it themselves) collection of managed resources.
Managed Resource	A resource that is defined to the HMC that represents a device. After a resource is defined to the HMC it is called a managed resource, for example, a CPC is a managed resource.
Managed Resource Group	A collection of resources that can contain Managed Resources or Managed Resource Groups, for example, a group of machines on the raised floor that represents all test machines.

Action or term	Meaning in HMC User Management contexts
Managed Resource Role	This is a type of User Role. It contains managed resources that are controlled by Users. An example is the default role Defined Fiber Saver Managed Objects. So as Fiber Savers are defined and undefined on the system, the user can see them come and go as long as they have permission for this default role Defined Fiber Saver Managed Objects.
User Role	A set of managed resources AND tasks that can be associated and controlled by a User. On the HMC, every user has a managed object role and a task role.
Modify	Allows the HMC Administrator to change the current permissions of the selected user or user-defined role.
Task	A task is a set of HMC actions that accomplish an activity in a predefined or known sequence.
Task Group	This is a collection of tasks.
Task Role	This is a type of User Role. It contains resources and the tasks that are allowed with each resource type. These resources are associated with a User to control.
User	Anyone that can log onto the HMC with assigned privileges or capabilities, for example an Operator or System Programmer or Customer Engineer.
Role	A role groups User capabilities or privileges to a known HMC responsibility. For example, an operator is a role that is responsible for daily System z monitoring and control. A role makes it easy to manage user access to the HMC as they can be granted or deleted easily. When a role is assigned to a User, the role becomes the User Role. Role and User Role are used interchangeably.

4.3 HMC user management definitions

Figure 4-1 on page 80 illustrates the relationships between the HMC User, an HMC Role, and an HMC Task. Each user is assigned a role on the HMC. A role must be assigned to the User, which allows them to manage an HMC task and an HMC resource. Similarly HMC tasks and HMC resources roles can also be assigned into groups, such as Administrator roles or Operator roles.

In a typical customer environment, the HMC has multiple Operators and System programmer roles. It is possible to assign a single User Role to multiple Users, as shown in Figure 4-1 on page 80. We describe the advantages of creating a new user with the same or similar role in section 4.5, “Adding a new user with an operator role ” on page 80. You can also assign multiple roles to a single user, which you perform when a single user needs privileges of both an Access Administrator and an Operator.

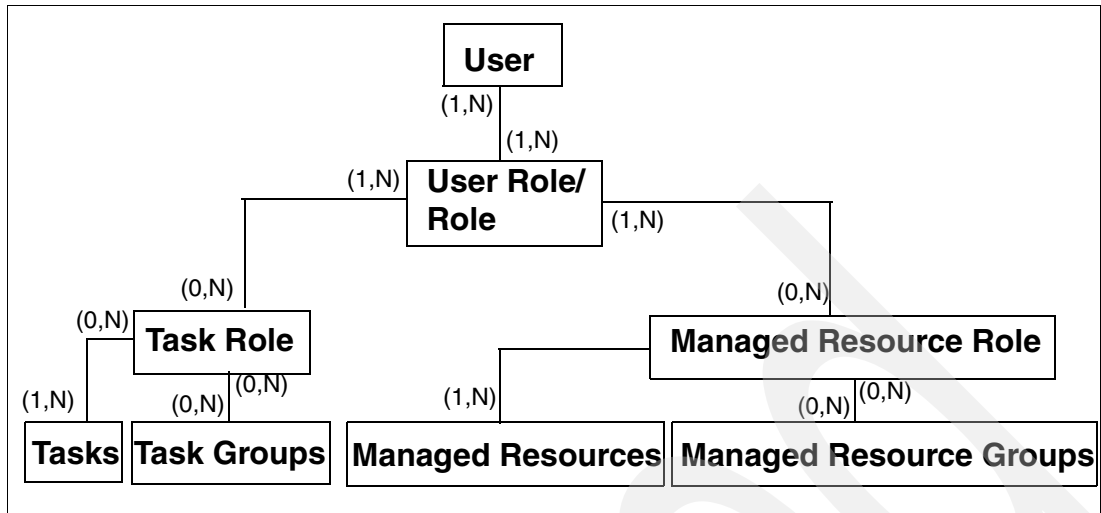


Figure 4-1 HMC user administration entity relationships

4.4 Manage User Profiles and access

The HMC administrator is responsible for managing the users of the HMC.

The User Profiles task, shown in Figure 4-2, is located in the Console Actions Work Area and displays all of the currently defined users on the system and has a description field for each user ID. It gives the Administrator options to Add, Copy, Delete, and Modify the currently defined users. Some user IDs are predefined. Any user ID can be copied using the User menu, and the newly created user ID must be modified, for example, you can copy the Operator User Id and modify its permissions. All user IDs can be deleted from the system with the exception of the Access Administrator. At least one Access Administrator level user must remain on the system.

User Profiles				
User ▾ Help ▾				
Select a User ID below and click "User" to manage the console users.				
Select	User ID	Description	Last Logon Date	Last Logon Time
<input checked="" type="radio"/>	ACADMIN	Access administrator level user	08/07/2009	09:09:55
<input type="radio"/>	ADVANCED	Advanced operator level user		
<input type="radio"/>	OPERATOR	Operator level user		
<input type="radio"/>	SERVICE	Service representative level user	07/21/2009	07:21:40
<input type="radio"/>	SYSPROG	System programmer level user	08/07/2009	08:29:55

Figure 4-2 User profiles window

4.5 Adding a new user with an operator role

Sometimes it is necessary to have more than one operator on an HMC at a time, each with different daily responsibilities. In this case, the Administrator might want to add a new user with a unique name and give this new user permission to the Operator Tasks role. Unique IDs

make it easy for the HMC to track what tasks are being performed by which user. If you have several people who need Operator level access, and they all share a single Operator ID (which as we know is a poor security practice), it could be difficult to figure out who caused a failure. In addition, some tasks cause their targets to be busied while they run. Therefore a CPC can be out of service for up to 20 minutes. Rather than having all of your Operators waiting for that one task to complete, unique IDs permit each user to work independently.

To add a new user:

1. Log onto the acsadmin user ID, and invoke the User Profiles task that is located in the Console Actions Work Area.
2. Select the **User** menu, and then select the **Add** menu item to invoke the Add User window. The HMC displays the Add User window, as shown in Figure 4-3.

Figure 4-3 Add user window

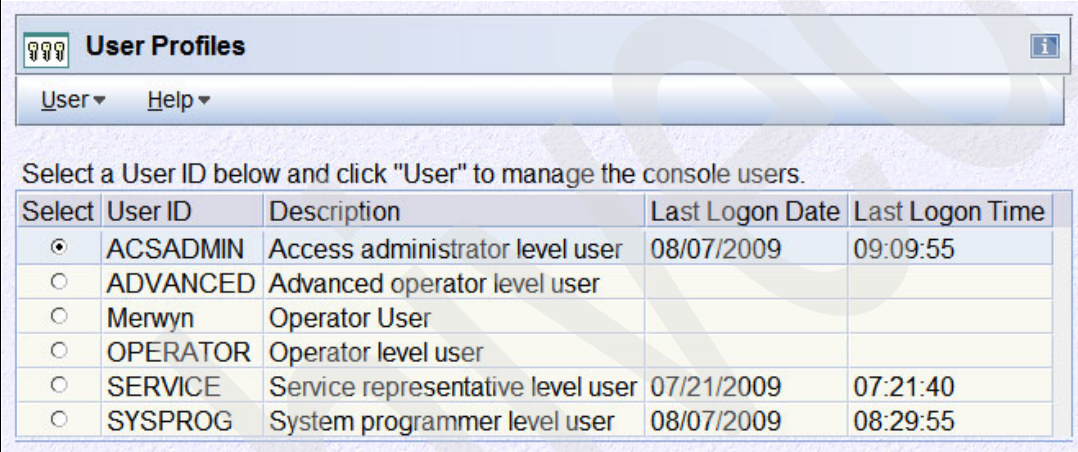
3. Insert the new user ID and a description of the user ID, and then select the authentication type for the new user ID.
4. Select at least one Managed Resource Role and the Access Administrator Tasks from Task Roles to create a new user with the Administrator permissions. To create a new user with the Operator permissions, select Operator Tasks from Task Roles.

Note: The administrator must supply a value for each field on this panel. A value is not automatically generated if a value is not provided for each of the fields on this panel.

Note: If you select **Local Authentication**, a password is required for the user to log onto the HMC. Therefore, on this window you must provide a password for the new user, and re-enter the new password for verification.

Optionally, password verification can be delegated to an LDAP server when the user logs on. If you select **LDAP Server authentication**, you must set up an Enterprise Directory Server definition using the Define Server button. For more information about this task, see Chapter 5, “Hardware Management Console remote access” on page 97.

Optionally, you can set the user properties (described in 4.5.1, “Customizing User Properties” on page 82). Click **OK** to create a new user. The new user ID is added in the User Profiles window, as shown in Figure 4-4.



The screenshot shows a window titled "User Profiles" with a toolbar containing "User" and "Help" buttons. Below the toolbar, there is a text instruction: "Select a User ID below and click 'User' to manage the console users." Underneath this instruction is a table with five columns: "Select", "User ID", "Description", "Last Logon Date", and "Last Logon Time". The table contains six rows of user data. The first row, "ACADMIN", is selected with a radio button. The other rows are "ADVANCED", "Merwyn", "OPERATOR", "SERVICE", and "SYSPROG".

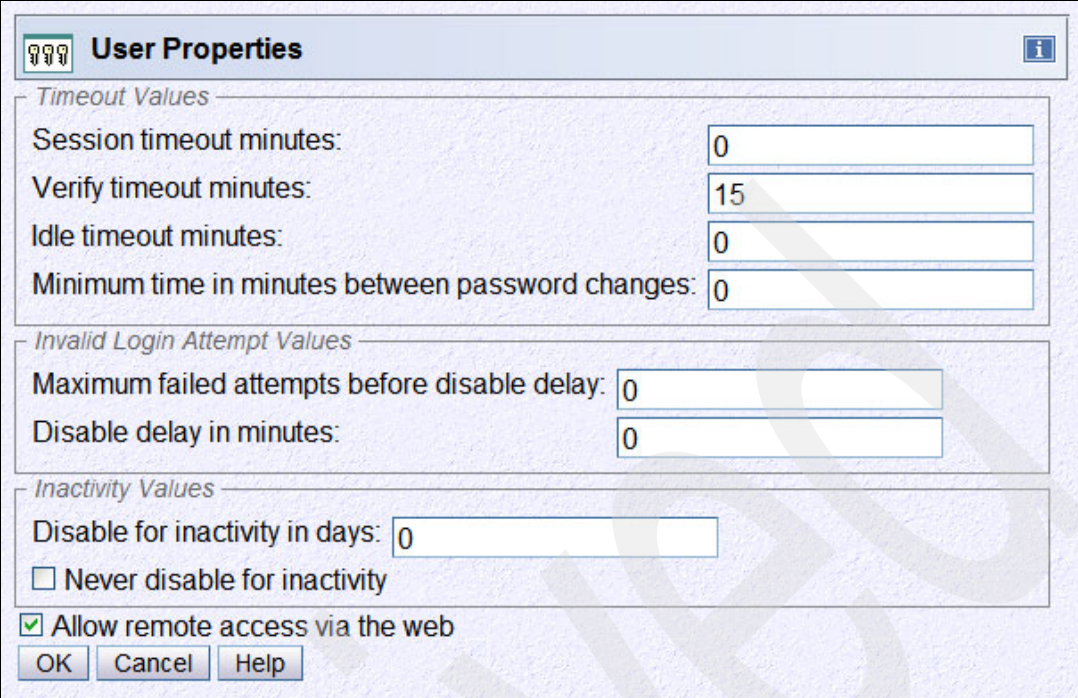
Select	User ID	Description	Last Logon Date	Last Logon Time
<input checked="" type="radio"/>	ACADMIN	Access administrator level user	08/07/2009	09:09:55
<input type="radio"/>	ADVANCED	Advanced operator level user		
<input type="radio"/>	Merwyn	Operator User		
<input type="radio"/>	OPERATOR	Operator level user		
<input type="radio"/>	SERVICE	Service representative level user	07/21/2009	07:21:40
<input type="radio"/>	SYSPROG	System programmer level user	08/07/2009	08:29:55

Figure 4-4 User ID list is updated

4.5.1 Customizing User Properties

To customize User Properties:

1. From the Add User window, Figure 4-3 on page 81, select the **User Properties** button to modify additional user properties, for example, the User Properties window provides the capability to give the new user ID permission to access the HMC remotely, as shown in Figure 4-5 on page 83.



User Properties

Timeout Values

Session timeout minutes: 0

Verify timeout minutes: 15

Idle timeout minutes: 0

Minimum time in minutes between password changes: 0

Invalid Login Attempt Values

Maximum failed attempts before disable delay: 0

Disable delay in minutes: 0

Inactivity Values

Disable for inactivity in days: 0

☐ Never disable for inactivity

☒ Allow remote access via the web

OK Cancel Help

Figure 4-5 User Properties window

2. Select **OK** to save the updated User Properties.

4.5.2 Testing the new user

As shown in Figure 4-6 on page 84, the Operator is only given a limited set of tasks that he can perform on the Managed Objects that are currently defined on the system. In this scenario, the Operator can see all Managed Objects that are defined on the system, but can only perform a restricted subset of functions on these objects. Due to the permissions of the Operator role, the Operator's tasks are restricted to actions that are not disruptive to the zSeries, for example, a disruptive task is Power Down.

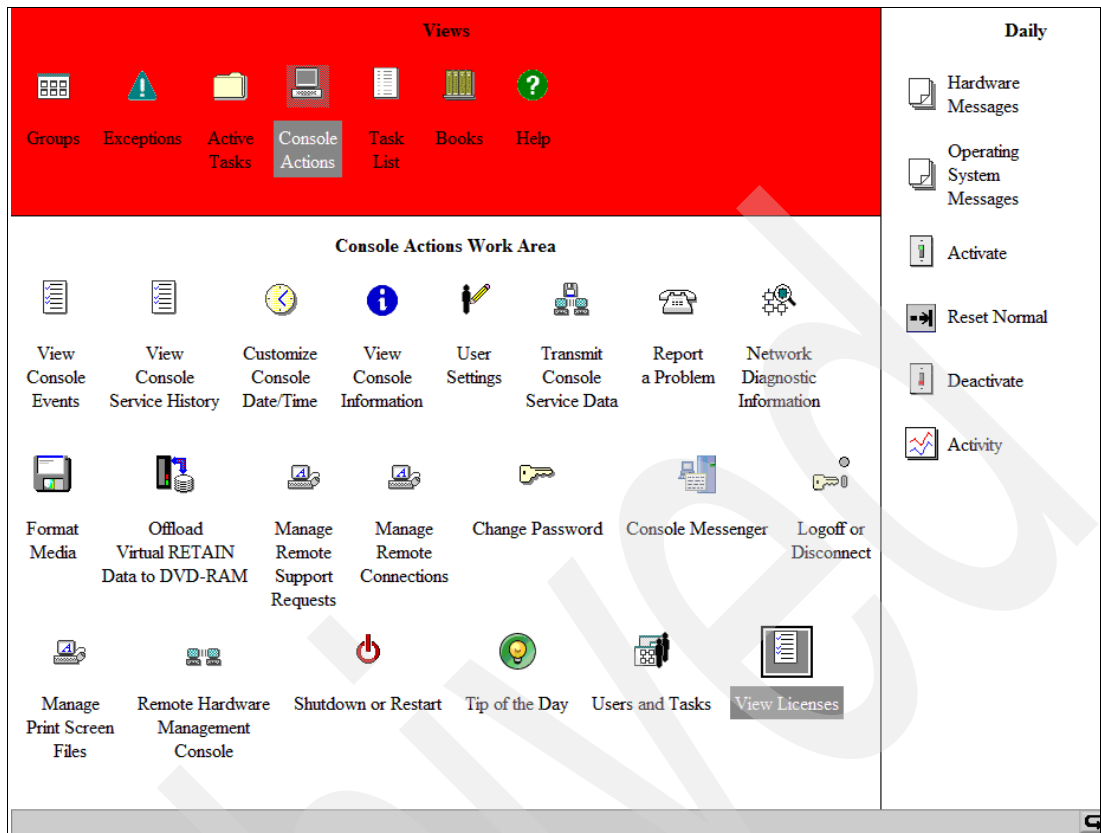


Figure 4-6 Daily tasks for default operator role

4.6 Changing the user password

The Administrator can give each user the capability to change their own password. If the user has permission for this task, simply log onto the HMC, and the Change Password task is in the Console Actions Work Area. The current password is needed for this option, and the new password must be different from the current password. Figure 4-7 on page 85 shows the Change Password window.

The Administrator can change a password for any user through the User Profiles task. If permitted, users can also change their own password.

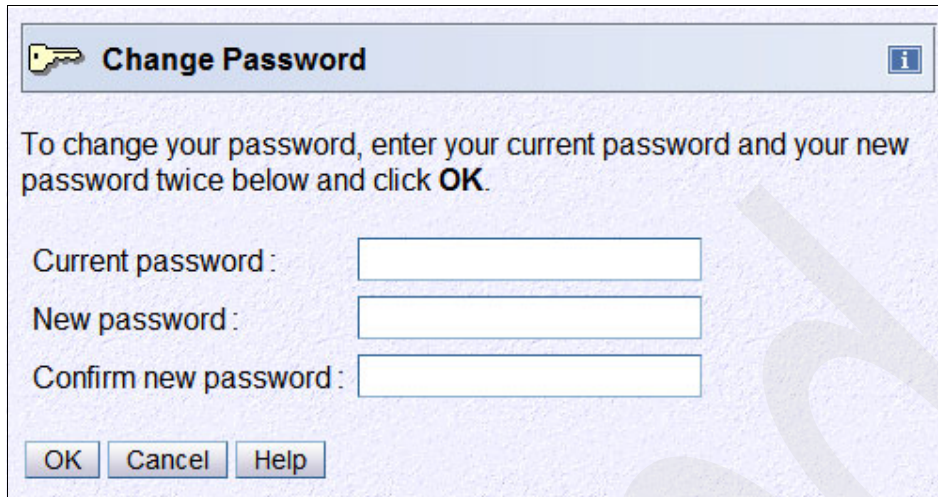
A screenshot of a 'Change Password' dialog box. The title bar is light blue with a key icon on the left and an information icon on the right. The main area has a light blue background. It contains the text: 'To change your password, enter your current password and your new password twice below and click OK.' Below this text are three text input fields. The first is labeled 'Current password :', the second 'New password :', and the third 'Confirm new password :'. At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Help'.

Figure 4-7 Change Password window

4.6.1 Password profiles

A user ID is associated with a single password rule. A user's password must conform to that password rule. Password rules are managed through the Password Profiles task.

The Password Profile task, shown in Figure 4-8 on page 86, can be used to create, customize, or verify the password rules that are assigned to the system users. Multiple password rules can be defined. Using the password rules section you can assign individual rules for the System User when creating a password. In addition, you can optionally set more specific rules for individual parts of the password in the Password Rule Parts section. A rule cannot be deleted if a user ID still references it. Password rules are made up of multiple parts that describe the details of the password criteria. The three predefined non-editable password rules are:

- ▶ Basic: Enforce strict password rule unchecked
- ▶ Strict: Enforce strict password rule checked
- ▶ Standard: Complies with industry accepted practices

Figure 4-8 Password Profiles task

Figure 4-9 on page 87 shows the Password Profiles panel and is where the Administrator defines the password rules for each role or user:

- Rule Data** Applies to the overall password rule itself. It is the length requirements for the password. It is how many times you can have a consecutive character in your password, and how many password change iterations before you can reuse a password.
- Rule Parts** The Rule Parts list includes the optional set of rules that you can define for the individual parts of the password. Rule Parts govern what the individual characters in the supplied password can and cannot contain. It has positional logic checking, which means that character one can be a number, character 3 cannot be a special character, character 8 must contain the & character, and so forth. There can be a different rule part for every character that is required in a password.
- Property Data** The rules that apply to individual characters in the supplied password. In Figure 4-9 on page 87, Rule Part 0 designates that the first four characters up to the first eight characters can be alphabetic, and can be numeric but cannot be special characters.

Figure 4-9 on page 87 and Figure 4-10 on page 88 specifies:

- Rule Data says the password has to be between four and eight characters.
- Property Data says that the first character in the password cannot be a special character, and cannot be numeric. It must be alphabetic. Notice the Part 1 and Part 2 elements in the two figures.

- Property Data says that the second character in the password can be numeric, cannot be a special character, and cannot be alphabetic.

The Property Data also specifies that there is a 1 for minimum and 4 for the maximum, which means, for at least one character and up to the next four characters in the password can be numeric.

- The rest of the characters are *do not care* in the Merwyn rule in this example.

Using these rules:

a2bc --> is valid

2abc --> not valid

22bc -> not valid

a234 -> is valid because there are no specific rules that govern the contents of the last two characters.

HMCLINUX: Password Profiles - Mozilla Firefox: IBM Edition

https://9.60.15.40/hmc/wd/T251

Password Profiles

Password Rules

- Basic
- Merwyn**
- Standard
- Strict

New Rule...
Delete Rule

Rule Data

Expiration day(s): 0
Minimum length: 4
Maximum length: 8
Consecutive characters: 0
Similarity count: 0
History count: 0
☐ Case sensitive

Password Rule Parts

- Rule Part 0
- Rule Part 1**

Property Data

Minimum characters: 1
Maximum characters: 1
Alphabetic: must
Numeric: not
Special: not
Specific property: not
Specific property: not

Defined Properties

New Property
Delete Property

OK Cancel Help

Done 9.60.15.40

Figure 4-9 Password Profile example 1

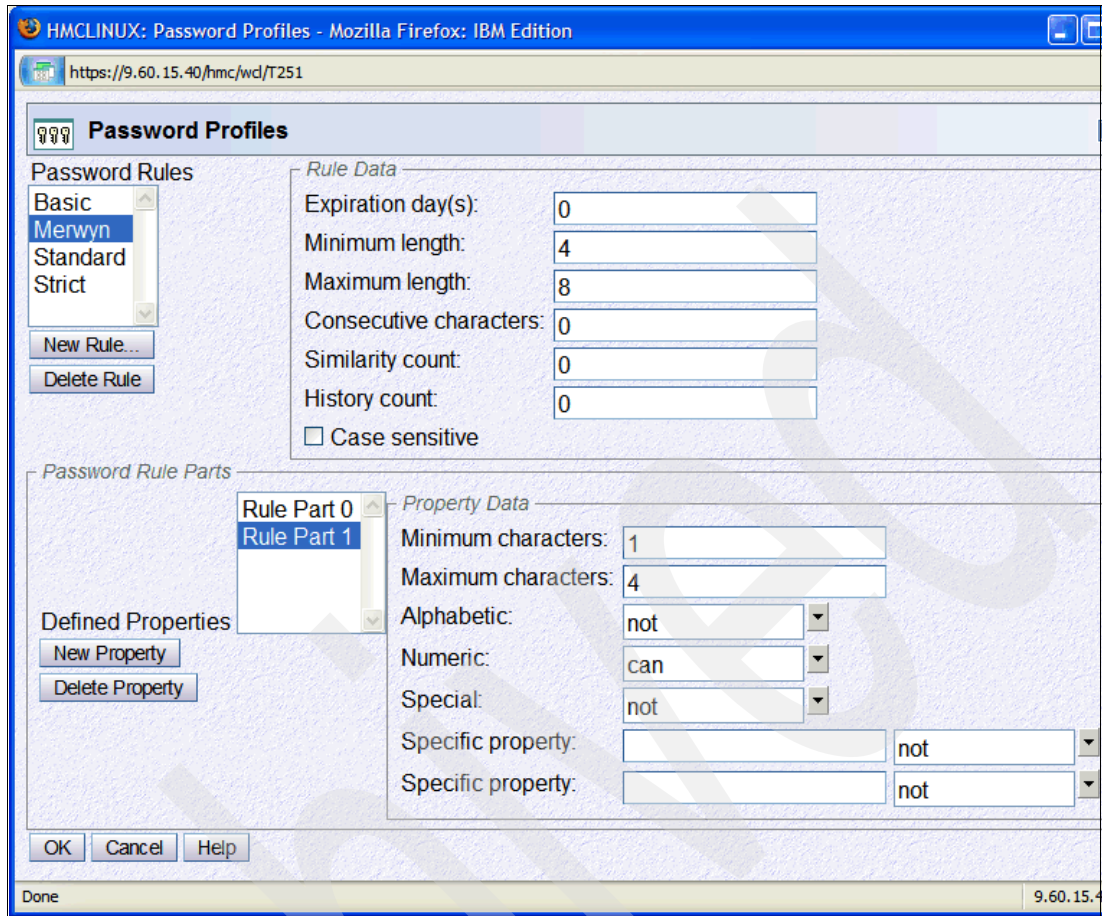


Figure 4-10 Password Profiles example 2 (partial panel)

4.7 Customize User Controls task

By default, the Administrator is the only user who has access to the Customize User Controls task, shown in Figure 4-11 on page 89. As roles are tailored and users are added, individual users, in addition to the Administrators, can be given access to this task. An Access Admin can access this task by logging on to the system, and the Customize User Controls icon is displayed in the Console Actions Work Area.

Use the Customize User Controls task to define and customize managed resource roles and task roles. An Administrator can customize HMC Task Roles and Managed Resource Roles through the HMC console. You can add new Task Roles and Managed Resource Roles based on existing roles in the HMC. You cannot modify system-defined roles; however, you can create a new role based on system defined roles or existing roles. Each role must be based on another role, and the newly defined role cannot have more power than the based-on role.

Managed Resource Roles contain Managed Resource Groups. These groups are created using the Grouping Task on the Console Actions.

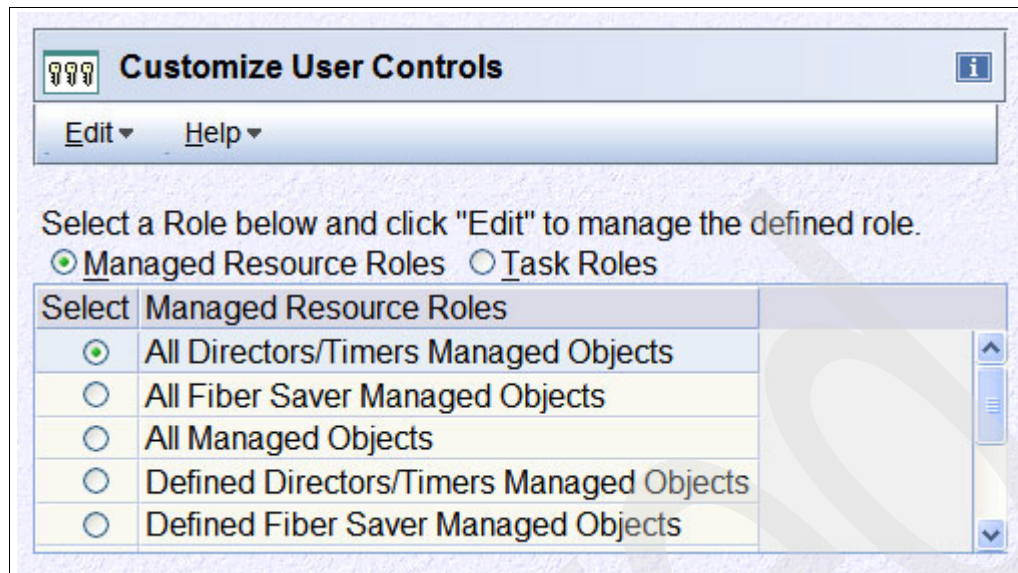


Figure 4-11 Customize User Control window

4.7.1 Creating, copying, modifying, or deleting Managed Resource Roles

In a Managed Resource Role, you can define HMC access to specific managed resources, which might be a complete system.

You can create a new Managed Resource Role, copy an existing Managed Resource Role, modify existing Managed Resource Role, or delete an existing Managed Resource Role from the Customize User Controls window. Select **Managed Resource Roles**, and then select the desired operation from the Edit menu.

Some of the predefined Managed Resource Roles are:

- ▶ All Directors/timers Managed Objects
- ▶ All Managed Objects
- ▶ Defined Fiber Saver Managed Objects
- ▶ Limited Managed Objects
- ▶ z/VM Virtual Machine Objects

4.7.2 Creating new Managed Resource Roles

To create a new Managed Resource Role:

1. From the Customize User Controls task, select **Edit** → **Add**. The Add Role window is displayed.
2. Enter the name for the new Managed Resource Role, and choose the Managed Resource Role from which the new Managed Resource Role objects will be based.
3. Select which object to be available for the new Managed Resource Role, and then click **Add** to add them to the new Managed Resource Role current objects.
4. Click **OK**. A new Managed Resource Role is created.

Figure 4-12 on page 90 shows the Add Role panel that is used to create a new Managed Resource Role. You can modify the Role Resource Groups by selecting a Managed Resource Group in the left tree and adding it to the right tree, which creates a new Role that is based on the selected “based on” role and has the items listed in the current resources tree.

Add moves the selected task and its Managed Resources to the Current Objects window, for example, selecting the **Defined CPCs Resource Group** and selecting **Add** moves the entire Defined CPCs tree to the Current Objects view.

If the Administrator only selects the CPC01 managed object and selects **Add**, the CPC01 is moved to the Current Objects view. All of the remaining objects in the Defined CPCs tree are not added unless the user decides to add them, thus an Administrator can make a subset of Managed Resource Groups and Managed Resources rather than accepting all predefined tasks for each Resource Group. The Add button copies the selected value to the Current Objects tree, which is a representation of the new role being created. The Remove button deletes a selected item in the Current Objects tree.

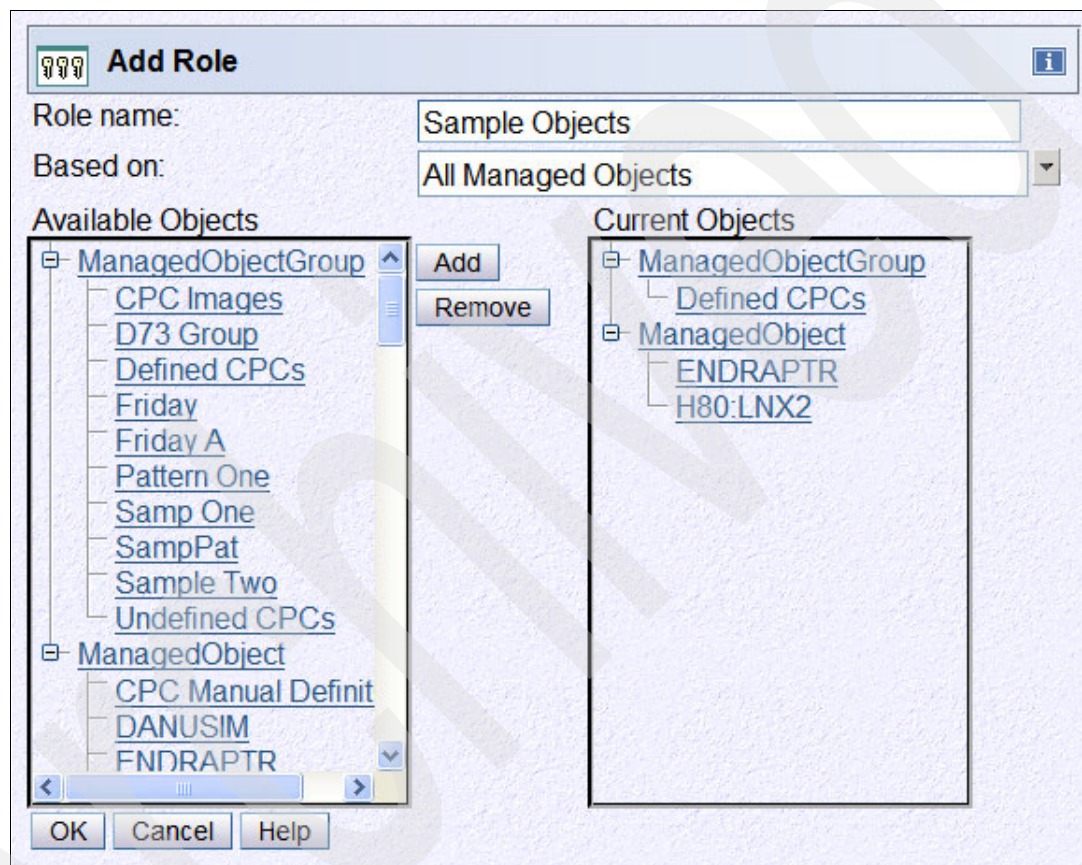


Figure 4-12 Add new Managed Resource Role

The Copy feature is similar to the Add panel in Figure 4-12. To copy a Managed Resource Role, select the desired Managed Resource Role, and select then select Edit/Copy. You cannot copy a user-defined managed system role that was created from the Add menu; however, you can copy system-defined Managed Resource Roles, such as Limited Managed Objects, from the Copy role window. You can customize the object configurations for a new copy of the Managed Resource Role.

To delete a Managed Resource Role, select the desired Managed Resource Role, and select Edit/Remove.

To modify an existing Managed Resource Role, select the Managed Resource Role that you want to change, and then select Edit/Modify. You can change the objects' configuration, and then click **OK** to save the changes.

4.7.3 Creating, copying, modifying, or deleting task roles

A task role defines the access level for a user to perform tasks on the managed object or group of objects, such as a managed system or logical partition. There are many system defined task roles.

Some of the predefined roles are:

- ▶ Access Administrator Director/Timer Tasks
- ▶ Access Administrator Fiber Saver Tasks
- ▶ Access Administrator Tasks
- ▶ Advanced Operator Tasks
- ▶ Service Representative Tasks
- ▶ System Programmer Tasks

You can create a new task role, copy an existing task role, modify an existing task role, or delete an existing task role from the **Customize User Controls** window, shown in Figure 4-13 on page 92. You cannot modify or remove system-defined task roles. You modify and remove system-defined task roles by selecting **Task Roles**, and then selecting the desired operation from the **Edit** menu.

Use the dialog shown in Figure 4-13 on page 92 to add a new role that contains a set of tasks, which can be issued against these Resources. Each new role must be based on another role, and the newly defined role cannot be more powerful than the based-on role. You can modify the Task Role by selecting an Available Task in the left tree and adding it to the right tree on the panel, which creates a new Role, which is based on the selected “based-on” role and has the items listed in the current task’s tree.

You cannot interchange Tasks and Task Groups. Add moves the selected task and its Task Group to the Current Tasks window, for example, selecting the **Daily Task Group** and selecting **Add** moves the entire Daily tree to the Current Tasks view. If the Administrator only selects the **Grouping task** and selects **Add**, the Grouping task and the Daily Task Group are moved to the Current Tasks view.

All of the remaining tasks in the Daily Task Group are not added unless you decide to add them, so that an Administrator can make a subset of Tasks rather than having to accept all predefined tasks for each user. Use the **Add** button to copy the selected value to the Current Tasks tree, which is a representation of the new role being created. The **Remove** button removes a selected item in the Current Tasks tree.

To create a new user task role:

1. From the **Customize User Control** window, select **Edit** → **Add**. The **Add Role** window is displayed, as shown in Figure 4-13 on page 92.
2. Enter the name for the new task role, and choose the task role from which the new task role objects will be based.
3. Select which object will be available for the new task role, and then click **Add** to add them to the new task role current objects.
4. Click **OK** to create a new task role.

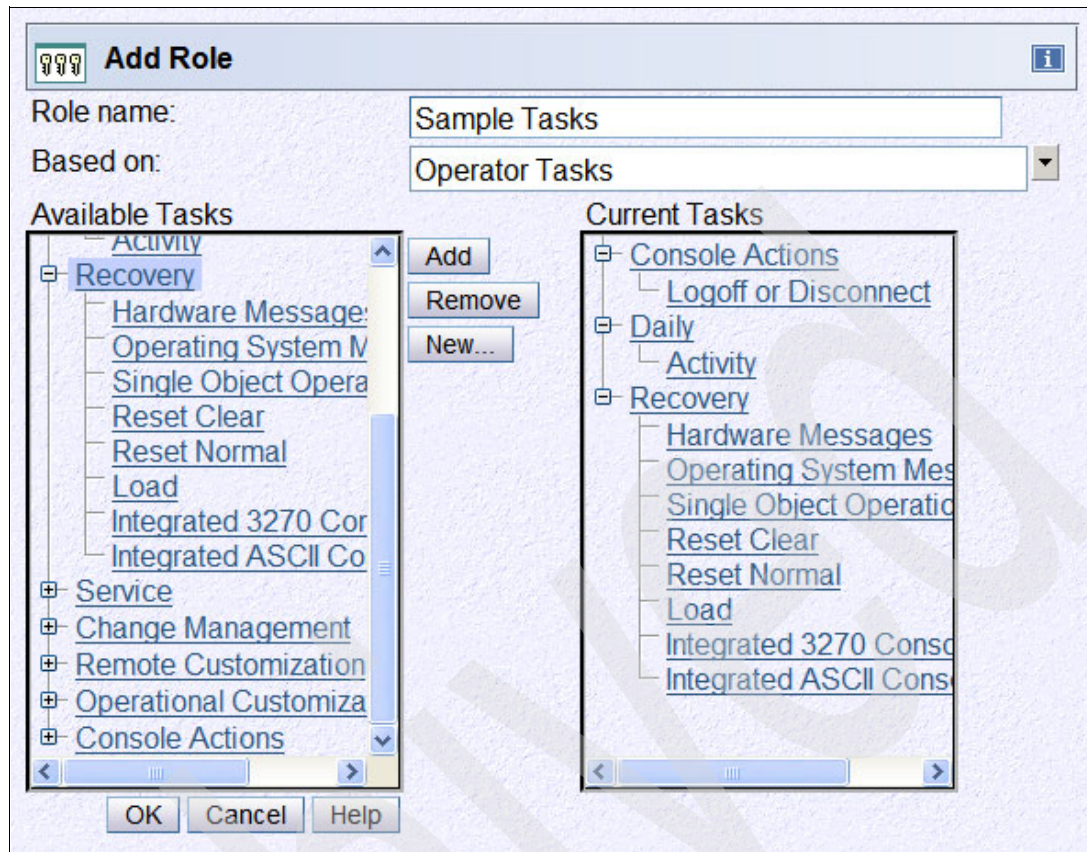


Figure 4-13 Add a task role

To copy a task role, select the desired task role, and then select **Edit/Copy**. From the Copy Role window, you can also customize the object configurations for a copy of the task role.

To delete a task role, select the desired task role, and then select **Edit/Remove**. A verification window is displayed. You cannot remove system-defined task roles.

To modify existing task roles, select a task role that you want to change, and select **Edit/Modify**. You can change the objects' configuration, and then click **OK** to save the changes. Only user-defined task roles that are created by HMC Administrators can be modified.

4.8 Managing users with data replication

Using the Customizable Data Replication task you can configure a set of HMCs to automatically replicate any changes to certain types of data so that the configured set of HMCs keep this data synchronized without manual intervention. When replication is configured properly, any changes that are made to a user, user roles, or passwords on your master HMC can be effortlessly replicated to any subordinate machines.

To manage users with data replication:

1. Simply invoke the HMC, and open the Hardware Management Console Settings task in the Console Actions Work Area.

2. Select the **Configure Data Replication task**, shown in Figure 4-14. Selecting the User Profile Data datatype ensures that your user management changes are replicated to the slaves that you defined for your system.

Configure Customizable Data Replication

Customizable Data Replication

☒ Enable ☐ Disable

Data Source(s)

LINSUMFI

New Delete

Customizable Data Types

Select	Data Types
<input type="checkbox"/>	Customer Information Data
<input type="checkbox"/>	Group Data
<input type="checkbox"/>	Remote Service Data
<input checked="" type="checkbox"/>	User Profile Data
<input type="checkbox"/>	Monitor System Events Data

Local Customizable Data Change Warnings

Select the customizable data types that should generate warnings when that type of data is manually changed on this Hardware Management Console and are also configured to be replicated from one or more data sources.

Select	Data Warning Types
<input type="checkbox"/>	Customer Information Data
<input type="checkbox"/>	Group Data
<input type="checkbox"/>	Remote Service Data
<input type="checkbox"/>	User Profile Data
<input type="checkbox"/>	Monitor System Events Data

Save Push to Slaves Sync from Master Status Cancel Help

Figure 4-14 Data Replication window

4.9 User settings

If permitted, users can tailor their individual settings on their HMC. Items, such as the default UI style, confirmation settings, colors and patterns, single object selection settings, and hover help setting can be set on a user-by-user basis. Figure 4-15 on page 94 through Figure 4-18 on page 96 show the various tabs in User Settings under Console Actions. We discussed UI styles in Chapter 3, “User experience” on page 49.

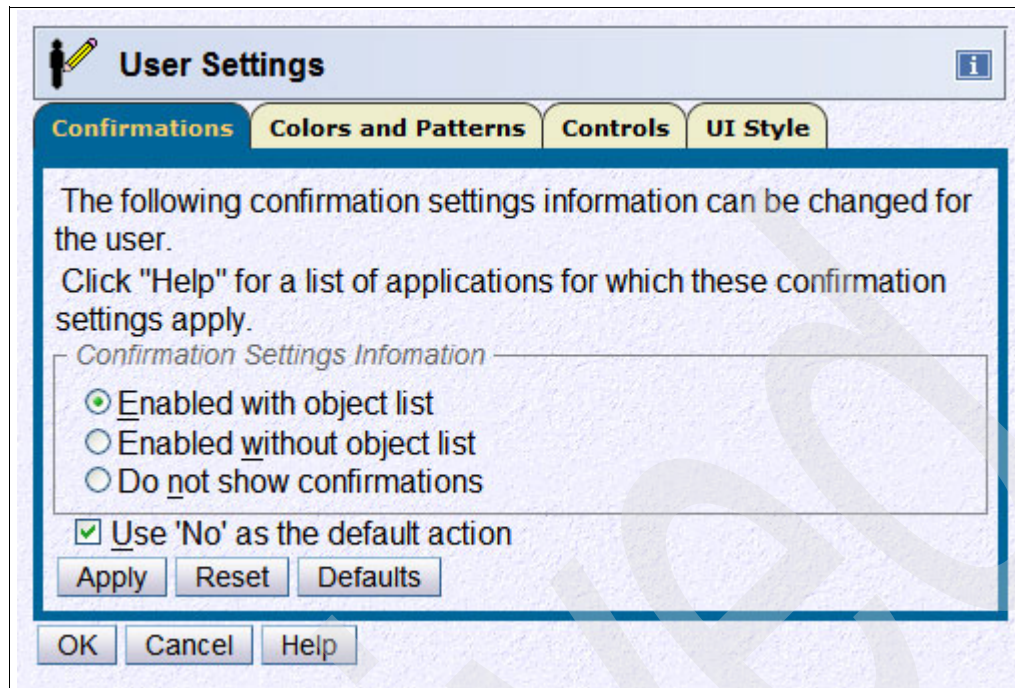


Figure 4-15 User Settings window - 1

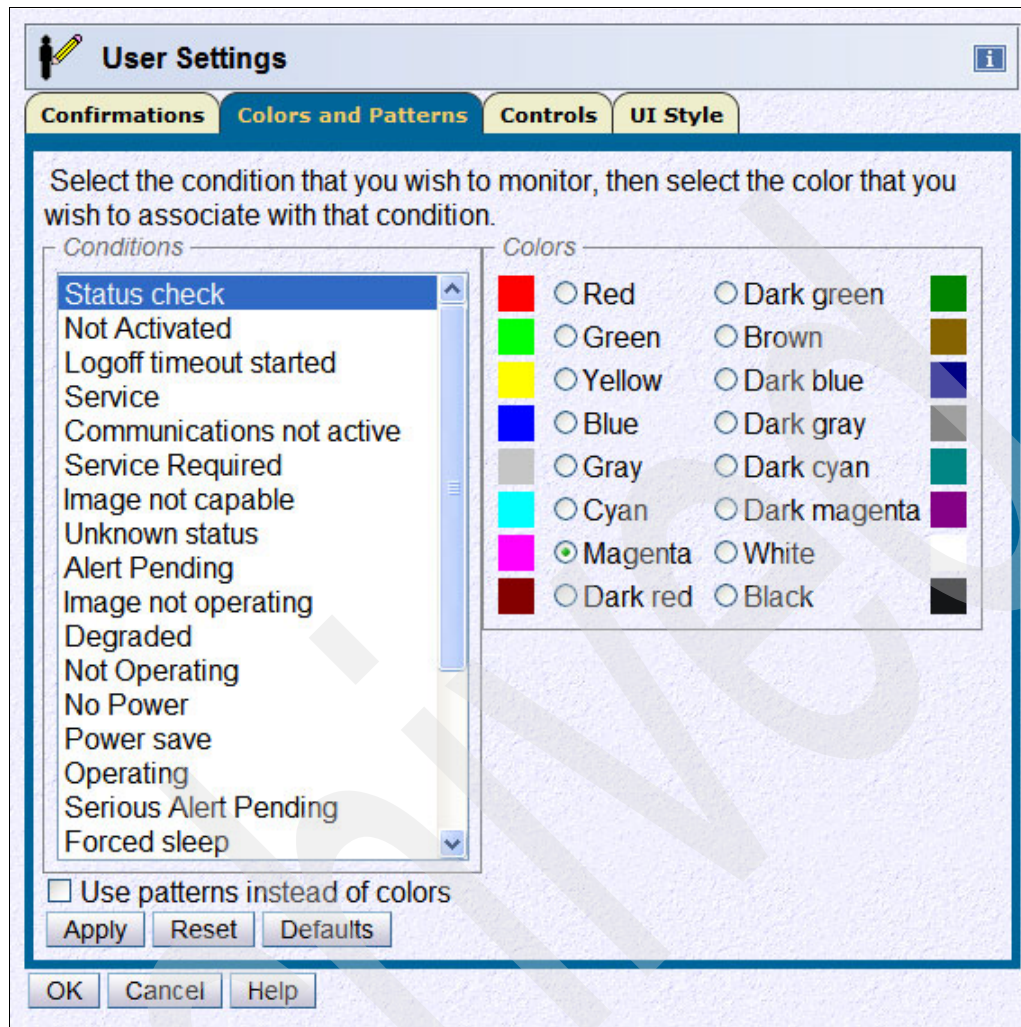


Figure 4-16 User Settings window - 2

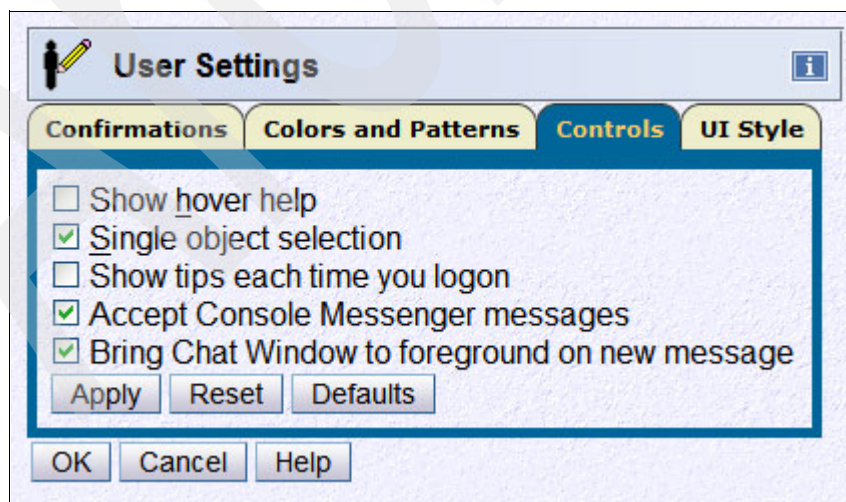


Figure 4-17 User Settings window - 3

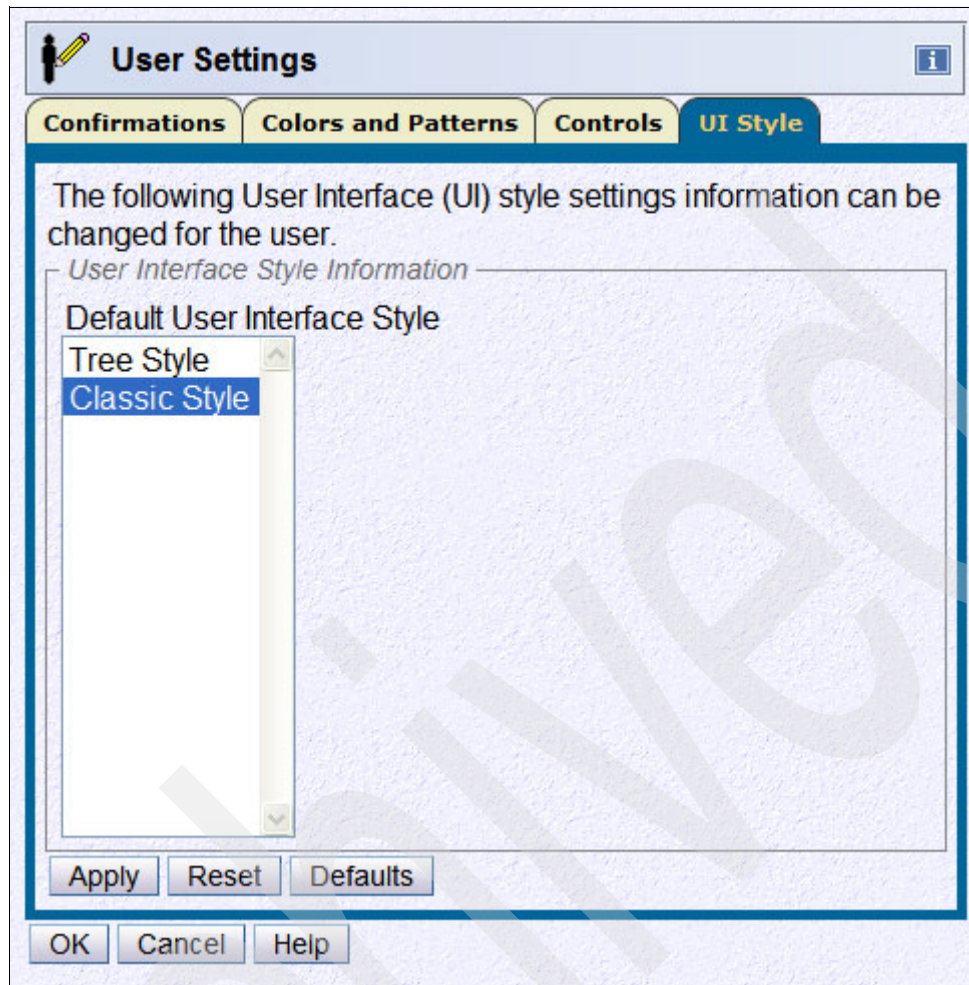


Figure 4-18 User Settings window - 4

4.10 Questions and discussion

1. This chapter is about user management. Which users are being considered?
2. We sometimes see elaborate password rules, which might require at least one number, at least one special character, a mixture of upper case and lower case, prohibit more than two sequential characters that are adjacent on the keyboard, and so forth. What are some of the problems with these rules?
3. Oddly enough, many HMC systems have practically no encryption or firewalls or passwords involved. Why is this?
4. Who establishes new HMC user IDs, sets passwords, determines privilege sets, and so forth? Is this the same person who performs these services for the general user community?
5. In a typical mainframe environment, would at least one HMC user be normally logged on to the HMC?
6. Can the HMC console be combined with, for example, the z/OS operator console? How is security assigned in this situation?

Hardware Management Console remote access

The Hardware Management Console (HMC) is a complex element that communicates at many levels with multiple mainframes in the enterprise. Given this responsibility the HMC manages what users are allowed to access its function and then based on who they are, allows the Administrator to customize what objects, systems, tasks, and status they can see or update. The HMC has a robust set of authentication and authorization functionality for the customers to use to integrate into the enterprise security policy. As such, security plays an important role in the design of the HMC.

In this chapter, we first review several general security topics, and then we discuss specific HMC details.

5.1 Cryptography

We refer to Cryptography as a way of coding data (encryption) for secure transmission to a specific endpoint or user. Cryptography is the basis of all Internet security. Modern cryptography, though, is more than just the study of encryption. Another very important area of cryptography is authentication, which is the process of verifying that the user who requests data is who he or she says he or she is. This area is every bit as important as encryption to ensure that only the HMC-authorized user is allowed on the HMC.

One must understand the importance of cryptography in our modern world. The services that are available to customers online are rapidly increasing. We can now perform Web banking, Internet shopping, book a flight, rent a car, and so forth. Because all of this can be done over the Internet, the chances that someone is attempting to listen to transactions and steal data that does not belong to them has increased significantly.

Cryptography is an ever evolving field because cryptography is accomplished by basing the encryption on a *current* problem that is difficult to solve, for example, a DES 56-bit key would take years to crack using the fastest personal computer. However, the Internet and grid computing made it possible to link thousands of machines together to break down the problem into smaller parts, and using this method, the DES 56-bit key was cracked in just 22

hours. Today 56-bit DES is considered a relatively weak encryption method because there are much more elaborate encryption schemes, which we review later. Although the problem seems almost impossible to solve in a feasible amount of time, as time goes on, new technology provides ways to counter this, which is why the study of cryptography and its use are constantly being researched and reinforced.

5.1.1 Encryption

Encryption is, taking data and changing it in a certain way that only the user who receives the data knows how to decipher it, which is commonly done with some form of a key. Deciphering the data requires that both users know the key, to first encode the information, transfer it, and then decode it. This process is referred to as *private-key cryptography*, where both users need to somehow share their key with each other.

The flaw in private-key cryptography is that the sharing of the key needs to be secure. This method works great if the private key can be sent from one user to another without another user intercepting the key. But in a situation where the key needs to be communicated across a network, this method becomes somewhat flawed. *Public-key cryptography* addresses this issue and provides a means to securely share data over a non-secured network.

5.1.2 Public-key cryptography

Public key cryptography removes the need to communicate a private key between users by assigning each user a public key that is available to anyone who wants to encrypt and send a message to that user. The user who receives the message then decrypts the message using their private key that only they have access to. Using this method a system can send messages to multiple users without ever having to share a private key or in this case without even knowing who the user is prior to the exchange, which allows communications across networks to be secure.

The flaw in public key cryptography is that because each user's private key is linked mathematically to the public key, and the private key can be derived from the public key with enough processing power. The solution to that problem is to make the private key as hard as possible to derive. It is very unlikely that it can be cracked, although theoretically possible. Cracking the private key from the public key usually involves factoring a very large number, which is unfeasible with today's computing power, which is why public-key cryptography is currently viewed as secure. Although the connection is now secure in terms of encryption, authentication (identifying that a user is who they say they are) still forms a security flaw. Certificates, which we discuss in 5.2.3, "Certificates" on page 99, addresses the authentication flow.

5.2 Remote access security

Remote access is the process of using a machine to access an application, resource, or service of another machine that might or might not be at the same physical location. Practical remote access involves a number of security elements, and we provide a brief description of the key elements in this section.

5.2.1 Diffie-Hellman key agreement protocol

The Diffie-Hellman key agreement protocol provides a way for two users to generate the same private key without ever having to share that key. As an example, two users, Alice and

Bob, need to generate a secret key. Alice and Bob each generate a private value, called a and b . The public values are derived from these private ones using the formula $g^a \equiv \text{mod } p$ and $g^b \equiv \text{mod } p$ where g and p are integers less than p with the property $g^k \equiv \text{mod } p$. The public values are exchanged, and each party will then use the public value to compute the secret key, following the formula $g^{a(b)} \equiv \text{mod } p$ and $g^{b(a)} \equiv \text{mod } p$ respectively, which generates the same private key k for both users without ever having to send the key.

Because the private key is linked mathematically, we have the reoccurring threat that it can be factored and computed. But as was shown, it is not feasible at this point in computer technology to try to factor the prime number p . However, this protocol is vulnerable to a difficult type of attack called the *man-in-the-middle attack*.

An example of the man-in the middle attack is, Charlie intercepts both Alice's public value and Bob's public value and generates the secret key using his public/private value combo, while at the same time sending Alice's information to Bob and Bob's information to Alice. Because both parties are receiving their information, the attack is invisible and Charlie can intercept every message, decode it, and then recode it using the appropriate values and send it to the intended receiver.

The solution to the man in the middle problem is signing the messages, which requires each users private key. Charlie could intercept, but without each users private key (which he can not get because they never share their private values) he cannot forge signatures on the messages, which defeats the man-in-the-middle attack.

5.2.2 Secure communication

Secure communication across the network is a key capability of the HMC. The concept of secure communication is a very simple application of cryptography. It is the process of one person sharing a message to another in a way that anyone eavesdropping cannot decipher it. This method obviously occurs through the use of encryption and keys. The problem with this standard approach (as discussed previously) is that the private key needs to somehow be shared between users, and if that is intercepted the entire system falls apart. But, as we already discussed, public key cryptography fixes the issue.

5.2.3 Certificates

The problem of someone using a phony key to impersonate someone else is solved using *certificates*. Certificates contain information that can identify a user. Certificates must contain a name with a public key, an expiration date for the certificate, a serial key, the name of the party authorizing the certificate, and so on. The most widely accepted form of certificates is the X.509 international standard. We go into more detail about how certification is done shortly. For now, the basic purpose of the certificate is to match a public key to a user.

Sometimes a chain of certificates are issued, each of which verifies the previous. This method allows the party that is issuing the certificates to reach a desired level of confidence that the user is who they say they are, for example, when a certificate is received, it must be verified. The certificate authority is checked against the local verifier's database to determine if the authority is trusted. If the certificate authority is trusted, the certificate is trusted; otherwise, the certificate is not trusted, a message is displayed to the user, and a prompt is displayed if they want to manually trust this certificate. After a certificate is trusted, additional certificates (that use the original certificate as the certificate authority) can now be trusted. Therefore, a chain of certificates can be made when each certificate verifies the next certificate in the chain.

5.2.4 Firewalls

A firewall is a set of security measures that are designed to prevent unauthorized access to a computer network. All traffic that enters and leaves the network passes through the firewall and is blocked if it does not meet the required security criteria.

5.3 Secure Socket Layer (SSL)

Netscape developed Secure Sockets Layer (SSL) protocol for authenticated and encrypted communication between clients and servers. SSL is above the TCP/IP protocol and below higher-level protocols, such as HTTP, LDAP, and IMAP. SSL uses TCP/IP to establish an encrypted connection from an SSL-enabled server to a SSL-enabled client, which is accomplished by:

1. The server authenticates to the client.
2. The client is then allowed to authenticate itself back to the server, typically by user name and password. Public-key cryptography is used by both the SSL server and client. The keys are checked, on both the client and the server, against a list of trusted certificates except when a user name and password is used to authenticate the client.
3. After the server and client establish that the other is trusted, an encrypted SSL connection is made. At this point, the client knows that it is connected to the authentic server and the server knows it is connected with the authorized user/client.

All information that is sent over an SSL connection is encrypted then decrypted, which provides a level of confidentiality in the data. A mechanism is built into SSL that detects if information was tampered with, which protects against a man in the middle attack. The SSL protocol has two sub protocols:

- ▶ SSL record protocol: Defines the format for data that is sent across encrypted connections
- ▶ SSL handshake protocol: Establishes authentication

5.3.1 Hypertext Transfer Protocol Secure

Hypertext Transfer Protocol Secure (HTTPS) is a combination of Hypertext Transfer Protocol and a network security protocol (SSL). It is invoked by using https:// instead of http:// when accessing a Web site. A server must create a public-key certificate to issue to clients to accept an HTTPS connections.

With this short security technology review, let us understand how the HMC uses these technologies to meet the enterprise security policy for remote access. An HMC is the highest-level of control for a group of mainframes. HMC security is critical to implementing the enterprise security policy. In many cases, the first element of the security policy is to address physical access, that is, the HMC must be in a closely-controlled area where few people can access it and all traffic to the secure area is monitored, authorized, and logged.

However, with the increased use of *lights-out* mainframe data centers and remote backup centers, customers require network access to the HMC for their Operators and System Programmers. The objective of HMC's security functions is to secure the HMC and allow network access to the HMC (HMC Remote Access).

5.3.2 HMC certificate management

All remote browser access to the Hardware Management Consoles must use SSL encryption. With SSL encryption required for all remote access to the Hardware Management Console, a certificate is required to provide the keys for this encryption. The Hardware Management Console provides a self-signed certificate that allows this encryption to occur.

Use the HMC *Certificate Management Task* to manage the certificates that are used on your Hardware Management Console. The HMC Certificate Management Task provides the capability of getting information about the certificates used on the console. Using this task you can create a new certificate for the console, change the property values of the certificate, work with existing and archived certificates, and sign certificates.

To use the HMC Certificate Management Task:

1. Open the Certificate Management task. The Certificate Management window is displayed, as shown in Figure 5-1.

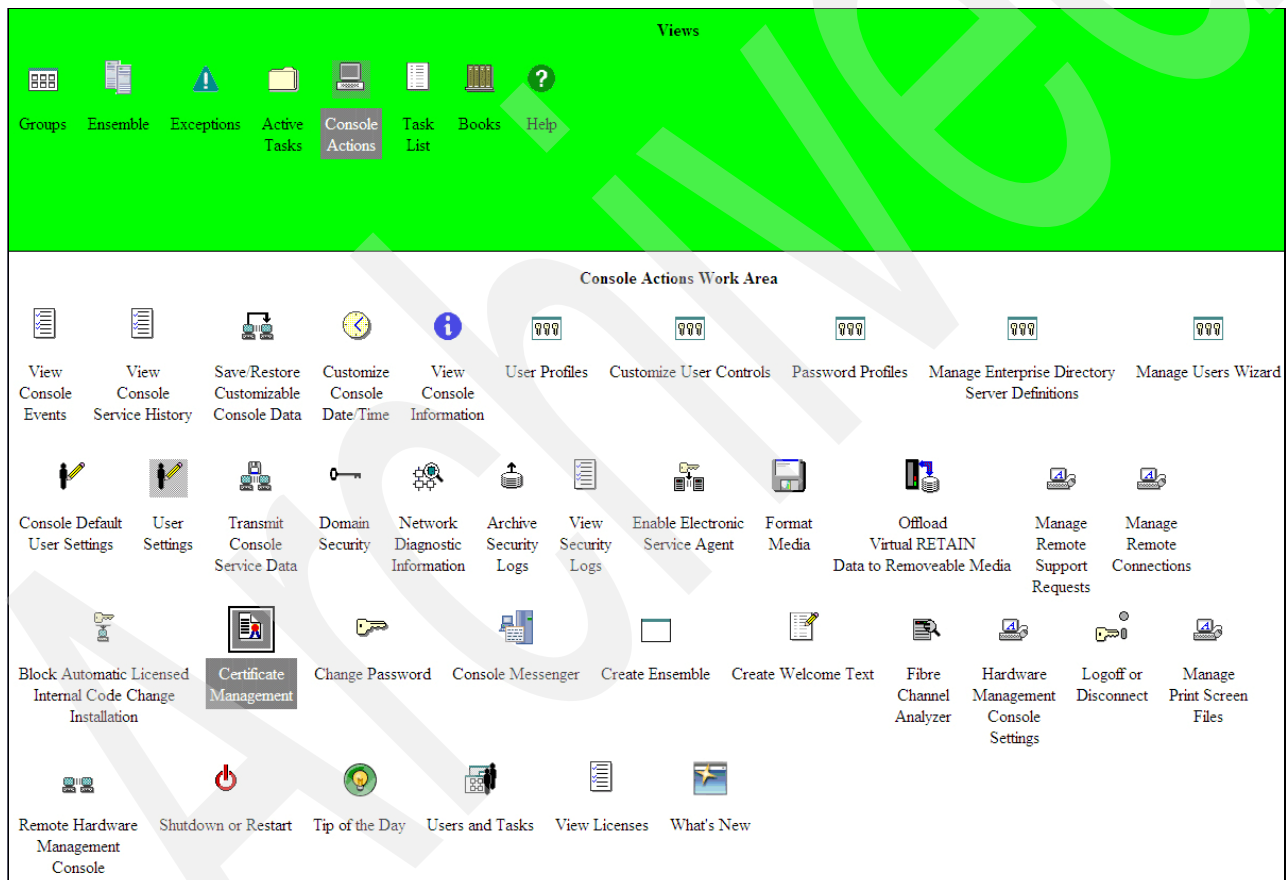


Figure 5-1 Certificate Management window

2. Use the menu bar from the Certificate Management window for the actions that you want to take with the certificates.
3. To create a new certificate for the console, click **Create** → **New Certificate**, as shown in Figure 5-2 on page 102.

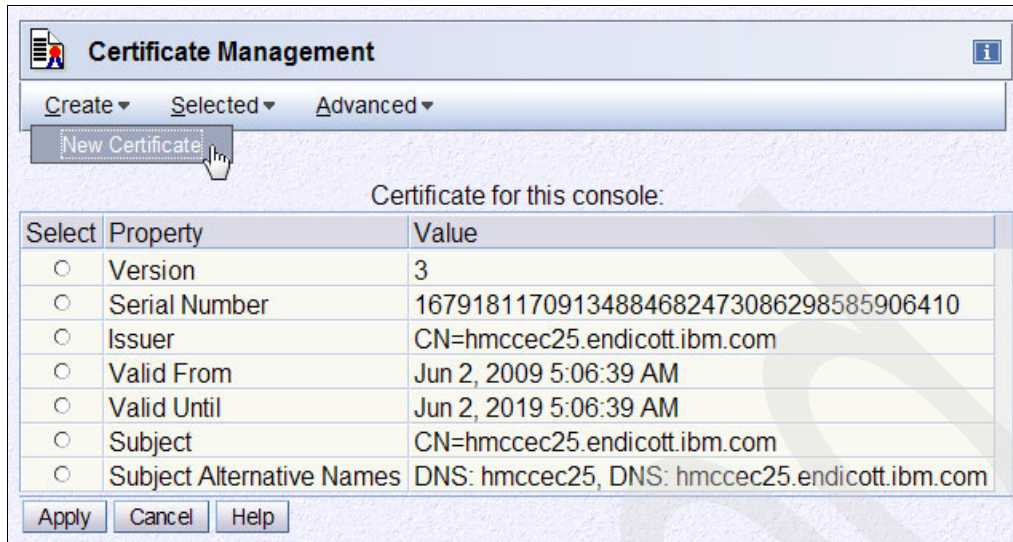


Figure 5-2 Certificate Management window - 2

4. Determine whether your certificate will be self-signed or signed by a Certificate Authority, and then click **OK**, as shown in Figure 5-3.

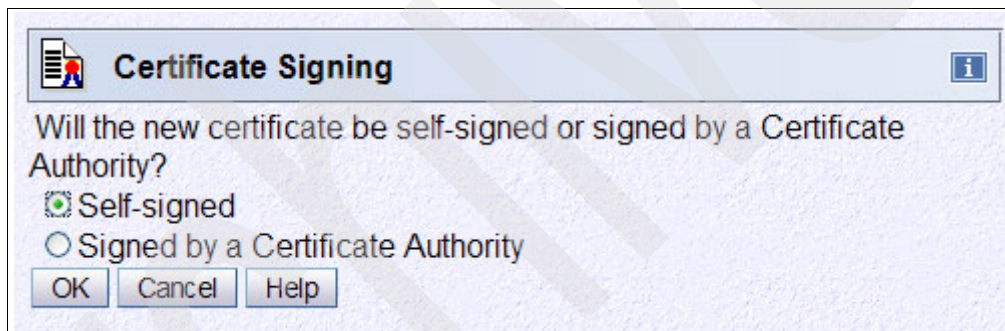


Figure 5-3 Certificate Signing window

5. To modify the property values of the self-signed certificate, from the Certificate Management Panel click **Selected** → **Modify**. The Modify Certificate window is displayed, as shown in Figure 5-4 on page 103. Make the appropriate changes, and then click **OK**.

Modify Certificate

Modify the following information appropriately:

Organization (e.g. IBM)

Organization unit (e.g. Hardware Development)

Two letter country or region code (e.g. US)

State or Province (e.g. CA)

Locality (e.g. Los Angeles)

E-mail address (e.g. xxxx@ibm.com)

Figure 5-4 Modify Certificate window

6. To work with existing and archived certificates or signing certificates, from the Certificate Management Panel, click **Advanced**, as shown in Figure 5-5 on page 104. Choose the following options:
 - Delete and archive certificate
 - Work with archived certificate
 - Import certificate
 - Manage trusted signing certificates
 - View issuer certificate

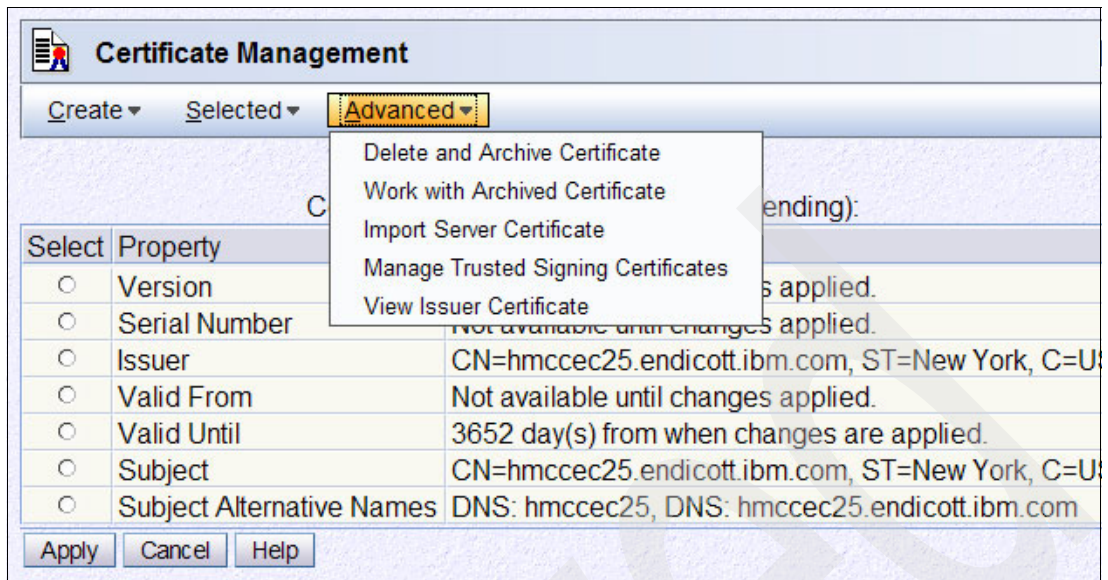


Figure 5-5 Certificate Management window - 2

- Click **Apply** for all changes to take effect, as shown in Figure 5-6.

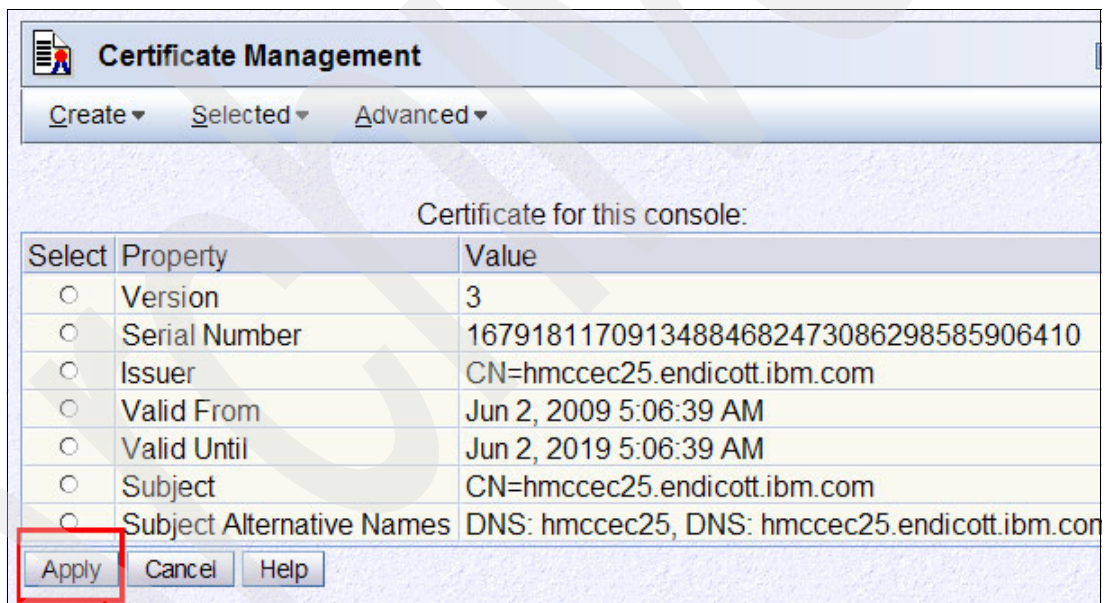


Figure 5-6 Certificate Management window - 3

- Use the online help if you need additional information about managing your HMC certificates.

5.3.3 Allowing HMC user IDs Web access to an HMC

To configure the Hardware Management Console for Web browser access, first you must enable the HMC user ID to have access:

- Log onto the Hardware Management Console with the ACSADMIN default user ID.
- Open the User Profiles task. The User Profiles window is displayed.

3. For each user that you want to allow Web browser access, select the user ID. From the menu bar, select User. When the User menu is displayed, click **Modify**. The Modify User window is displayed.
4. On the Modify User window, click **User Properties**. The User Properties window is displayed.
5. Select **Allow remote access through the Web**, and then click **OK**.
6. Open the Customize Console Services task. The Customize Console Services window is displayed.
7. On the Remote Operation selection, choose **Enabled**. Click **OK**.

5.3.4 Accessing an HMC from another HMC

In the previous section, we discussed how to access an HMC from any workstation using a browser. In this section, we describe how to access an HMC from another HMC.

You can establish a remote session to another Hardware Management Console only if the following conditions are met:

- The Hardware Management Consoles are in the same security domain. To verify the domain, use the Domain Security task. See Figure 5-7 and Figure 5-8 on page 106.

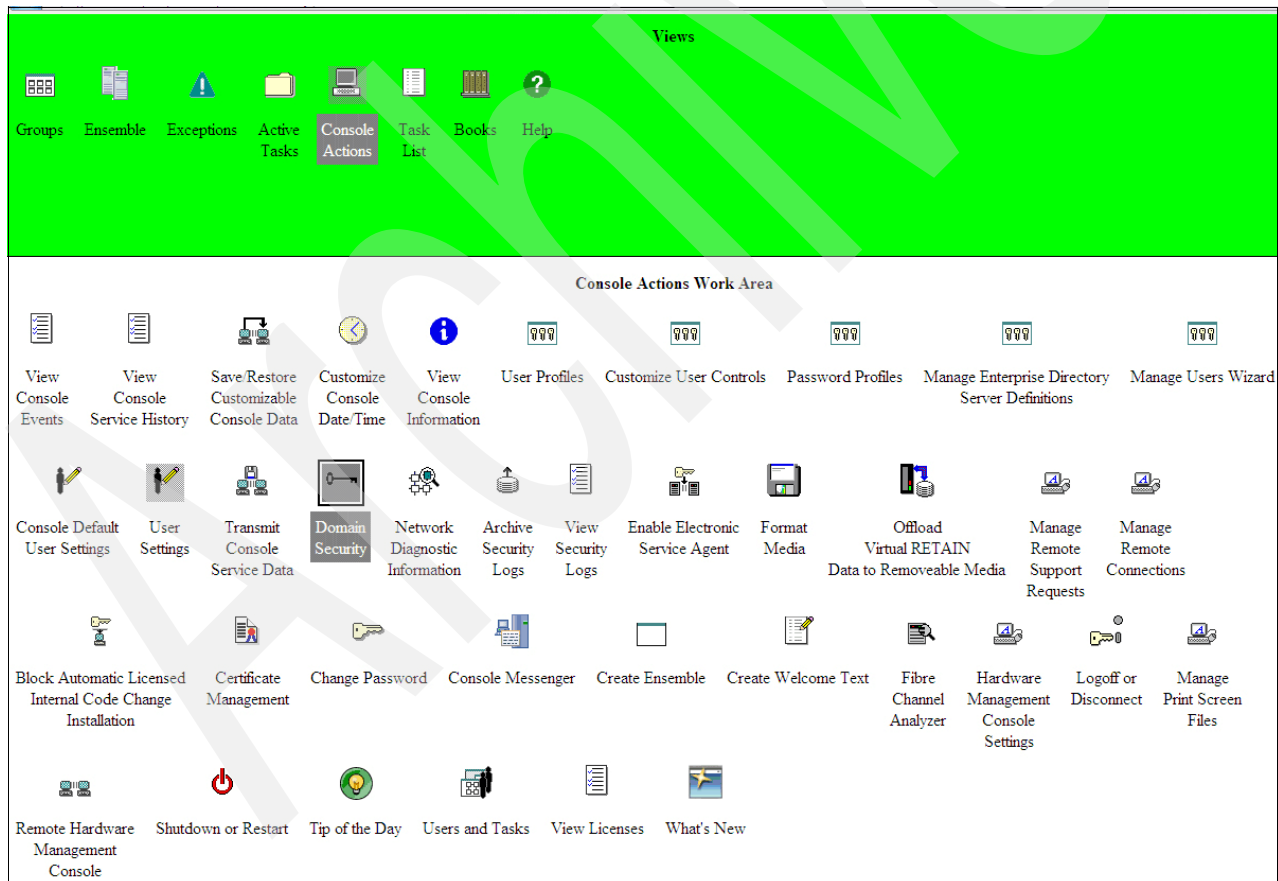


Figure 5-7 Domain Security task window

Domain Security

Current domain name: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Domain name:

New password:

Verify password:

☒ Apply to the Hardware Management Console

☐ Apply to defined objects and the Hardware Management Console

Click "Help" for important information about using this task correctly, and about the consequences of applying a customized domain name or password to this console or its defined objects.

Figure 5-8 Domain Security task passwords

- Remote operation is enabled on the target Hardware Management Console. To enable the target HMC's access through a remote HMC, use the Customize Console Services task. Use this task to allow both browser and HMC access to be performed on the target Hardware Management Console.

To establish a remote session:

1. Open the Remote Hardware Management Console task. The Remote Hardware Management Console Addressing Information window is displayed, as shown in Figure 5-9 on page 107 and Figure 5-10 on page 107.

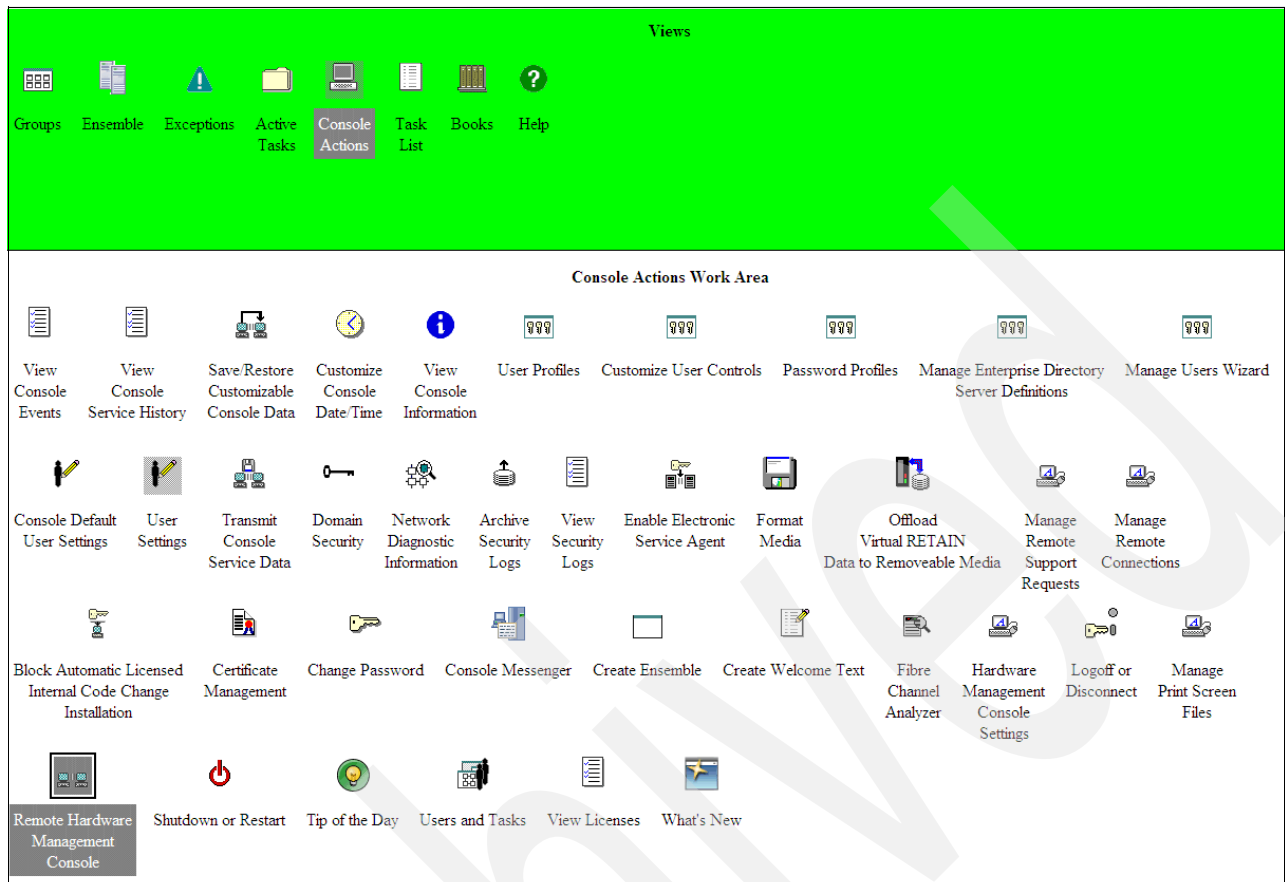


Figure 5-9 Select Remote Hardware Management Console

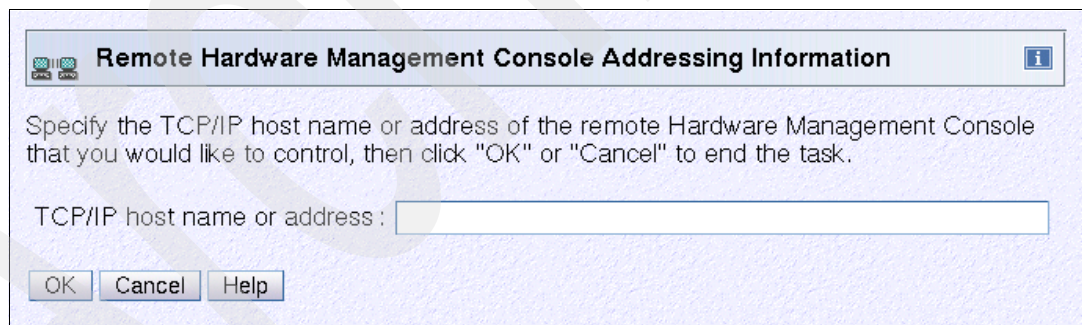


Figure 5-10 Remote HMC Addressing Information

2. Specify the IPv4 or IPv6 TCP/IP address or host name of the remote Hardware Management Console that you want to contact.
3. Click **OK** to complete the task or **Cancel** to exit. Use the online help if you need additional information about contacting another Hardware Management Console.

5.3.5 Using a Web browser to access an HMC

If you need to do occasional monitoring and control of your System z resources using a single local HMC, the remote Web browser is a good choice. An example of using the Web browser

for remote access might be off-hours operation monitoring of system status or system performance from home by a System Programmer.

First the Hardware Management Console must be configured to allow Web browser access (see 5.3.4, “Accessing an HMC from another HMC” on page 105). With this permission the remote user has access to all of the configured functions of a local Hardware Management Console, except those functions that require physical HMC access, for example, functions that use the local diskette or DVD media. The user interface that is presented to the remote Web browser user is the same as the user interface of the local Hardware Management Console.

The Web browser can be connected to the local HMC to be controlled using either a LAN or a switched (dial) network Point-to-Point Protocol (PPP) TCP/IP connection. Both types of connections can only use encrypted (HTTPS) protocol. For a LAN connection, you must have properly setup firewall access between the HMC and the Web browser. If a switched connection is to be used, for optimum security, the PPP login name and password must be configured in the local Hardware Management Console and the remote browser system.

Logon security for a Web user (System Programmer, Operator, and so on) is provided by the Hardware Management Console user logon procedures. You must have a valid user ID and password that is assigned by the Access Administrator for Hardware Management Console Web access. Certificates for secure communications are provided, and can be changed by the user if desired.

Finally, if you are using PPP, the Hardware Management Console must not already be configured as part of a 192.168.33.0/24 subnet to prevent routing problems when remote operation is being used.

5.3.6 4.4.5 Logging into the HMC using the Web

Use the following steps to login to the HMC from a LAN connected Web browser:

1. Ensure that your Web browser PC has LAN connectivity to the desired Hardware Management Console (ping the address).
2. From your Web browser, type the Web address of the desired Hardware Management Console using the format `https://hostname.domain_name` (for example: `https://hmc1.ibm.com`), IPv4 address `https://xxx.xxx.xxx.xxx`, or IPv6 address `https://[xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx]`. If this is the first access of the Hardware Management Console for the current Web browser session, you might receive a certificate error. This certificate error is displayed if:
 - The Web server that is contained in the Hardware Management Console is configured to use a self-signed certificate, and the browser was not configured to trust the Hardware Management Console as an issuer of certificates.
 - The Hardware Management Console is configured to use a certificate that is signed by a Certificate Authority (CA), and the browser was not configured to trust this CA. By default, your browser pre-configures a list of well known CAs. There are several CAs that offer certificates that are not in the pre-configured list.

In either case, if you know that the certificate that is displayed to the browser is the one that the Hardware Management Console used, you can continue, and all communications to the Hardware Management Console are encrypted. If you do not

want to receive notification of the certificate error for the first access of any browser session, you can configure the browser to trust the Hardware Management Console or the CA. In general, to configure the browser:

- You must indicate that the browser will permanently trust the issuer of the certificate.
 - By viewing the certificate and installing to the database of trusted CAs the certificate of the CA that issued the certificate that the HMC uses. If the certificate is self-signed, the HMC itself is considered the CA that issued the certificate.
3. Click **Log on and launch the Hardware Management Console Web application**.
 4. When prompted, type the user name and password that your Access Administrator assigned.

5.4 HMC User authentication

To logon to the Hardware Management Console, click **Logon**, and launch the Hardware Management Console Web application from the Welcome window. Default user IDs and passwords are established as part of a base Hardware Management Console. The Access Administrator must assign new user IDs and passwords for each user and remove the default user IDs as soon as the Hardware Management Console is installed by using the User Profiles task or the Manage Users Wizard. Table 5-1 shows the predefined default user roles, user IDs, and passwords.

Table 5-1 Standard user IDs

Type of user	User ID	Default password
Normal operator	OPERATOR	PASSWORD
Advanced operator	ADVANCED	PASSWORD
System programmer	SYSPROG	PASSWORD
Access administrator	ACSADMIN	PASSWORD

Letter case (uppercase, lowercase, mixed) is not significant for the default user IDs or passwords. The default passwords must be changed to provide any degree of security.

To logon:

1. Enter one of the default user ID and password combinations, the user ID and password combination assigned to you by your Access Administrator, or your LDAP user ID.
2. Click **Logon**. If you are accessing the HMC remotely the Logon button might initially be disabled until the Logon window completely loads. Use the online help if you need additional information by clicking **Help** from the Logon window.

After you logon, the Hardware Management Console Workplace window is displayed. If enabled, the Tip of the Day window is displayed. From the Hardware Management Console Workplace window, you can work with tasks for your console and CPCs (servers). Not all tasks are available for each user ID. If at any time you do not know or remember what user ID is currently logged on to the Hardware Management Console, click the user ID that is located on the Task bar in the tree style user interface, or open the Users and Tasks task in the classic style user interface.

5.4.1 4.5.1 LDAP support for user authentication

Using the LDAP support you have the option to configure your Hardware Management Console to use an LDAP server to perform user ID and password authentications. An LDAP server maintains a tree-structured database that serves as a convenient place to put hierarchical information, such as a corporate employee directory. Each level of the LDAP tree generally represents a different type of information.

LDAP support for Hardware Management Console user authentication allows a Hardware Management Console to be configured to use an LDAP server to perform user ID and password authentication. The user ID is defined on the Hardware Management Console along with the roles to be given to the user ID (see the User Profiles task). The Hardware Management Console settings that are related to the user ID will continue to reside on the Hardware Management Console, and the LDAP directory is used to authenticate the user, thus eliminating the need to store the user ID's password locally.

There are several benefits to creating new enterprise (LDAP) definitions or editing and removing existing enterprise directory server definitions. Creating and editing LDAP definitions is designed to more easily assist System Administrators in creating Hardware Management Console user IDs, matching existing company user names, and eliminating the need to create and distribute passwords. Both SSL and non-SSL connections to the LDAP server are supported.

Using the User Profiles task you choose which type of server authentication you prefer: Local Server or LDAP Server. To add, edit, or remove an enterprise directory (LDAP) server:

1. Open the Manage Enterprise Directory Server Definitions task. The Manage Enterprise Directory Server Definitions window is displayed, as shown in Figure 5-11.

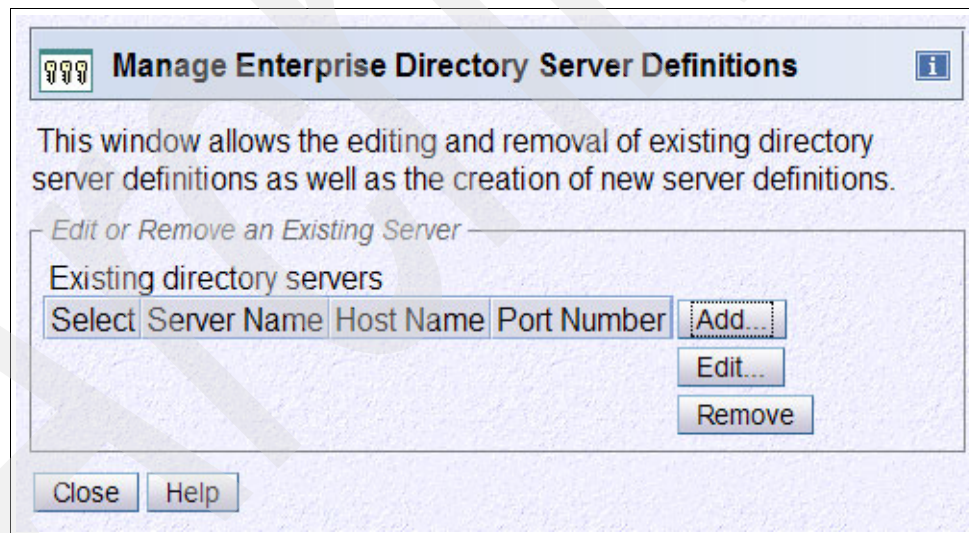


Figure 5-11 Manage Enterprise Directory Server Definitions

2. To add a server, click **Add**. The Add Enterprise Directory (LDAP) Server window is displayed. Provide the appropriate information, and then click **OK**, as shown in Figure 5-12 on page 111.

Figure 5-12 Add Enterprise Directory (LDAP) Server

3. To edit a server, select a server, and then click **Edit**. The Edit Enterprise Directory (LDAP) Server window is displayed. Provide the appropriate information, and then click **OK**.
4. To remove a server, select a server, and then click **Remove**.
5. When you complete the task, click **Close**. Use the online Help if you need additional information for setting up an LDAP server.

5.5 Questions and discussion

1. Cryptography discussions are often interesting. Are there any unbreakable encryption methods? If so, why are we so concerned about breaking ciphers?

In theory, one time pads, or the equivalent automated techniques, are unbreakable. They are impractical for HMC usage or for most common purposes. For practical purposes, a reasonable symmetric cipher (based on practical key lengths) is needed. In principle, such encryption schemes are breakable, but possibly only with very large efforts.

2. DES (56-bit) is still widely used. What is the practical degree of exposure in this?

In theory, 56-bit DES is breakable with today's fast computers, but it requires a concentrated, skilled effort. Many current LAN protocols involve temporary DES keys that are set through public key negotiations. Breaking a DES key is meaningful only for that one session. Extremely sensitive links, such as might be used by major banks or defense establishments, should worry about DES. More routine installations are likely to have many more potential security exposures to worry about before coming to potential DES exposures.

3. Why is public key cryptography not used for all communication?

Encryption and decryption with current public key schemes requires intense computation to handle only a few bytes of data. In practice, it is better to use public key cryptography to exchange only symmetric keys or something similar, such as identity certificates, and then use much faster symmetric methods for data exchange.

4. Remote access to the HMC functions appears to be very convenient. Why is that not every installation uses it?
5. Access to an HMC can completely disrupt all of the mainframes that are managed by that HMC. It is possible to provide secure remote access to the HMC if all the security controls and passwords are properly managed. Some installations might have doubts about these controls, for example, despite all rules to the contrary, some people still write down their passwords near their terminal. The HMC functions are so critical that some installations simply do not want the potential exposures created by remote users. These concerns might be a little paranoid, and the use of remote HMC operation is spreading.

HMC networking

In this chapter, we cover the following topics:

- ▶ Integrating the HMC into your network.
- ▶ Settings to consider when including the HMC into your Network:
 - Internet protocol Version 4 (IPv4) only:
 - DHCP
 - Static
 - IPv6 only
 - Both
 - Network settings on the HMC
- ▶ Firewall usage
- ▶ SSL certificates that grant access within corporate network
- ▶ Network accessibility with two Ethernet Adaptors

6.1 Overview of HMC networking

The *Customize Network Settings* task on the HMC provides an easy-to-use interface for the user to integrate the HMC into their network without needing to understand the underlying operating system mechanics. The HMC platform operating system provides low-level methods for manipulating HMC network configurations. These methods involve detailed shell commands and multiple configuration files in various locations. The Customize Network Settings task acts as a layer of abstraction to these complex mechanisms, providing ease of use and protecting the user from making common configuration mistakes. In fact, the HMC user does not have direct access to the HMC operating systems network configuration elements at all; instead, the only way for the user to manipulate the network configuration is through this task. Therefore, it is important that the task have sufficient functionality to cover all customer networking requirements, while still being easy-to-use.

In this chapter, we:

- ▶ Review how to integrate the HMC into a network
- ▶ Discuss the network protocols that are used
- ▶ Describe how to customize the HMC to a specific network
- ▶ Wrap up with an overview of the HMC firewall capabilities and configuration options

6.2 Integrating the HMC into your network

Figure 6-1 on page 115 shows a simplified diagram of a typical customer network in a System z environment. A typical customer's network usually includes a dedicated network on which only System z servers and HMCs are present. This network is separate from the customer's other networks and network services, such as printing, file transfers, and internet access. This dedicated network provides increased security because the server is not exposed to potentially dangerous traffic that often exists on a typical corporate network. HMCs are often configured with two network interfaces so that they can exist on both the dedicated network and a more general corporate network, which provides convenience for the customer because they can access the HMC from anywhere that their network exists (and it might be connected to the public Internet); however, this exposes the HMC to all of the potential security threats on these more public networks.

When an Internet connection is used, the HMC can be configured to use a second network card to physically separate a private LAN connection from an Internet-enabled network. Depending on the network configuration, the HMC might be able to connect to IBM service using the IPv4 or IPv6 protocols. This selection is made in the Customize Outbound Connectivity task.

While HMCs often have two network interfaces (to a dedicate network and to a public network), the Support Elements within a System z are normally connected only to the dedicated network.

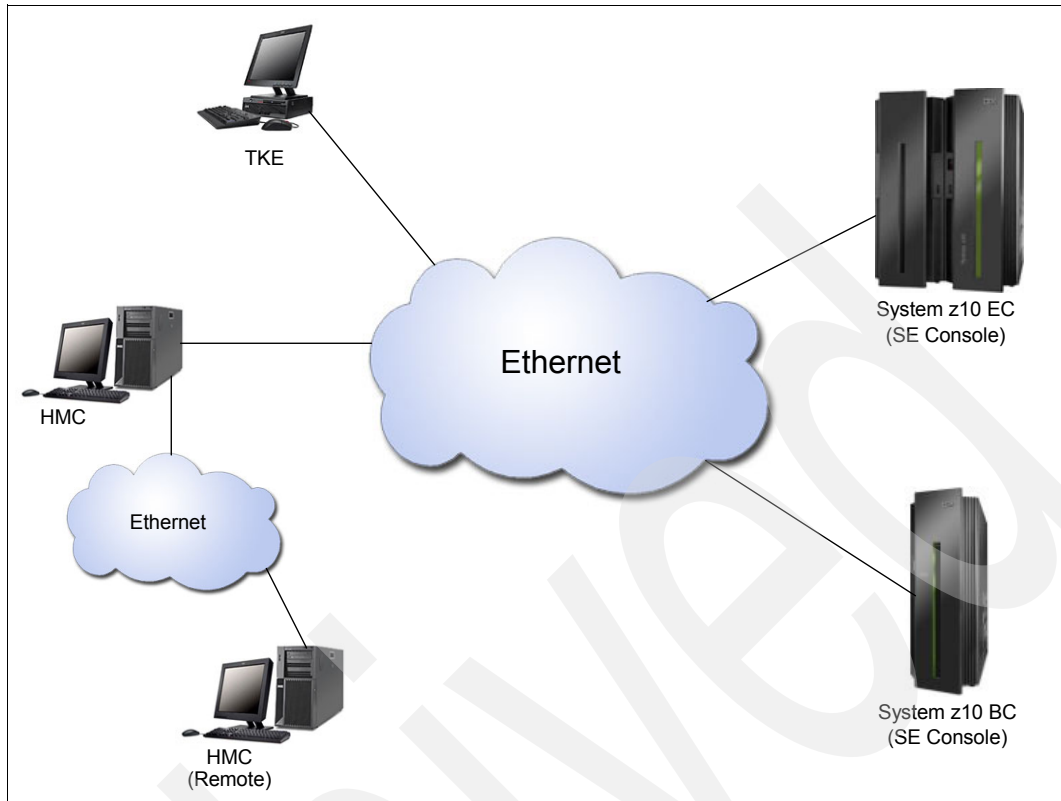


Figure 6-1 Network overview

When integrating the HMC into an enterprise network we must consider many aspects of the network and determine how the HMC can effectively interact with the network. Some of these considerations are whether our network uses IPv4 or Internet protocol Version 6 (IPv6) and whether it uses DHCP or Static Addressing. We also must consider specific DNS needs for resolving network names into IP addresses and specific routing needs for reaching other subnets.

6.2.1 Identification

On the Customize Network Settings panel, Figure 6-2 on page 116, the Identification tab allows configuration of the HMC's console name, domain name, and a description of the HMC's role:

- Console name** Provides the HMC name that identifies this console to other consoles in the network. This is the short host name, for example NEXTGEN.
- Domain name** The domain portion of the fully qualified DNS host name, for example, if the fully qualified host name is xyz.endicott.ibm.com, the host name is xyz, and the domain name is endicott.ibm.com.
- Console description** A short text field that is normally used to express the role of this particular HMC.

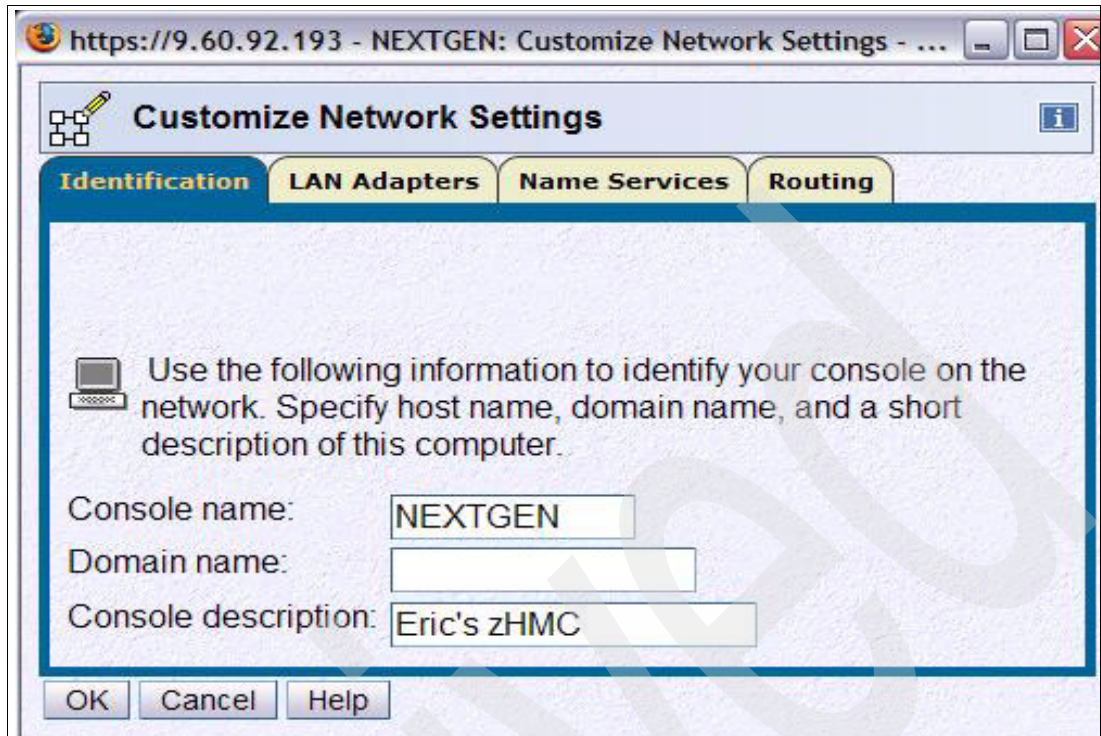


Figure 6-2 Network settings window / identification

6.3 Network protocols overview

In this section, we provide an overview of network protocols:

IPv4

IPv4 has 32 bits (bytes) for addressing, which allows 2^{32} (4,294,967,296) unique addresses. A sample of this type of addressing is 127.0.0.1, which is the loopback address that a host uses to send a message back to itself. Currently IPv4 is the Internet protocol that is generally used on the Internet.

IPv6

With IPv6 we can have 2^{128} (approximately 3.4×10^{38}) unique addresses giving us many more addresses to use. As more devices and machines become Internet capable, more and more IP addresses are needed. With the increasing demand for IP addresses, IPv6 will most likely be the successor to IPv4 as the standard for Internet connectivity.

Dynamic Host Configuration Protocol

Dynamic Host Configuration Protocol (DHCP) is a protocol for assigning addresses to devices on a network. It minimizes the number of addresses that the network needs to assign because each address is assigned dynamically for only the period of time that the device is in use. Also, DHCP cuts down on administrative tasks for the users because it allocates the necessary addresses dynamically, and the user does not need to configure the network connections manually.

Static addressing

Static addressing refers to a manually-entered (hard coded) address that applies to only one host on a network. This protocol can be necessary for many reasons, including that a host or network simply does not support DHCP. Static addressing is commonly used for a host that must be reachable at a specific address that does not change, such as a network printer. In such cases static addressing is a good solution.

6.3.1 LAN adapters

A summarized list of all (visible) Local Area Network (LAN) adapters that are installed in the HMC are in the Customize Network Settings panel on the LAN Adapters tab, as shown in Figure 6-3. You can select any of the LAN adapters, and click **Details** to open a window where you can work with the basic LAN settings or IPv6 settings.

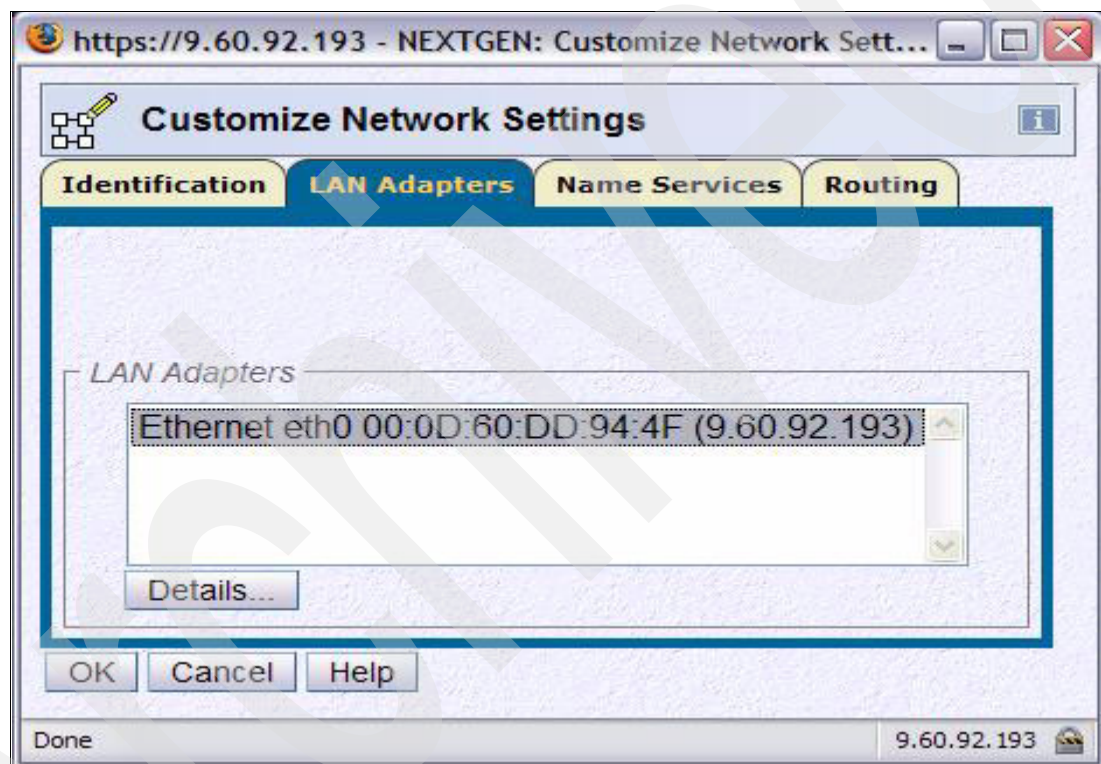


Figure 6-3 Customized Network Settings panel / LAN Adapters

Selecting Details produces the LAN Adapter Details panel, shown in Figure 6-4 on page 118. The user can configure a number of options on this panel.

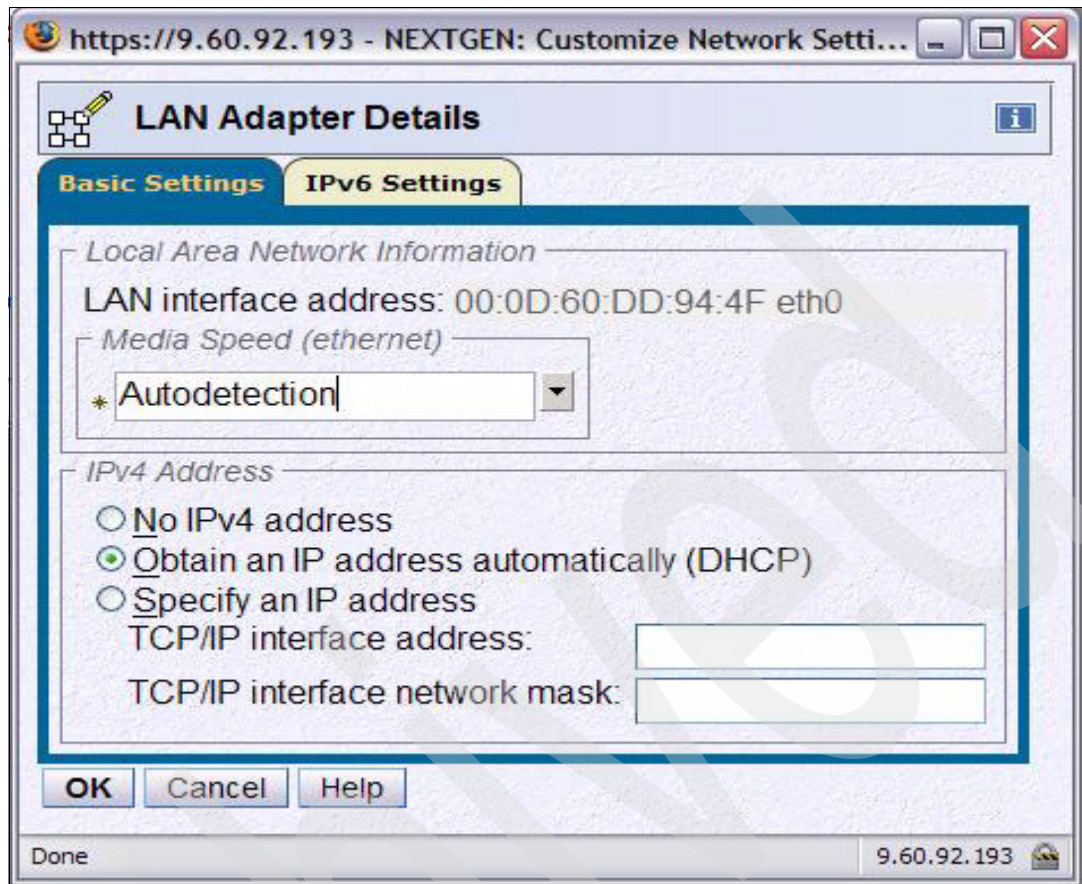


Figure 6-4 LAN adapter details settings

The tabs on Figure 6-4 are:

Basic Settings

Allows you to view and change current LAN adapter settings for the console, specifically for IPv4 addresses. Using this page you can also indicate that an IPv4 address is not available.

IPv6 Settings

Allows IPv6 configuration settings for the selected network adapter that is defined on this console. An IPv6-capable host automatically generates a link-local address that can be used to communicate with other hosts that are directly connected to the same physical line (a switch or hub). To communicate with hosts beyond the physical link, the host must acquire a suitable address to use. The most common method of acquiring an IPv6 address is by IPv6 autoconfiguration. A host with autoconfiguration enabled sends a request to the IPv6 router on the physical link. The router responds by sending the host a network prefix. The host then adds a unique suffix (based on the hardware MAC address) to create a complete IPv6 address, for example (Figure 6-5 on page 119), the router might respond with prefix 2001:00B2. The host then appends a unique suffix, such as 0202:B3FF:FE1E:8329, to make the complete IPv6 address 2001:00B2:0000:0000:0202:B3FF:FE1E:8329. IPv6 rules allow this to be expressed in a shorter form as 2001:B2::202:B3FF:FE1E:8329.

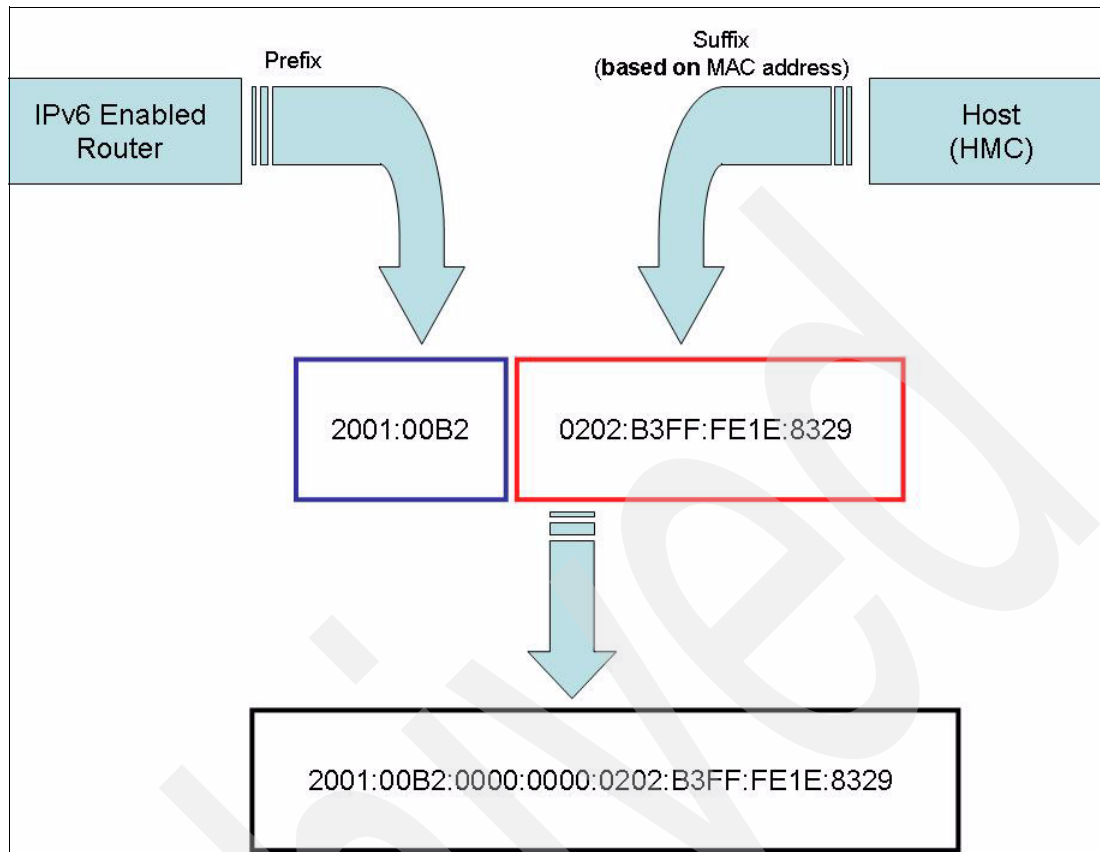


Figure 6-5 IPv6 Stateless Autoconfiguration Link-Local Address

There is an IPv6 version of DHCP that is known as DHCPv6. This protocol works in the same manner as DHCP for IPv4: A server reachable on the local link assigns a full IPv6 address to the client. One benefit of DHCPv6 over standard IPv6 autoconfiguration is that DNS information can be provided by the DHCPv6 server.

6.3.2 Customize network settings and Name Services tab overview

DNS is a distributed database system for managing host names and their associated Internet Protocol (IP) addresses. Using DNS means that people can use names, such as `www.jkltoys.com` to locate a host rather than using the IP address (`xxx.xxx.xxx.xxx`). A single server might only be responsible for knowing the host names and IP addresses for a small subset of a zone, but DNS servers can work together to map all Domain names to their IP addresses. Using DNS servers that work together means that computers can communicate across the Internet.

The Name Services tab

The HMC manages which DNS(s) to use and which order to use them when determining the IP address for a domain. Using the Name Services tab you can configure the DNS settings.

Figure 6-7 on page 121 shows the Customize Network Settings window with the Name Services tab selected.

The fields in the Name Services tab are:

- ▶ **DNS enabled:** To enable the DNS, select DNS enabled. You might want to enable DNS because you have DNS servers for mapping IP addresses to host names.
- ▶ **DNS Server Search Order:** Allows you to add IP addresses to the list to be searched for mapping the host names to IP addresses. You can also delete IP addresses from the search list. The top suffix is always searched first.
- ▶ **Domain Suffix Search Order:** Allows you to add a domain suffix to the list to be searched. You can also delete a domain suffix from the search list. The top suffix is always searched first.

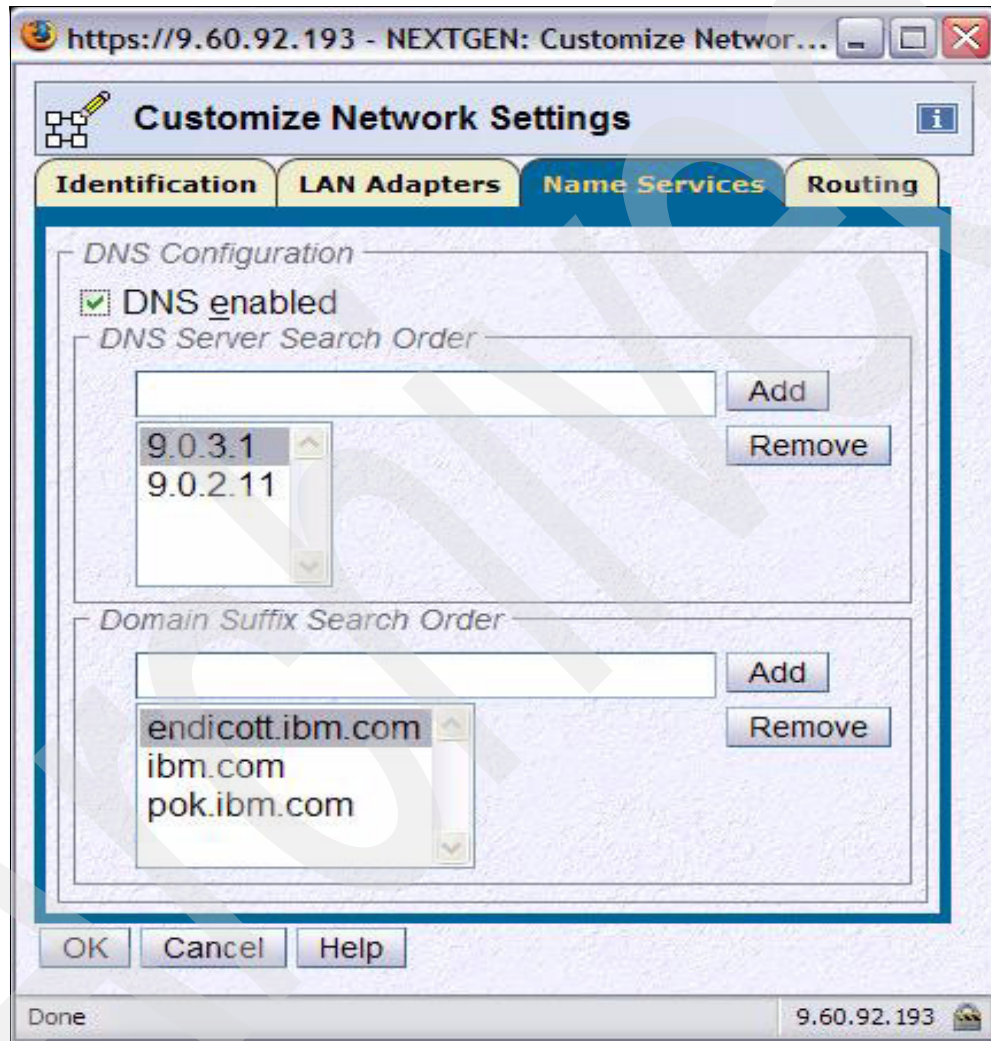


Figure 6-6 Name services entry

6.3.3 Routing

Figure 6-7 on page 121 shows routing information and default gateway information. The Gateway address is the route to all other networks. The default gateway address (if defined) informs this HMC where to send data if the target station does not reside on the same subnet as this HMC, which is necessary to allow the HMC to connect to the IBM Service Support System using the Internet. You can assign a specific LAN to be the Gateway device, or you can choose **any**. You can select **Enable routed** to start the routed daemon.

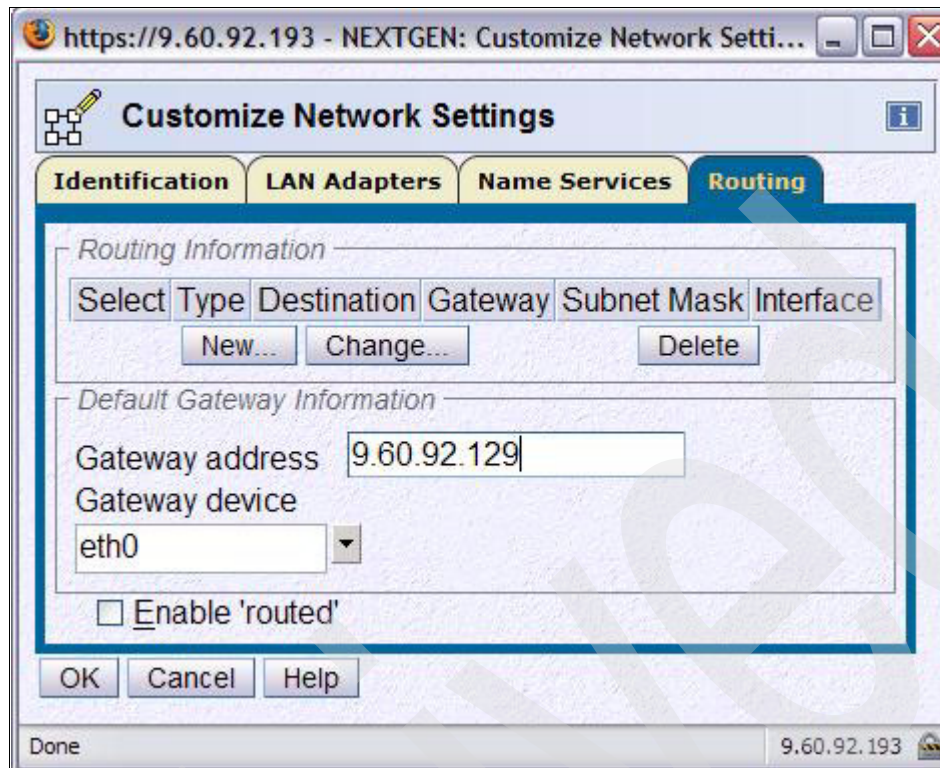


Figure 6-7 Customize Network settings, Routing tab

6.4 Firewall

The network is an integral part of the HMC and HMC management of the associated zSeries processors and related devices. The network is also one of the primary areas of concern related to the security and integrity of the Next Generation HMC. To help address this area of concern, the Next Generation HMC is configured with a firewall to limit network access in and out of the HMC. By default, no external connections are allowed through the firewall. There are no firewall user accessible functions. The HMC manages and controls the external network access transparently to the operator.

6.4.1 The HMC Firewall

The HMC Firewall is implemented using the standard Linux IP packet filter, which is manipulated using the **iptables** command. The IP filter table is organized into chains, which represent three basic types of packets: INPUT, FORWARD, and OUTPUT:

- ▶ The INPUT table is used for packets that the HMC receives from other hosts, destined for the HMC.
- ▶ The FORWARD table is used for packets that the HMC receives that are destined for other hosts (this should never happen, but the table exists so that it can be used by network routers).
- ▶ The OUTPUT chain is used for packets that originated from the HMC that are destined for outside hosts. Each of these chains has a default policy and can contain rules to further control the handling of packets.

The default policy can be to either ACCEPT or DROP the packet:

- ▶ If the packet is ACCEPTed, the packet is passed on to the next level of IP processing.
- ▶ If the packet is DROPed, the packet is ignored, and no further IP processing occurs.

The rules inside each chain can also DROP or ACCEPT packets if the packet matches the rule. An example rule would be to DROP any packet that comes from host 9.60.15.40.

When a packet is handed to the IP packet filter (either received from an outside host or generated by some HMC component), it chooses the appropriate chain for the packet (generally either INPUT or OUTPUT), and then matches the packet against each rule in that chain, in order. If the packet matches any of the rules, the action for that packet is executed; otherwise, the default policy for the chain is executed.

The HMC sets the default policy of the INPUT chain to DROP all packets, and the OUTPUT chain to ACCEPT all packets, which means that all incoming packets are ignored while all outgoing packets are allowed. Denying all incoming packets is done for security because we want to strictly control the packets that actually reach HMC components. Allowing all outgoing packets is done for convenience because we trust that no malicious code is running on the HMC, thus any outgoing packets are safe to send. We also set the policy of the FORWARD chain to DROP to prevent hosts from attempting to use the HMC as a router.

When an HMC must initiate a connection to an outside host, packets must flow in both directions to and from that host. The packets flowing from the HMC to the host are allowed because the policy of the OUTPUT chain is ACCEPT, but the packets flowing back from the host to the HMC are dropped by the DROP policy on the INPUT chain. To handle this, we add a rule to the top of the INPUT chain to accept any packet that is associated with an established connection, which uses the Linux IP filters connection tracking capability. When the HMC sends the first packet to the outside host, Linux remembers it, so that when the response packet comes in from the outside host, it is allowed. This connection tracking capability is a huge convenience because it allows outgoing connections to work without performing any dynamic changes to the firewall.

When an HMC component needs to accept a connection initiated by an outside host, the component must dynamically add a rule to the INPUT chain to allow the connection to be made. All HMC components manipulate the firewall using an API that allows the user to add rules that are based on the source address, protocol, and port of the incoming packet. Using this API, a user can, for example, add a rule to allow a connection to TCP port 6667 from a host with address 9.60.74.149 or to allow UDP packets to port 161 from any host on network 9.60.0.0/255.255.0.0. For security purposes, HMC components often allow access to only a single host that they expect to connect. Occasionally, HMC components allow access to a port for all hosts for a short period of time. Each HMC component is also responsible for removing any rules that they add after they are no longer needed.

Any incoming connection attempt that is not accepted by a rule in the INPUT chain is logged and dropped. The number of attempts that get logged is limited so that the log is not overrun with illegal access attempts.

6.5 Questions and discussions

1. One of the figures contains a token ring LAN. Where are these used? What are their advantages? Disadvantages?
2. A significant number of HMC installations ignore all of the LAN security guidelines that we discussed here, but are not considered insecure. How is this possible?

3. What is unique about IP addresses 192.168.xxx.xxx? IP addresses 10.xxx.xxx.xxx? Why are many of the examples using IP addresses 9.xxx.xxx.xxx?
4. IPv6 has been available for some years now, but is not used much (yet). Why?

Archived

Archived

Remote Support Facility

The Remote Support Facility (RSF), which is also commonly referred to as call home, is one of the key components that contribute to zero downtime on system z Hardware.

The Remote Support facility is designed to deliver information that is generated by HMC components to IBM support in a timely manner, to ensure that:

- ▶ Detected hardware and firmware problems are analyzed and IBM Service Representatives (SSR) are dispatched without customer intervention to replace parts
- ▶ Known issues in the customer service levels are preemptively addressed
- ▶ Firmware code updates are automatically obtained
- ▶ On demand configuration changes can occur without the need to dispatch an SSR, and associated billing is kept accurate
- ▶ Customer hardware inventory is accurately maintained to enable IBM Technical Sales to generate viable configuration updates
- ▶ Unexpected issues are analyzed by the IBM Engineering team for future product improvements

Considering system z design of redundant components (for example, a primary and backup power supply) when call home is configured, IBM service can effectively address issues with minimal customer involvement or impact.

The goal of the call home component is to securely and reliably transfer necessary data between the customers machine and IBM support. Redundancy of every component and path is designed into the support.

We also briefly discuss the Transmit Service Data (TSD) task in this chapter. TSD enables customers and SSRs to manually transmit hardware data to the IBM support center or alternatively to media for analysis. TSD is typically used when initial data collected by Problem Analysis was insufficient for IBM to diagnose a problem. Transmission to media is essential where the amount of data required to solve a problem is very large, there are network problems that preclude a successful call home, or where a third-party maintainer is contracted to service the customer's system.

After reading this chapter you should understand the purpose and general usage of the RSF/call home function, and how it fits into the overall system management design.

7.1 RSF/call home overview

The Hardware Management Console Remote Support Facility provides communication to an IBM support network, known as RETAIN®, for hardware problem reporting and service. When an HMC enables RSF, the HMC then becomes a call home server. The types of communication that are provided are:

- ▶ Problem reporting and repair data.
- ▶ Fix delivery to the service processor and Hardware Management Console.
- ▶ Hardware inventory data.
- ▶ System updates that are required to activate Capacity On Demand changes.

7.2 RSF security characteristics

Typically systems are configured to send hardware serviceability data using the Transmit System Availability Data (TSAD) operation. The data that is captured contains a snapshot of the hardware configuration, the performance data, and usage data. IBM service and engineering teams use this data to analyze actual system usage, reliability, and performance to effectively maintain high-system availability. TSAD is an HMC/SE scheduled operation and occurs automatically, by default, on a weekly basis.

The following call home security characteristics are in effect regardless of the connectivity method that is chosen:

- ▶ Remote Support Facility requests are always initiated from the Hardware Management Console to IBM. An inbound connection is never initiated from the IBM Service Support System.
- ▶ All data that is transferred between the Hardware Management Console and the IBM Service Support System is encrypted in a high-grade Secure Sockets Layer (SSL) encryption.
- ▶ When initializing the SSL-encrypted connection, the Hardware Management Console validates the trusted host by its digital signature issued for the IBM Service Support System.
- ▶ Data sent to the IBM Service Support System consists solely of hardware problems and configuration data. No application or customer data is transmitted to IBM.

7.3 Choosing a connectivity method for remote support

You can configure the Hardware Management Console to send hardware service-related information to IBM by using a dialup connection over a modem or using an Internet-based connection. The advantages to using an Internet connection are:

- ▶ Significantly faster transmission speed.
- ▶ Ability to send more data on an initial problem request potentially resulting in more rapid problem resolution.

- Reduced customer expense (that is, the cost of a dedicated analog telephone line).
- Greater reliability.

Unless your enterprise's security policy prohibits any connectivity from the Hardware Management Console over the Internet, an Internet connection is recommended.

Note: You can choose to configure a Hardware Management Console for both Internet and modem connectivity. In this case, the Hardware Management Console tries to establish an Internet connection first and treats modem connectivity as a backup mechanism if the Internet connection is not successful. Prior to the introduction of the z9 HMC, only a dial-up connections was supported.

7.3.1 Modem connectivity

Figure 7-1 displays a typical dial-up environment. This configuration allows the Hardware Management Console to use a modem through which connections are made to the IBM Service Support System. The Hardware Management Console automatically detects the modem when it initializes.

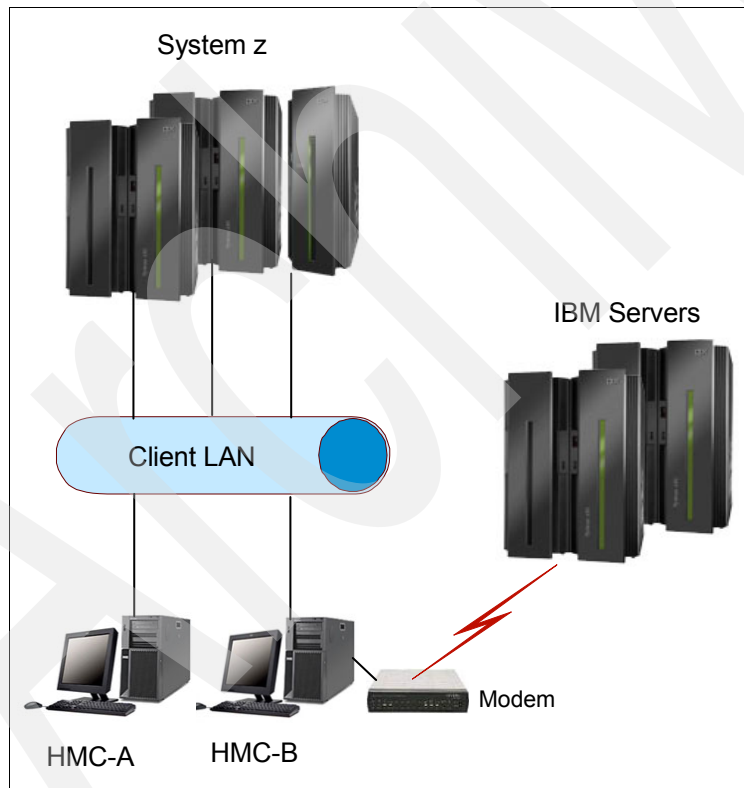


Figure 7-1 Modem connectivity

If a modem is used, the HMC requires a dedicated analog line to connect to the modem. When a serious system error occurs, call home requires the dedicated line so that an IBM serviceability action can be initiated immediately. The Hardware Management Console uses one of the configured telephone numbers to dial the modem and connects to the Global Network. The Hardware Management Console creates a TCP/IP connection through a fenced

Internet, which completes the network between the Hardware Management Console and the IBM servers.

The fenced Internet connection uses a firewall to limit access between the Hardware Management Console and the Internet. Specifically, it allows communication only between the Hardware Management Console and a list of IBM IP addresses. All other access to and from the Internet is blocked.

7.4 Internet connectivity

The Hardware Management Console can be configured to use an existing Internet connection to connect to the IBM Service Support System. All of the communications are handled through TCP sockets that the HMC initiates, and all data is transmitted using a minimum of 128-bit SSL encryption. The destination TCP/IP addresses are published to enable you to set up firewalls allowing connection.

The HMC can connect to IBM service using IPv4 (basic) or IPv6 Internet protocols, which we discussed in 6.3, “Network protocols overview” on page 116. Depending upon your network configuration, you can configure the IPv4 and IPv6 settings from the Customize Outbound Connectivity task, as shown in Figure 7-8 on page 135. The standard HTTPS port 443 is used for all communications.

When the HMC and SE hardware are set up, there are typically two network interface cards (NIC) on the HMC. Using a second network card adds additional security because two physically separate cards have separate memory buffers that prevent any communication between the HMC and the SE from being viewed or tampered with by the outside network. This allows customers to use a private network between the HMC and SE, but allows the HMC to connect to the corporate intranet or Internet.

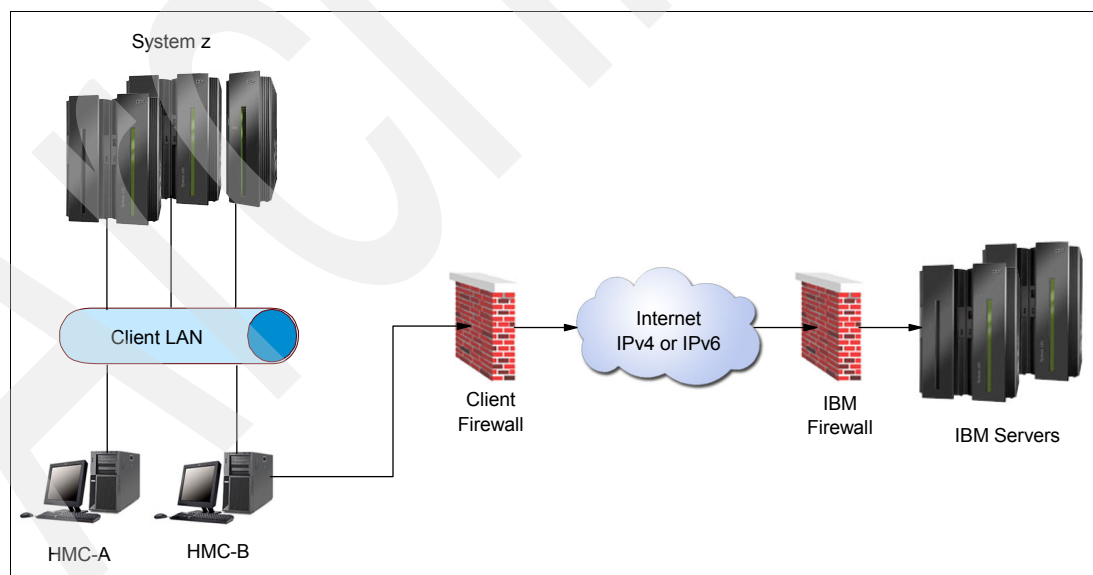


Figure 7-2 SSL connectivity

The Hardware Management Console can be enabled to connect directly to the Internet, as shown in Figure 7-2, or to connect indirectly from a customer-provided proxy server, as shown in Figure 7-3 on page 129. The decision about which of these approaches works best for your installation depends on the security and networking requirements of an enterprise.

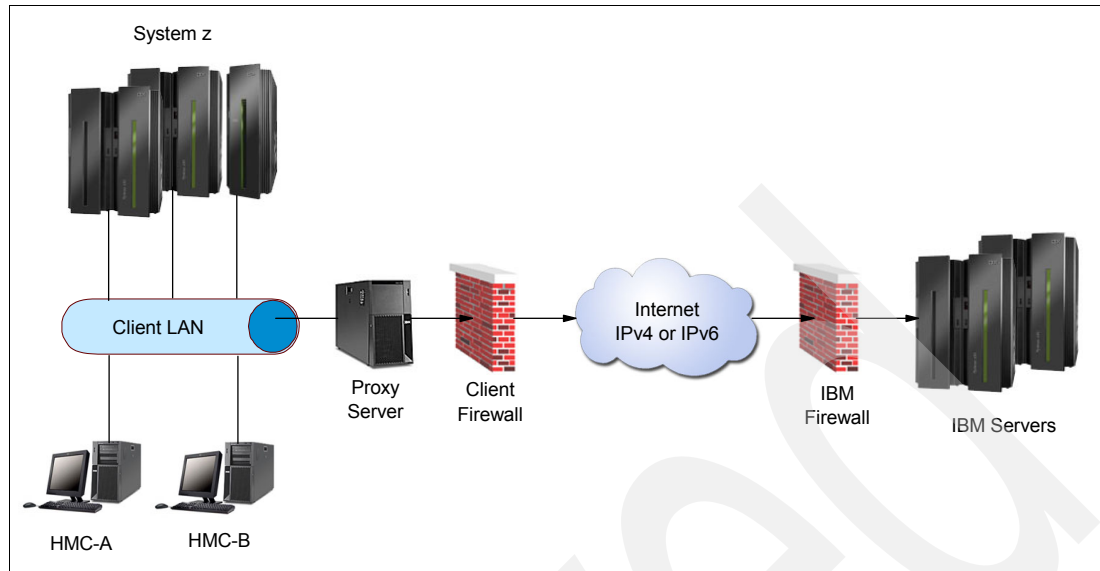


Figure 7-3 SSL with proxy server connectivity

7.4.1 Hardware management console direct Internet SSL connection

If your Hardware Management Console can be connected to the Internet, and the external firewall can be set up to allow established TCP packets to flow outbound to the destinations that are specified in an IBM server address list, you can use a direct Internet connection, as shown in Figure 7-2 on page 128, for a sample configuration. Using Source Network Address Translation (SNAT) and masquerading rules to mask the Hardware Management Console's source IP address are both acceptable.

7.4.2 Using an indirect Internet connection with proxy server

If your installation requires the Hardware Management Console to be on a private network, you might be able to use an Internet connection indirectly using an SSL proxy, which can forward requests to the Internet, as shown in Figure 7-3, for a sample configuration. One of the other potential advantages of using an SSL proxy is that the proxy can support logging and audit facilities.

To forward SSL sockets, the proxy server must support the basic proxy header functions (as described in RFC 2616) and the CONNECT method. Optionally, basic proxy authentication (RFC 2617) can be configured so that the Hardware Management Console authenticates before attempting to forward sockets through the proxy server.

For the Hardware Management Console to communicate successfully, the client's proxy server must allow connections to port 443. You can configure your proxy server to limit the specific IP addresses to which the Hardware Management Console can connect. We provided a list of IP addresses for the IBM servers in "IBM server address list" on page 130.

Depending on your configuration, the Hardware Management Console might be able to connect to your SSL proxy using a different Internet protocol than what is required for the proxy to connect to the Internet. In this case, the value that is selected in Protocol to Internet on the Internet page from the Customize Outbound Connectivity task, shown in Figure 7-8 on page 135, must reflect the Internet protocol that the SSL proxy uses to connect to the Internet, for example, the Hardware Management Console can connect to your SSL proxy using an IPv6 address, but the proxy might be configured to connect to the Internet using

IPv4. In this case, you enter the IPv6 address of your SSL proxy server on the Internet page, but select IPv4 for Protocol to Internet from the Customize Outbound Connectivity task.

IBM server address list

The Hardware Management Console uses the following IP addresses (depending on your Hardware Management Console version) when it is configured to use Internet connectivity. All connections to these IP addresses use port 443 TCP. The IP addresses you must allow depends on the protocol that you are using:

- ▶ For Hardware Management Consoles at Version 2.9.2 or earlier, you must allow access to the following IPv4 addresses:
 - 129.42.160.48 and 207.25.252.200: Allow the Hardware Management Console access to the System Authentication Server (SAS).
 - 129.42.160.49 and 207.25.252.204: Allow the Hardware Management Console access to IBM Service for North or South America.
 - 129.42.160.50 and 207.25.252.205: Allow the Hardware Management Console access to IBM Service for all other regions.

Note: You only need to specify the IP address that is necessary to set up access to SAS and those that are appropriate for your region.

- ▶ For Hardware Management Consoles at Version 2.10.0 or later, Internet connectivity to the IBM Service Support System was enhanced to enable access from the IPv6 Internet and the IPv4 Internet. If you require access to the IBM Service Support System using the IPv6 Internet, you must allow access to all of these additional addresses:
 - 2620:0:6C0:1::1000
 - 2620:0:6C1:1::1000
 - 2620:0:6C2:1::1000
- ▶ For Hardware Management Consoles at Version 2.10.1, Internet protocol addresses using IPv4 requires outbound connectivity to the following IP addresses:
 - 129.42.26.224
 - 129.42.34.224
 - 129.42.42.224

If your Hardware Management Console connects to the Internet using IPv6, use the definitions that we described in this section.

7.4.3 Defining HMCs as call home servers

A Support Element can RSF service events using any Hardware Management Console that is configured for outbound connectivity, and the CPC object is defined to have the HMC act as a phone server, as shown in Figure 7-11 on page 138. To change the CPC definition after it is already defined, use Change Object Definition under the Object Definition task. If the CPC was not yet defined, you can define the CPC by going to Undefined CPCs under Groups and using the Add Object Definition under the Object Definition task.

A Hardware Management Console can RSF service events using any Hardware Management Console that was automatically discovered or was manually configured using the Customize Outbound Connectivity task, shown in Figure 7-8 on page 135.

A Hardware Management Console is discovered under the following circumstances:

- ▶ It is at the same level or higher as the source Hardware Management Console.

- ▶ It is enabled to call home, and the Outbound Connectivity Settings window is configured.
- ▶ It is configured to communicate on the same TCP/IP subnet that the source Hardware Management Console is configured to communicate.

To avoid a single point-of-connection to IBM failure and avoid bottlenecks, we recommend that you configure each Support Element and Hardware Management Console to use multiple call home servers. The first available call home server attempts to handle each service event. If the connection or transmission fails with this call home server, the service request is retried using the other available call home servers until one is successful or all are tried. When using modem support, it can take hours to complete any given activity, so it is critical that another HMC is available to transmit time-sensitive requests, such as opening problem records or Capacity on Demand activities.

7.5 RSF setup

RSF/call home setup requires the proper configuration of both a reporting (initiating) system and a sending (transmitting) system.

The sending system is always a Hardware Management Console (HMC). Generally the reporting system is a CPC, but the HMC can also act as a reporting system, for example, when errors are detected in the HMC itself. It is strongly recommended that the CPC, and the HMCs that act as its call home server, reside at the same customer site to reduce transmission errors. While call home setup is typically done by IBM Support as part of the installation process, it can also be performed by a customer.

In this example, we set up connectivity first on the HMC, and enable it to also report its own problems. The HMC's network settings and other (firewall, proxy) configuration and phone lines are put in place prior to setting up call home.

7.5.1 Enabling RSF on the HMC

There are three steps to enable RSF on an HMC. These steps are to configure customer information, configure outbound activity, and to customize remote service. We describe these steps here.

1. Configure customer information:
 - a. First ensure that the Customer Information is correctly set.
 - b. From the HMC Settings, in Console Actions, select **Customize Customer Information**, as shown in Figure 7-4 on page 132. This data is required for call home to be successful.

Note: Customer information consists of three components:

- ▶ Administrator
- ▶ System
- ▶ Account information

The first tab is contact information for the system administrator. This data is used by IBM support to contact the customer. The country information is used to route calls to the appropriate processing location.

Customize Customer Information

Administrator System Account

Contact Information

Company name: * The Barber Shop

Administrator name: * Sweeney Todd

Email address: stodd@customer.shop

Phone number: * 800-123-4555

Alternate phone number:

Fax number:

Alternate fax number:

Mailing Address

Street address: * 1701 North Street

Street address 2:

City or locality: * Endicott

Country or region: * United States (of America)

State or province: * New York

Postal code: * 3760

OK Cancel Help

Figure 7-4 Customize Customer Information panel - 1

It is common for the system and administrator to be the same, but the system can have a separate address, as shown in Figure 7-5 on page 133.

Customize Customer Information

Administrator System Account

System Location

☒ Use the administrator mailing address

Street address: * 1701 North Street

Street address 2:

City or locality: * Endicott

Country or region: * United States (of America)

State or province: * New York

Postal code: * 3760

OK Cancel Help

Figure 7-5 Customize Customer Information panel - 2

The account information, Figure 7-6, is typically not sent to IBM, but available for reference.

Customize Customer Information

Administrator System Account

Account Information

Customer number: 111222

Enterprise number: 222

Sales branch office: 754

Service branch office: 754

Area: 000

OK Cancel Help

Figure 7-6 Customize Customer Information panel - 3

2. Configure Outbound Connectivity:

- a. From the HMC Settings, in Console Actions, select **Customize Outbound Connectivity**, shown in Figure 7-7 on page 134.
- b. In the Local Console Configuration portion of the panel, click **Configure** to set up this HMC's connectivity to IBM support.

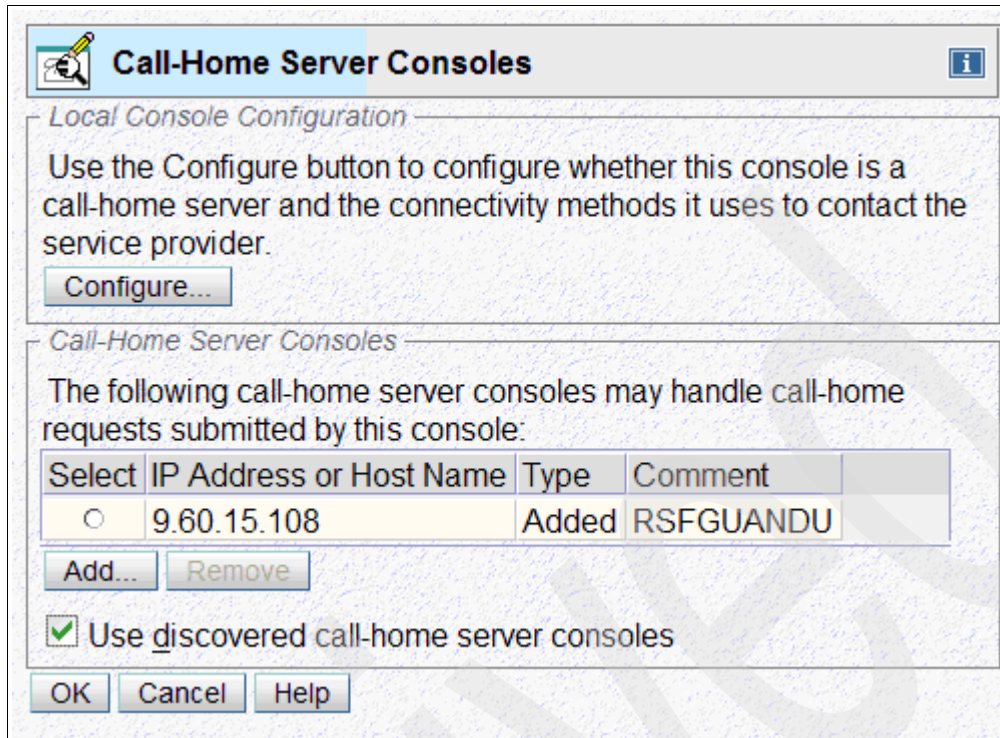


Figure 7-7 Call Home Server Consoles panel

After you select **Configure**, you will see three tabs:

- Local modem
- Internet
- External Time Source

Note: Setting up an External Time Source is not related to call home capability; instead, it is used to synchronize the system's clock with an external provider, which we do not discuss here.

You can configure an HMC to use the Internet or the local modem to call home. If both are chosen, the HMC attempts to use the Internet, and if the Internet connection fails, it tries to use a modem connection to call home.

In this example, we set up the HMC with an indirect Internet connection using an SSL Proxy. Select **Enable the local console as a call-home server**, and then enable the Internet connection by selecting **Allow an existing Internet connection for service**, shown in Figure 7-8 on page 135. This enables the “Use SSL proxy” check box. This configuration assumes that you have an external SSL proxy set up and configured to accept the HTTPS **connect** method, know the address or host name of the SSL proxy, and the SSL proxy port. The choice of a proxy is up to the customer.

In this example, the SSL proxy requires user ID/password format authentication. The identification that is used here is used for all connections from this HMC to IBM.

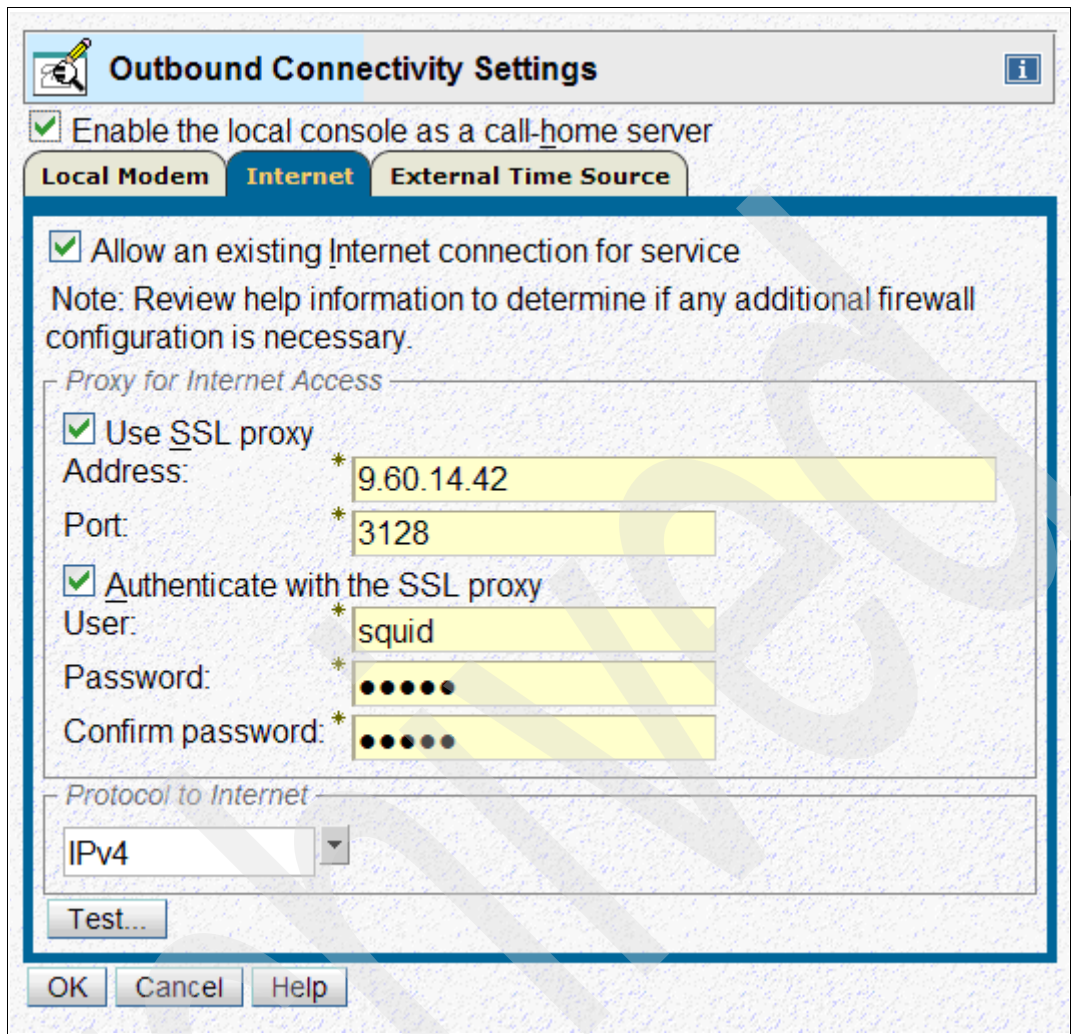


Figure 7-8 Outbound Connectivity Settings panel

Finally, it is highly recommended that you test the connection you configure before saving and exiting the task. At the bottom of the Customize Outbound Connectivity panel, Select **Test**, shown in Figure 7-8, and then at the Test Internet panel, select **Start**.

Figure 7-9 on page 136 illustrates an example of a successful test to IBM support using the previous configuration. Some of the more common reasons for Internet failures are:

- External (to HMC) firewall issues,
- SSL proxy filtering issues
- HMC networking errors
- Configuration errors to SSL proxy

If a modem connection is selected, you can test individual phone numbers.

Be sure to save your configuration after a successful test.

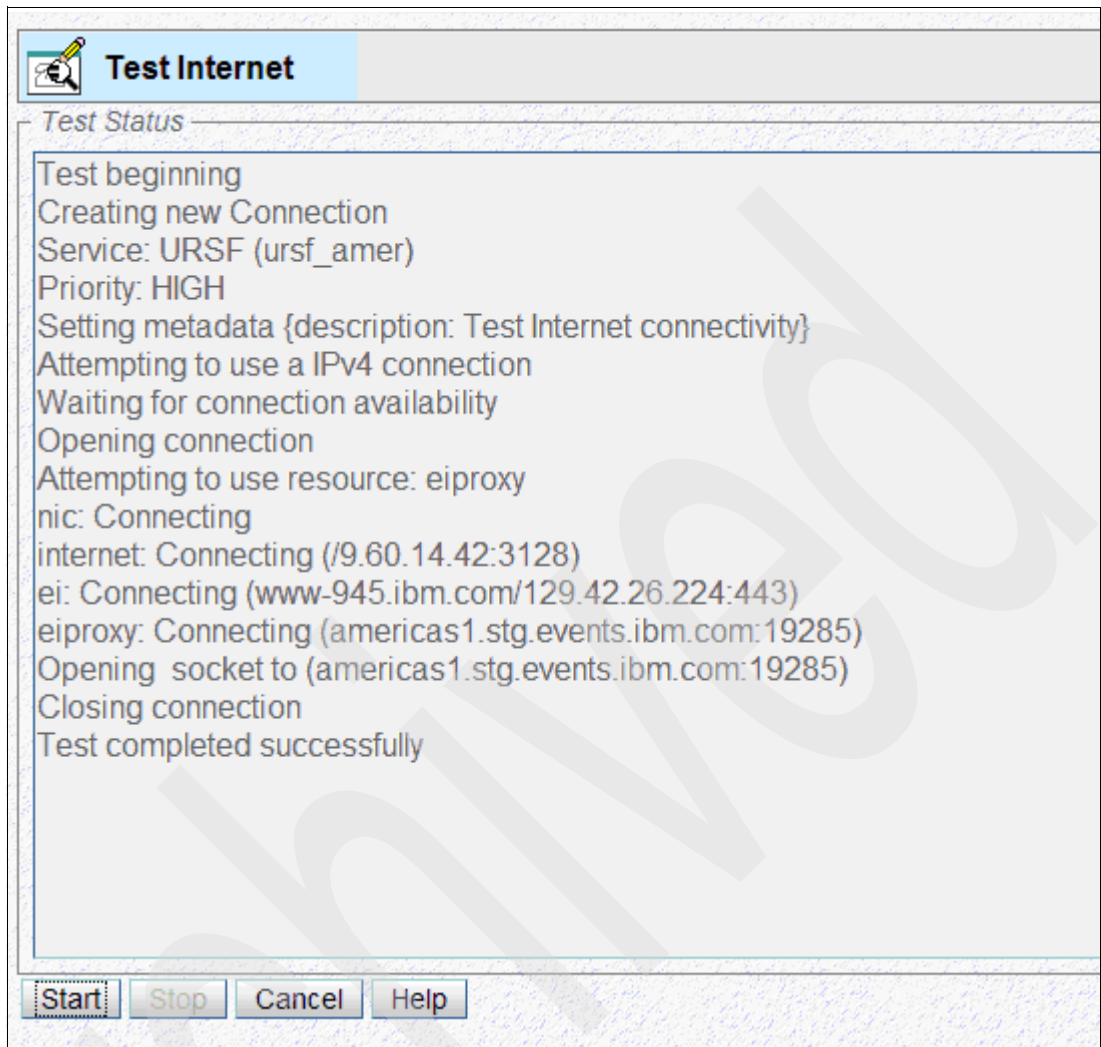


Figure 7-9 Test Internet panel

3. Customize Remote Service

Figure 7-10 on page 137 determines if and how remote support requests are initiated from this machine, which controls the usage of call home when HMC is the reporting machine. This setting is independent from the use of this system as a call home server.

To customize Remote Service:

- a. Go back to the HMC Settings of the Console Actions pane, and select **Customize Remote Service**.

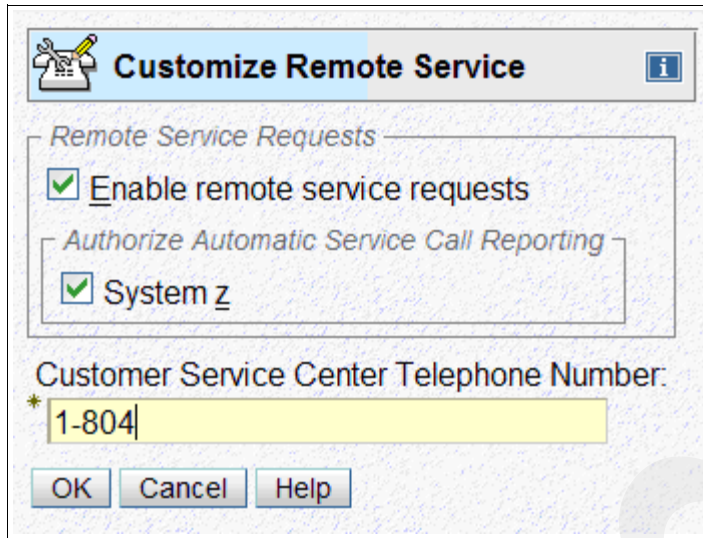


Figure 7-10 Customize Remote Service panel

- b. To enable this HMC to call home problems or make other remote requests, select **Enable remote service requests**. Selecting **Authorize Automatic Service Call Reporting** means that a problem detected by this HMC opens a problem with the support center without requiring manual intervention. Generally, this option should be selected when IBM warranty or maintenance applies. If this is NOT selected, hardware problems are viewable as hardware messages and can be transmitted manually to IBM support; however, this can result in a delay of critical problems.

HMC call home setup is complete

At this point you completed setting up call home for the HMC.

There are additional facilities that use call home setup that you can configure, which include: Scheduled Operations to retrieve fixes, transmit service data to IBM, and various console services. Refer to Chapter 11, “Change management” on page 223 and Chapter 10, “Problem analysis” on page 207.

7.6 Enabling a CPC to call home using this HMC

Connections to IBM support for a CPC are done indirectly using an HMC. An HMC can be configured as a call home server for a system if it:

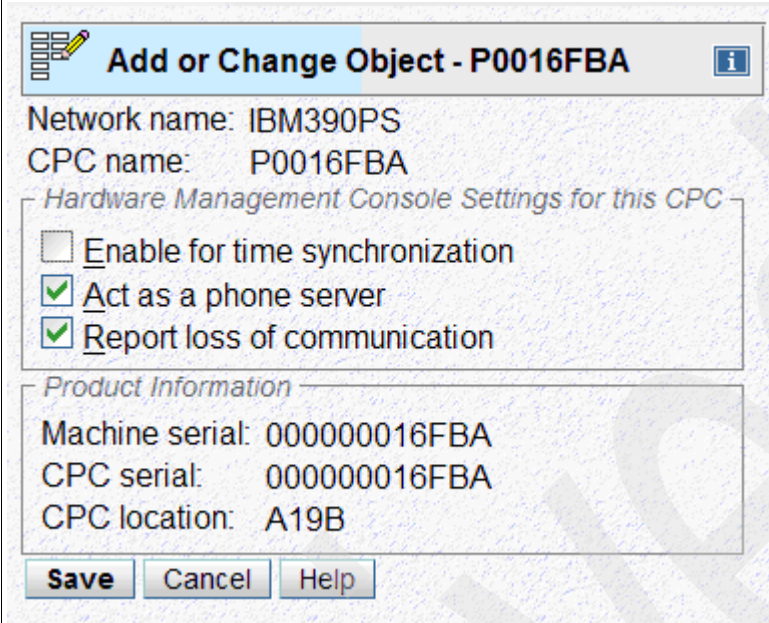
- ▶ Is at the same service level or higher than the CPC
- ▶ Can be reached in the network
- ▶ Is in the same Security Domain (if Domain Security is configured)

It is not necessary for the HMC and the CPC to be on the same subnet. However, IBM strongly recommends that the CPC, and the HMC that acts as a call home server, be at the same physical site.

An explicit action is required for a system to be enabled to call home using a particular HMC, which you do using the Object Definition panel:

1. Select a CPC, and then from the Object Definition Task Panel, select **Change Object Definition**.

2. In the Add or Change Object Panel, Figure 7-11, select **Act as a phone server**, and click **Save**.



Add or Change Object - P0016FBA

Network name: IBM390PS
CPC name: P0016FBA

Hardware Management Console Settings for this CPC

☐ Enable for time synchronization
☒ Act as a phone server
☒ Report loss of communication

Product Information

Machine serial: 000000016FBA
CPC serial: 000000016FBA
CPC location: A19B

Save **Cancel** **Help**

Figure 7-11 Add or Change Object panel

3. Ensure that the customer information is set up on the CPC, which is under Customer Information in the Remote Customization Task Panel, shown in Figure 7-12 on page 139. This information does not have to match that of the HMC, but unpredictable results can occur if the HMC and CPC are in different countries.

Customize Customer Information - P0LXSM27

Administrator System Account

Contact Information

Company name: * Demon Barber of Fleet Street

Administrator name: * Sweeney Todd

Email address: stodd@barbershop.com

Telephone number: * (607) 429-5793

Alternate telephone number: tie 620-5793

Fax number:

Alternate fax number:

Mailing Address

Street address: * 1701 North Street

Street address 2:

City or locality: * Endicott

Country or region: * United States (of America)

State or province: * New York

Postal code: * 13760

Reset

OK Cancel Help

Figure 7-12 Customize Customer Information panel (partial)

4. To enable the CPC to automatically transmit problems to IBM, in the Remote Customization Task Panel for the CPC, select Remote Service, as shown in Figure 7-13 on page 140.

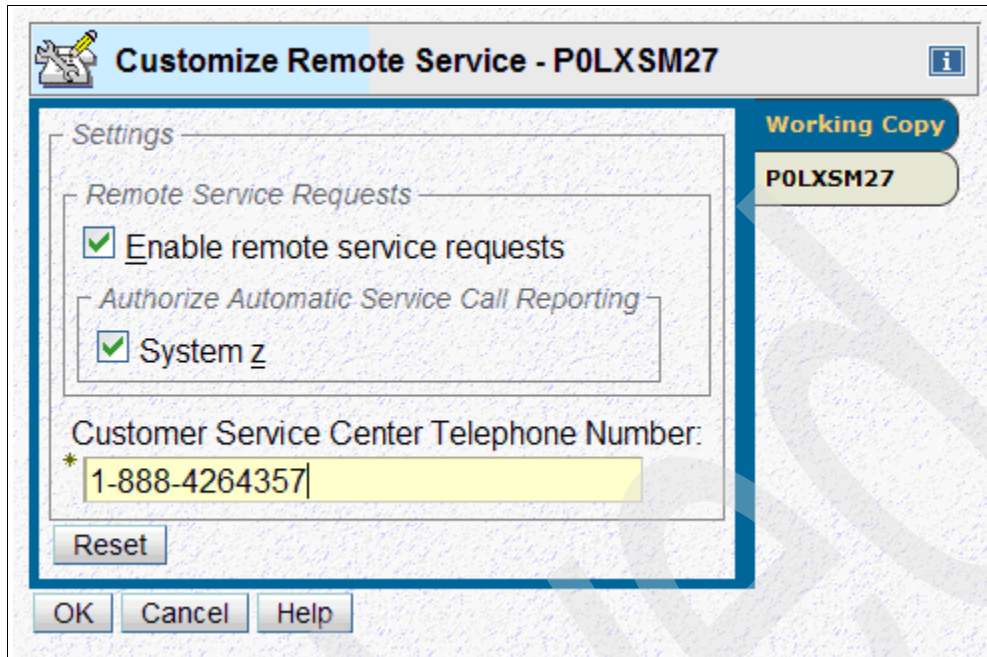


Figure 7-13 Customize Remote Service panel

Call home setup is complete.

We recommend that you test this process by generating a test problem from the CPC using the Report a problem task from the Service Task Panel.

7.7 Questions

1. Does the HMC call home only when there is a problem?
2. Some installations cannot permit any outside system connections. How is this situation handled?
3. Customer data (from applications) is not sent through the RSF facility, but it might be possible to derive competitive information from the statistical data that is sent through RSF. How is this potential exposure handled?
4. In a normal mainframe environment, might the HMC call home without the customer being aware of the action?

HMC-based partitioning

In this chapter, we describe how the Hardware Management Console (HMC) gives System Administrators the ability to manage the configuration and operation of partitions on a zSeries server. The goal of this chapter is to:

- ▶ Define Logical Partitions
- ▶ Describe System z partitioning
- ▶ Describe System z partition configuration options
- ▶ Discuss using the HMC to plan for, create, and configure partitions on the System z

At the end of this chapter you should know how to create and activate a System z partition using an Activation Profile from the HMC.

8.1 Introduction to logical partitions

Virtualization has emerged relatively recently in the mainstream as an important technology, although it has been available on IBM mainframes for more than 40 years. Through virtualization, servers, storage, and other computer resources can be made more flexible, adaptable, and useful. These improvements lead directly to cost savings and other efficiencies that reduce the Total Cost of Ownership (TCO) of an IT solution, for example, by using virtualization to consolidate multiple independent servers on fewer physical systems, floor space, power, and cooling requirements can decrease, sometimes dramatically, which leads to considerable cost savings.

IBM System z provides two levels of virtualization. Logical partitioning supports a relatively small number (up to 60) of high-performance virtual servers. Software partitioning through the z/VM hypervisor supports a much larger number of virtual servers with highly granular resource sharing. Both forms of virtualization are supported by the Processor Resource/Systems Manager (PR/SM) hardware to provide a highly-scalable environment that can accommodate a broad range of needs:

- ▶ On System z, a virtual machine, such as a logical partition, is a set of virtualized resources – including processors, memory, input/output, and networks – that can be used to run an operating system. The z/VM hypervisor is implemented as software and virtualizes resources to a much greater extent than the LPAR hypervisor.

- On System z, a logical partition (LPAR) is a set of virtualized resources – including processors, memory, input/output, and networks – that can be used to run an operating system. The LPAR hypervisor is implemented as part of the machine firmware. On modern System z processors, an LPAR is the only environment that is supported for operating system deployment.

On System z, LPAR Partitioning is built into the basic design, not added onto an existing element. The foundation of System z virtualization is a robust, reliable hardware platform. The System z hardware platform provides legendary reliability, availability, scalability, and security. System z LPAR Partitioning is hardware and firmware that maps virtual resources to real hardware resources. It is a set of techniques that allow computer system resources, such as memory, processors, and I/O devices, to be manifested without requiring a direct correspondence with an underlying real resource that has the same characteristics, for example, real memory can be used to present a resource that appears to be a hard drive. The virtual hard drive exhibits the behavior of its real counterpart in many ways, except that it probably has a much faster access time and is much more reliable, though unlike a real hard drive, the data might not be recorded permanently. The interfaces to the virtual resources and their behaviors conform to an architectural definition (zSeries Architecture Principle of Operations), regardless of whether they are mapped to part of a real resource, to several resources, or are presented by emulation in the virtualization layer.

Logical partitioning is an inherent part of every System z server, so processor and I/O device sharing are built-in and so is high-speed inter-partition communication. The IBM zSeries servers can be *partitioned* into separate logical computing systems. System resources (memory, processors, I/O devices) can be divided or shared among many independent logical partitions under the control of the LPAR hypervisor, which comes with the standard Processor Resource/Systems Manager (PR/SM) feature on all zSeries servers. Each LPAR supports an independent operating system that is loaded by a separate initial program load (IPL) operation.

Logical partitions contain one or more logical central processing units (CPUs), which can be defined to be dedicated or shared. The hypervisor creates a state description for each logical processor that contains related state information. It issues the Start Interpretive Execution (SIE) instruction to direct the machine to dispatch the logical processor on the physical processor and execute instructions. The LPAR hypervisor dispatches the logical CPU exclusively on a chosen physical CPU. Shared logical CPUs are dispatched on any remaining physical CPUs that are chosen not for dedication but in accordance with a user-assigned priority expressed in terms of a relative weight. The LPAR hypervisor maximizes the use of the available physical CPUs depending on the activity and the weight of the logical CPUs. You will learn more about dispatching later in this chapter.

The System z virtualization dimension adds flexibility and power to a computing environment. The hardware and virtualization dimensions of System z provide a solid foundation for an open, stable application dimension. Based on open and stable operating system interfaces with awareness of the virtual server environment on which they operate, enterprise applications can be deployed and operated efficiently and cost effectively.

Logical Partitioning is created by a combination of two separate elements:

- An IOCDs, which is created by IOCP, that defines the I/O configuration that each partition sees.
- Profiles, created using HMC functions, that define how the System z processors and memory are divided among the partitions.

The names of the partitions comprise the link between these two partitioning elements.

The System z operator can reset the total system, load a new IOCDS and profile, and have a completely different partition arrangement, which is known as a power-on reset (POR) that completely disrupts everything that is running in the System z, of course, and most System z installations very seldom perform a power-on reset.

8.2 How logical partitions are used

Logical partitions are used for hardware consolidation and workload balancing. Partitions are managed by the Processor Resource/Systems Manager (PR/SM), which is built into a System z machine. The System Operator defines the resources that are to be allocated to each logical partition. Most resources can be reconfigured non-disruptively (or without requiring a power-on reset) to other partitions. After the logical partition is defined and activated, an operating system can be loaded into that logical partition.

Figure 8-1 shows the multi-dimensional overview of System z virtualization technology. System z provides logical (LPAR) and software (z/VM) partitioning. PRSM enables highly scalable virtual server hosting for LPAR and z/VM virtual machine environments. IRD coordinates allocation of CPU and I/O resources among z/OS and non-z/OS LPARs.

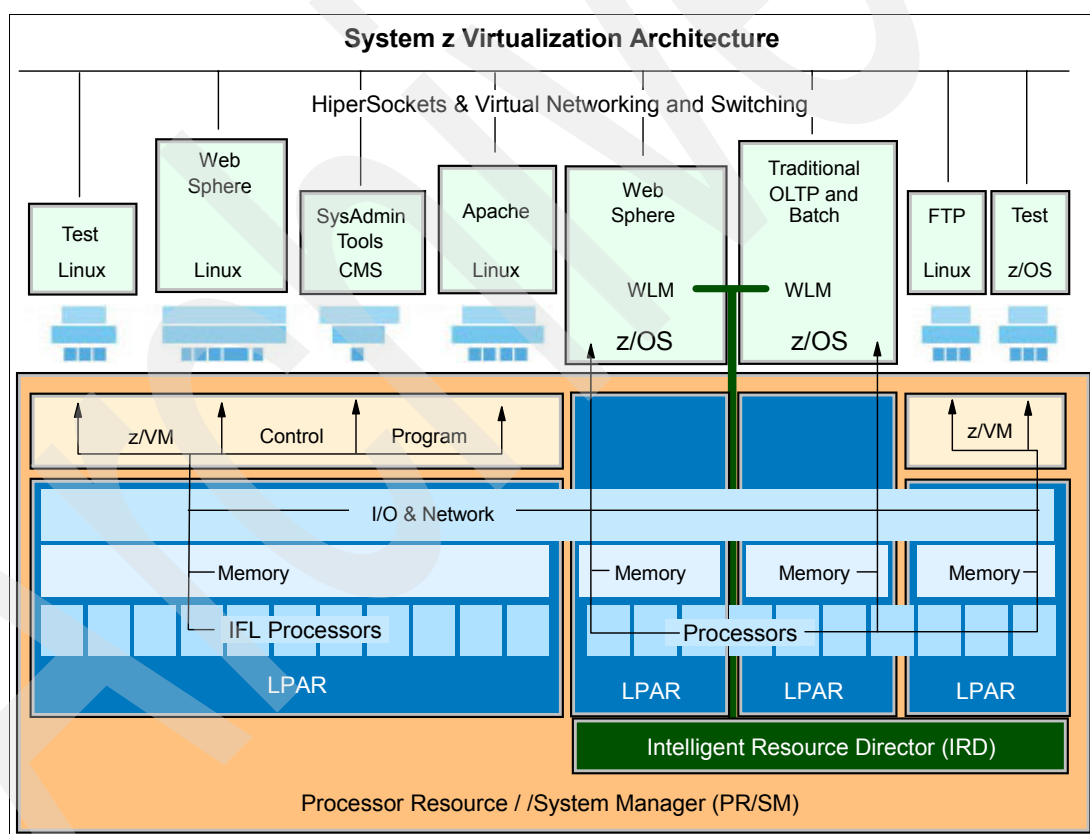


Figure 8-1 System z virtualization architecture

Figure 8-2 on page 144 shows some of the characteristics that can be defined for a logical partition. You can view each logical partition as a Central Processor Complex operating within the physical CPC.

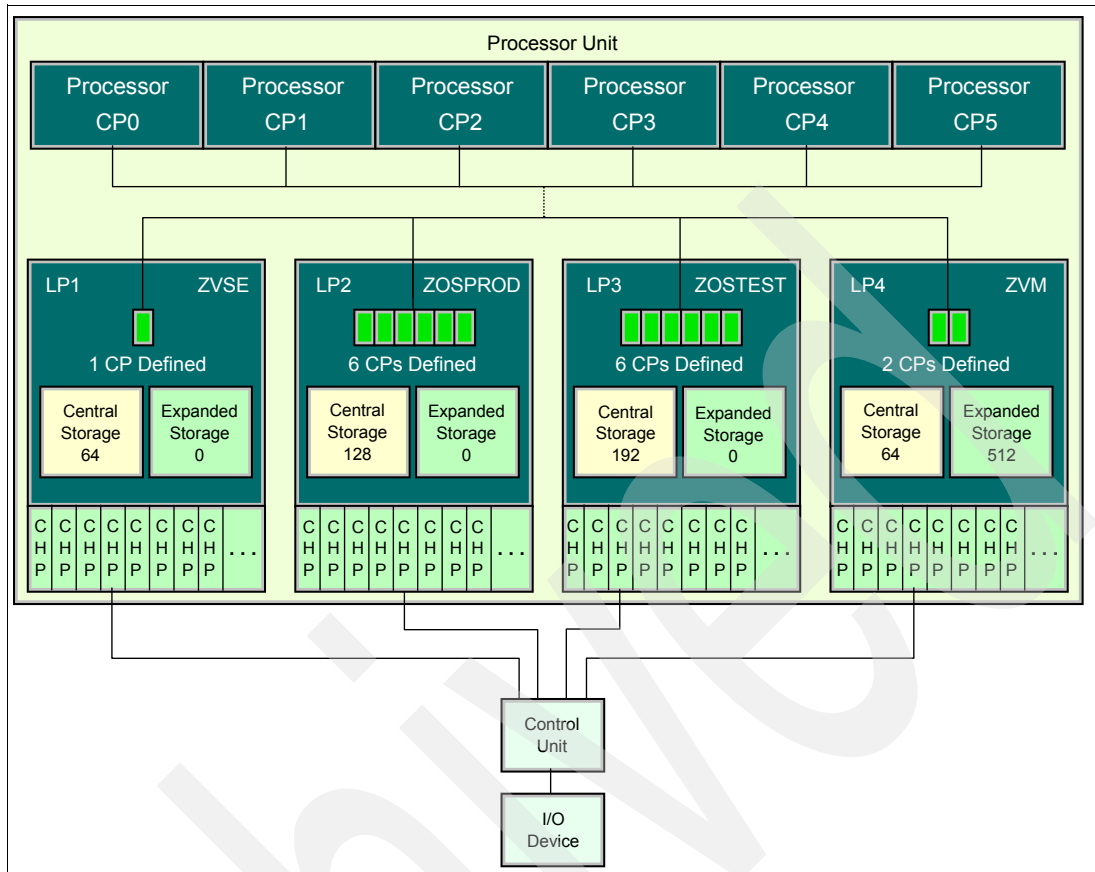


Figure 8-2 Central processor complex

8.2.1 Defining logical partitions

At least one logical partition is required on a machine; however, up to 60 partitions can be defined and active at any given time on some of the larger machines. Logical partition names and associated I/Os are set using the I/O Configuration Program (IOCP). The complete I/O definition results in an I/O Configuration Dataset (IOCDs), which is used to initialize the machine's I/O configuration during power-on reset. The following IOCP input statement shows an example of defining three logical partitions in the configuration:

```
RESOURCE PARTITION=(CSS(0),(LP01,1),(LP02,2),(LP03,3))
```

This statement in the IOCDs creates Channel Subsystem 0 for three partitions. The partitions are named: LP01, LP02, and LP03 and numbered 1,2, or 3.

Other logical partition configuration definitions are defined and contained in activation profiles. Each activation profile contains information, such as the number and type of processors and the amount of storage.

8.2.2 Partition modes

There are several configuration options to consider when defining logical partitions. One is the partition mode.

Depending on which of these partition modes are available on your System z server, you can define your logical partition to have any of these possible modes:

- ▶ ESA/390
- ▶ ESA/390 TPF
- ▶ Coupling facility
- ▶ Linux only
- ▶ z/VM

Table 8-1 provides additional information about the partition modes.

Table 8-1 Partition modes

Partition mode	z/OS	z/VM	TPF	Linux	z/VSE TM
ESA/390	Yes	Yes	Yes	Yes	Yes
ESA/390 TPF	No	No	Yes	No	No
Coupling Facility	No	No	No	No	No
Linux only	No	Yes	No	Yes	No
z/VM ^a	No	Yes	No	No	No

a. Other operating systems can be run as guests under z/VM when using this partition mode.

8.2.3 Processor types

Another configuration option is the definition of processor types. The types are:

- ▶ GP: General Processors
- ▶ zAAP: Application Assist Processor
- ▶ zIIP: Integrated Information Processor
- ▶ IFL: Integrated Facility for Linux
- ▶ ICF: Internal Coupling Facility

Table 8-2 provides the LPAR mode and the processor type.

Table 8-2 LPAR mode and processor type

Partition mode	GP	zAAP	zIIP	IFL	ICF
ESA/390	Yes	Yes	Yes	No	No
ESA/390 TPF	Yes	No	No	No	No
Coupling Facility	Yes	No	No	No	No
Linux only	Yes	No	No	Yes	No
z/VM	Yes	Yes	Yes	Yes	Yes

8.2.4 Shared and dedicated processors

Physical processors are either:

- ▶ Shared amongst all logical processors of the same processor type in any logical partition. Using shared processors is the best way to maximize machine utilization. Excess

processor resource from one partition can be used by another partition. This processor resource can be limited using per partition capping.

- Dedicated to a single logical processor in a single logical partition. Dedicating a processor provides the best performance. But, it does not allow excess processor time to be used by other logical processors.

Work is dispatched on a logical processor basis, not on the partition as a whole. Tasks from different partitions can be operating in parallel on different physical processors.

8.2.5 Processing weights

For shared processors, the amount of physical processor time that is given to a partition is based on the partition's logical processor weight, relative to the rest of the active partitions.

Dedicated processors get the full weight capacity of that processor.

8.3 LPAR advanced configuration options

In this section, we present the System z partition management advanced configuration options. Later in the chapter, we discuss how to implement these configuration options from the HMC/SE.

8.3.1 Dynamically adding and removing logical processors and storage

You can add or remove processors and storage to logical partitions dynamically by configuring or deconfiguring resources. This action requires that you specify reserve resources at partition activation to add processors and storage as needed, for example, you might initially need two general processors, but might want to reserve the right to add up to four more as workload requires. Newer servers, such as z10 EC and BC, allow you to change a partition definition for processors dynamically, so that pre-planning is not required.

8.3.2 Adjusting a partition's weight

The LPAR hypervisor associates logical processors with a logical partition. When there is work to be run on a logical processor, a physical processor is selected from the corresponding pool and used to dispatch the logical processor. Each logical partition is configured by the customer with a partition weight – a number between 1 and 999. The proportion of the total of all partition weights that is assigned to a particular partition determines how much of the processor resource it is entitled to use. As different logical processors become ready to run, the LPAR hypervisor dispatches them according to their relative processing weights, ensuring that each LPAR receives the amount of processor resource that the customer allocated.

Processing weights are specify the portion of the shared CP resources that are allocated to a logical partition. Although PR/SM always manages sharing logical partitions according to the specified processing weights, there are times when a logical partition receives either more or less than its processing share:

- A logical partition receives more than its processing share when there is excess CP capacity, provided that it has work to do and other logical partitions are not using their share.

- ▶ A logical partition receives less than its processing share when its workload demand drops below the capacity that is specified by its weight.
- ▶ A logical partition will not receive more than its processing share when the CP resources for that logical partition are capped.

If the logical processing weight for a partition is increased or decreased, the total system weight changes; therefore, all of the partition shares are recalculated. As a result, when there is contention for processor resource, the partition with the larger processing weight takes more physical general processors while the other partitions take less, for example, LP01 is defined to have a weight of 200, and LP02 has a weight of 100. If both are then activated:

- ▶ LP01 gets up to 200/300 of the total processing power of the machine
- ▶ LP02 gets up to 100/300 of the total processing power of the machine

If a third partition, LP03, is activated with a weight of 100, the weight distribution changes:

- ▶ LP01 gets up to 200/400 processing power of the machine
- ▶ LP02 gets up to 100/400 processing power of the machine
- ▶ LP03 gets up to 100/400 processing power of the machine

8.3.3 LPAR group capacity limits

An LPAR capacity limit is a means of controlling an LPAR's rolling four-hour average utilization in a subcapacity software pricing environment. If you established a capacity limit but the rolling four-hour average utilization of the LPAR does not reach the defined capacity, the software charges are based on the highest observed rolling four-hour average utilization. We review subcapacity software pricing in more detail in a future chapter.

Customers can define LPAR group capacity limits, specifying one or more groups of LPARs on a server, each with its own capacity limit, which allows z/OS to manage the groups in such a way that the sum of the LPARs' CPU utilization within a group does not exceed the group's defined capacity. LPAR group capacity limits can help provision a portion of a server to a group of logical partitions. This provision allows the CPU resources to float more readily between those logical partitions, but not to exceed the group capacity in total, resulting in more productive server utilization, which can be an effective tool for services bureaus to allow them to host many small or midsized companies' applications on one system, which minimizes administration costs.

8.3.4 Using Workload Manager

Workload Manager (WLM) is a component of the z/OS system. Workload Manager constantly monitors the system and adapts processing to meet the defined performance goals. It adds and removes logical partition processors and shifts weight between members of a sysplex on the same machine (a cluster).

WLM's LPAR CPU Management component, together with the LPAR clustering technology of the System server, provides the ability to dynamically manage workloads within an LPAR cluster that is comprised of multiple logical z/OS images on a single System z10™ server. Each logical partition is assigned a transaction goal (desired response time) and an importance level. WLM monitors how well each logical partition is achieving its goals. A donor/receiver approach is utilized to reapportion CPU resources between logical partitions in the cluster. When WLM LPAR Weight CPU Management decides to change the weight of an logical partition, it adjusts the receiver logical partition and the donor logical partition by a percentage of the current weight of the receiver.

8.3.5 Multiple Image Facility

The Multiple Image Facility (MIF) allows channel sharing among active logical partitions. Shared channels require that the channel subsystem establish a logical path for each channel image corresponding to an active logical partition that has the channel configured online. A logical path is a logical connection between a control unit and a channel for I/O operations to occur.

A failure in a shared channel path or I/O device can affect more than one LP; therefore, critical I/O devices (for example, DASD containing vital data sets) still need multiple channel paths. You can provide multiple shared channel paths (up to eight) to critical I/O devices. A control unit requires a logical path for each active logical partition that can access I/O devices through a shared channel path.

8.3.6 Multiple Channel Subsystem

With the advent of the multi-book multiprocessor family of server offerings, significant increases in total system capacity and scalability were realized. Essential to an increased processing capacity is the corresponding need for significant increases in total I/O scalability and connectivity. With the z990, increased I/O capacity is provided by increasing the number of physical I/O channels that can be configured to the system and by restructuring the physical channel subsystem (CSS) into logically distinct channel subsystems. This restructuring is commonly called the multiple-channel subsystem (MCSS) facility.

Multiple Logical Channel Subsystems (MCSS) supports a definition of up to 256 channels. A CSS is a collection of subchannels that directs the flow of information between I/O devices and main storage, relieves the processor of communication tasks, and performs path management functions.

In later chapters, we go into more detail about this subject.

8.4 Logical partition planning

A key part of the System z installation process is the *Systems Assurance Planning Record (SAPR) Guide*. IBM has long supported their customers prior to receiving their systems with a series of system planning sessions using the SAPR guide to ensure that the customer requirements are implemented and met during System z install.

Planning a customer's system z partitioning design is a key step to ensuring that the customers' expectations for System z performance, reliability, and availability are met. Understanding the customer's performance expectations and their plan for measuring and evaluating the performance of the processor provides the configuration options that are necessary to set up the System z LPAR configuration. Some of the key considerations when setting up the system z partitions (for current System z machines at the time of writing) are:

- ▶ Up to sixty (60) logical partitions are possible, each capable of running a different operating system.
- ▶ Specialty Engines: There are a number of specialty engines (IFL, zIIP, zAAP, ICF). This is the time when we plan the intended configuration of the specialty engines for the specific customer workload.
- ▶ The overall system performance is a culmination of the performance of all of the LPARs.

- ▶ Main storage and expanded storage are not shared between logical partitions. The appropriate amount of main storage and expanded storage must be assigned to each partition:
 - Main storage that is assigned to a logical partition is dynamically reconfigurable between logical partitions, subject to rules in the *PR/SM Planning Guide* and, *LPAR Dynamic Storage Reconfiguration*.
- ▶ ESCON channels can be configured to be shared using the Multiple Image Facility (MIF). Channel paths must be assigned to each logical partition.
- ▶ Operating System Consoles can be configured in a number of ways, depending on the operating systems that are involved. One method is using I/O Service Processor and the AWS3274 Console Unit Emulator code, which allows the IOSP Display and eNetwork Personal Communications to emulate a 3274 non-SNA control unit in support of 3270 devices, such as 3277s, 3278s, and 3279s. The Personal Communications emulation session might be created to act as both operating system consoles and user consoles.
- ▶ LPAR weights are enforced to within plus or minus 3.6 percent and typically track within plus or minus one percent.
- ▶ Logical processor shares are recalculated when a logical partition has a logical processor that is configured off-line or online.
- ▶ Unused CP cycles that are available to partitions are dispatched on a priority (weight) basis.
- ▶ Each individual logical partition is limited to a maximum of amount of main storage, which is model dependent due to memory sizes on the various models.
- ▶ The maximum number of supported devices for a partition is limited by the operating system that is used in that partition. In planning an I/O configuration, installation management must be aware of the maximum number of devices that are supported by the operating system that is run in each logical partition.
- ▶ The IOCP or HCD utility programs define the I/O configuration for the system and the partitions. During initialization of the CPC, the definitions of a selected IOCDS are transferred to the hardware system area (HSA). The IOCDS define the I/O configuration data that the CPC requires to control I/O requests. Channel paths in an IOCDS are assigned to one or more logical partitions. The results is an IOCDS that defines the I/O paths for all of the LPARs in the system.

8.5 Creating logical partitions

In this section, we discuss the process of creating logical partitions and the many operations that can be specified on the HMC/SE to implement the customer's performance objectives through System z partitioning. Logical partitions are created by defining the configuration characteristics of the partition in an image profile and then activating the logical partition by passing these characteristics to the LPAR Hypervisor.

8.5.1 Customizing and deleting activation profiles

Recall that the Activate task controls (on the HMC) that are used when starting up the system, including power-on reset, partition activation, and initial program loading of your operating system software. Activate is the primary function for CPC or CPC image start up. Activate senses the status of its object, and then performs only those actions that are necessary to get the object to an operational state. The Activate function uses profiles, which

contain the parameters that are needed to put the system in an operational state. The profiles are stored in each CPC Support Element.

Activation profiles are required for CPC and CPC image activation. They are used to tailor the operation of a CPC and are stored in the Support Element that is associated with the CPC. There are four types of activation profiles:

- ▶ Reset
- ▶ Image
- ▶ Load
- ▶ Group

Figure 8-3 illustrates the Customize/Delete Activation Profiles task that is available in the Operational Customization tasks list to open an image profile for a logical partition.

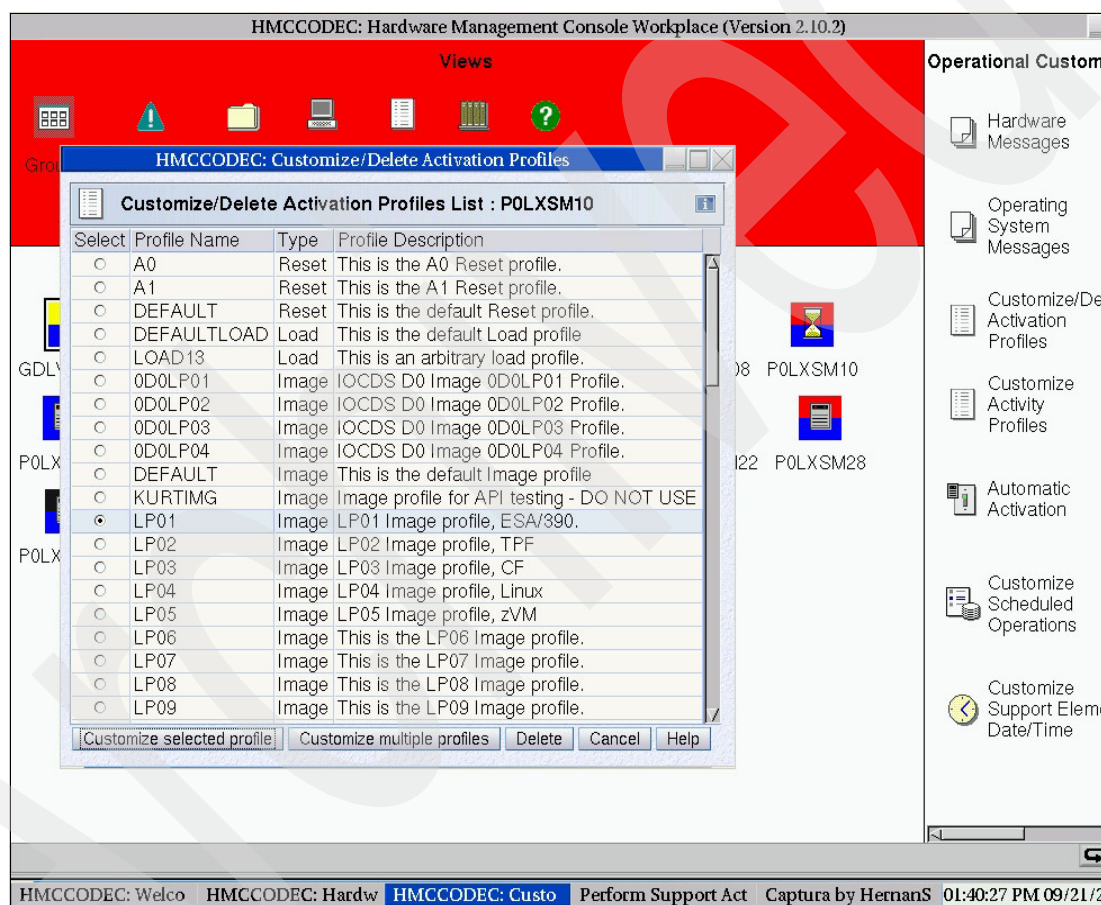


Figure 8-3 Customize/Delete Activation Profiles panel

In the next sections, we describe steps that you can use to work with these panels.

8.5.2 Defining the general settings

Open the General page to define the following logical partition characteristics, shown in Figure 8-4 on page 151:

- ▶ Profile name and description
- ▶ Logical partition identifier

- ▶ LP mode of operation
- ▶ Clock type assignment

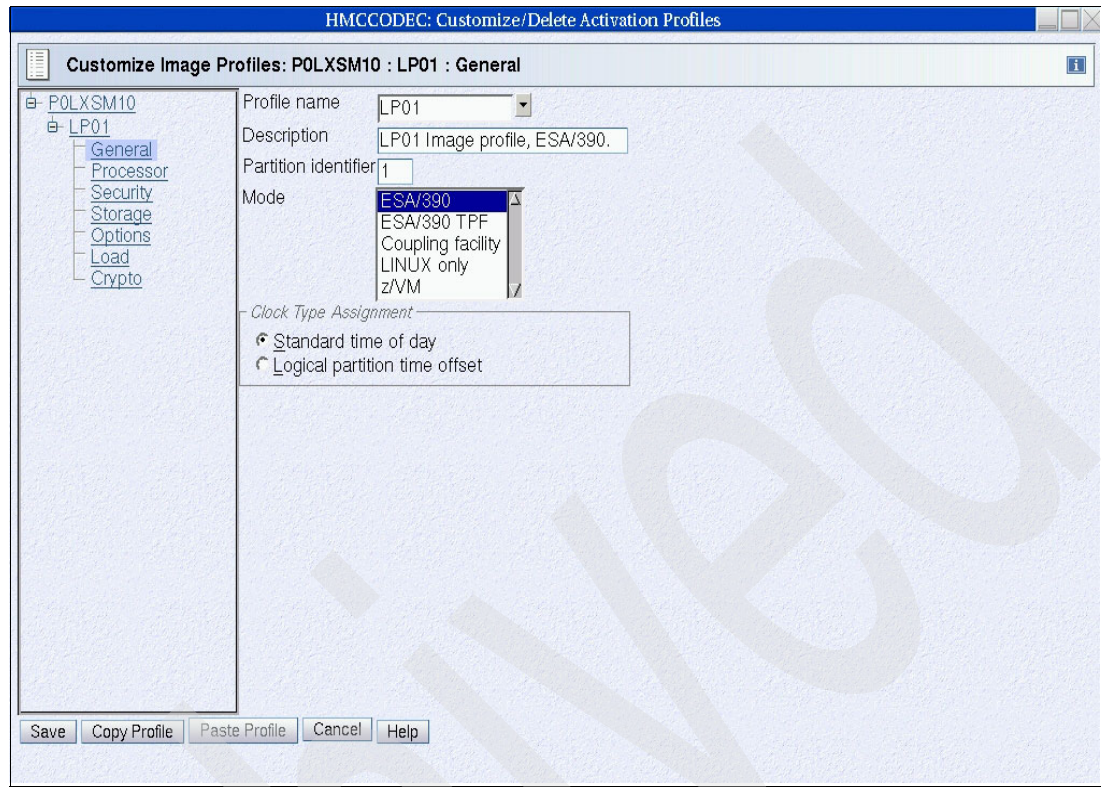


Figure 8-4 General settings for an image profile

The fields in this panel include:

- ▶ **Profile name and description:** Identify the image profile with a unique name (the same name as the logical partition) and a brief description.
- ▶ **Partition identifier:** This parameter identifies the logical partition and is used as the operand that is stored by the Store CPU ID instruction for each logical CP in the logical partition. The partition identifier must be unique for each active logical partition. Enter a hex value (X'00' through X'3F') for the LP.
- ▶ **Mode:** Select an operating mode from this scrollable list - ESA/390, z/VM, ESA/390 TPF, Linux-Only, or Coupling Facility mode.
- ▶ **Logical partition time offset:** Select this choice to set the logical partition's clock using an offset from the time of day supplied by its time source. Next use the Time Offset window to set the offset.

8.5.3 Defining logical processors

Open the Processor page, Figure 8-5 on page 152, to define the following logical partition characteristics:

- ▶ Group name
- ▶ Logical processor assignments (whether processors are Dedicated or Shared, the type of processors and the Number of initial and reserved processors)

- Non-dedicated processor details (Initial processing weight, Initial weight capping, Workload manager enablement, Minimum processing weight, and Maximum processing weight)

Note: Depending on the system configuration, some of the following parameters might not be present.

The screenshot shows the 'HMC CODEC: Customize/Delete Activation Profiles' window. The title bar is blue with white text. Below the title bar, the window is titled 'Customize Image Profiles: P0LXSM10 : LP01 : Processor'. On the left, there is a tree view with 'P0LXSM10' expanded, showing sub-items: 'General', 'Processor' (selected), 'Security', 'Storage', 'Options', 'Load', and 'Crypto'. The main area is divided into sections. At the top, 'Group Name' is set to 'GROUP2'. Below this is the 'Logical Processor Assignments' section, which includes a 'Dedicated processors' checkbox (unchecked) and a table for 'Select Processor Type'. The table has columns for 'Initial' and 'Reserved'. It lists three processor types: 'Central processors (CPs)' with Initial=1 and Reserved=2, 'zSeries application assist processors (zAAPs)' with Initial=2 and Reserved=1, and 'System z integrated information processors (zIIPs)' with Initial=0 and Reserved=1. Below the table is the 'Not Dedicated Processor Details for:' section, with radio buttons for 'CPs' (selected), 'zAAPs', and 'zIIPs'. Under 'CPs', there is a 'CP Details' section containing 'Initial processing weight' (set to 90, with a range of 1 to 999 and an 'Initial capping' checkbox), 'Enable workload manager' (checked), 'Minimum processing weight' (set to 10), and 'Maximum processing weight' (set to 190). At the bottom of the window are buttons for 'Save', 'Copy Profile', 'Paste Profile', 'Cancel', and 'Help'.

Figure 8-5 Processor panel of an image profile

The fields in the Processor panel of an image profile include:

- **Group Name:** If you choose to assign the processor to a group, select a defined Group from the drop-down list or create a new group.
- **Dedicated processors:** Select this option if you want the logical processors to be dedicated when the logical partition is activated. Do not select this option if you want the logical processors selected for this logical partition to be shared by other logical partitions.
- **Processor Type:** Select the type of processors that you want assigned to the logical partition. You can then specify the number of initial and reserved processors for each type. An initial processor is what is configured to the partition when it is activated. Selecting reserved identifies a specific number of processors that can be brought online after the partition is activated.
- **Not Dedicated Processor Details:** Select the processor (CPs, zAAPs, or zIIPs) to display details, such as Initial processing weight, Initial capping, and Enable Workload Manager. This option expands the panel to show the following settings for CPs Details:
 - **Initial processing weight:** Enter a value between 1–999 to set the processing weight for the Type of processor for a Depending on the system configuration, some of the following parameters might not be present. The default value is 10.

- Initial capping: Select this option to cap the CP resources for the Depending on the system configuration, some of the following parameters might not be present. Capping has no effect on logical partitions with dedicated CPs.
- Enable Workload Manager: Select this option so that CPU and I/O resources can be managed by WLM using clustering technology.
- Minimum processing weight: Select this option to establish a minimum processing weight that WLM allocates to the logical partition. Do not specify a value here unless you determine a true need for it in your configuration. Specifying a value here can needlessly constrain what WLM can do to optimize the management of your workload.
- Maximum processing weight: Select this option to establish a maximum processing weight that WLM allocates to this logical partition. Do not specify a value here unless you determine a true need for it in your configuration. Specifying a value here can needlessly constrain what WLM can do to optimize the management of your workload.

8.5.4 Security

Open the Security page, Figure 8-6, to define the following logical partition characteristics:

- ▶ Partition Security Options
- ▶ Counter Facility Security Options
- ▶ Sampling Facility Security Options

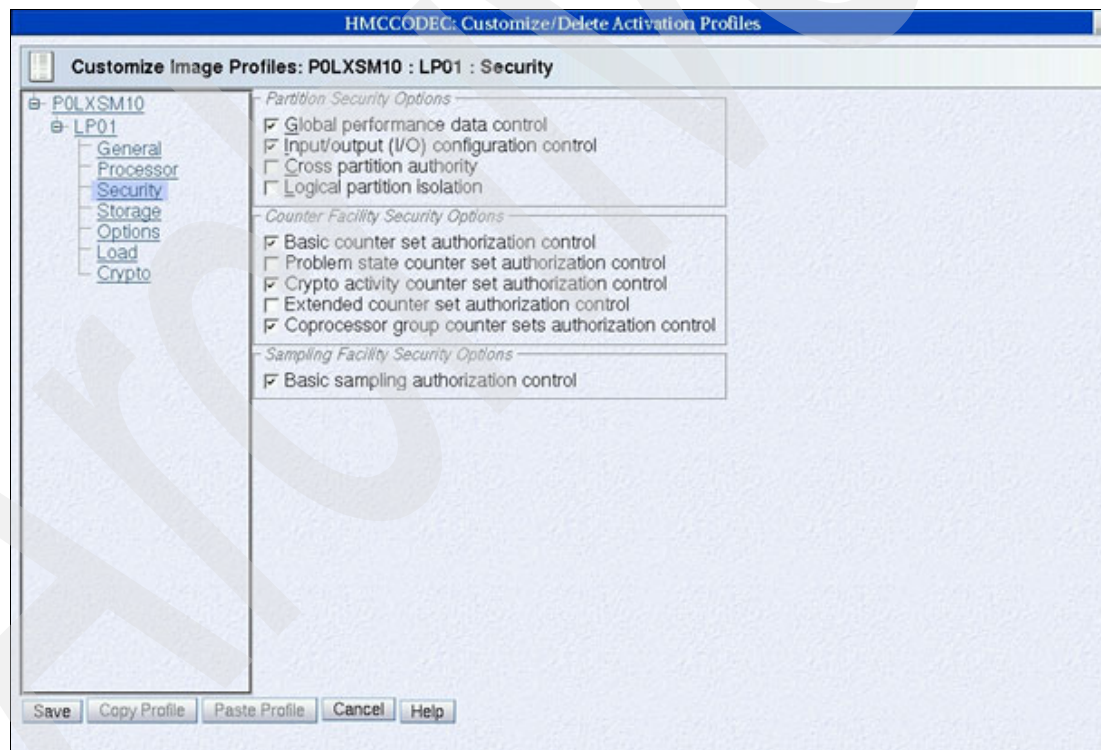


Figure 8-6 Security panel of an image profile

The partition security controls include:

- ▶ Global performance data control: Select this option to allow the logical partition to view the CPU utilization data and the Input/Output Processor (IOP) data for all logical partitions in the configuration. Not selecting this option only allows the logical partition to view its own CPU utilization data. Additionally, gathering FICON channel measurements require selection of this parameter. The default is selected.

Note: A logical partition that is running a level of RMF that supports FICON requires Control authority even if no FICON is installed.

- ▶ Input/output (I/O) configuration control: Select this option to allow the logical partition to read or write any IOCDS in the configuration and to make dynamic I/O changes. Additionally, this parameter allows the OSA Support Facility for z/OS, z/VM, and z/VSE to control OSA configuration for other logical partitions. The default is selected.
- ▶ Cross partition authority: Select this option to allow the logical partition to issue control program commands that affect other logical partitions, for example, perform a system reset of another logical partition, deactivate a logical partition, or provide support for the automatic reconfiguration facility. The default is not selected.
- ▶ Logical partition isolation: Select this option to reserve unshared channel paths for the exclusive use of the logical partition. The default is not selected.
- ▶ Counter Facility Security Options: Use this section to specify the counter facility security options for the logical partitions that are activated by the profile. The default is not selected:
 - Basic counter set authorization control: To indicate that the LPAR is allowed to use the basic counter set, select **Basic counter set authorization control**. The set can be used in analysis of cache performance, cycle counts, and instruction counts while the logical CPU is running.
 - Problem state counter set authorization control: To indicate whether the LPAR is allowed to use the problem-state counter set, select this option. The set can be used in analysis of cache performance, cycle counts, and instruction counts while the logical CPU is in problem state.
 - Crypto activity counter set authorization control: To indicate whether the LPAR is allowed to use the crypto-activity counter set, select this option. The set can be used to identify the crypto activities contributed by the logical CPU and the blocking effects on the logical CPU.
 - Extended counter set authorization control: To indicate that the LPAR is allowed to use the extended counter set, select this option. The counters of this set are model dependent.
 - Coprocessor group counter sets authorization control: To indicate that the LPAR is allowed to use the coprocessor-group counter sets, select this option. This set can be used to count the crypto activities of a coprocessor.
- ▶ Sampling Facility Security Options: Use this section to specify the sampling facility security options for the logical partitions that are activated by the profile:
 - Basic sampling authorization control: To indicate that the LPAR is allowed to use the basic-sampling function, select this option. The sample data includes an instruction address, the primary ASN, and some state information about the CPU. This option allows tooling programs to map instruction addresses into modules or tasks, and facilitates determination of hot spots. The default is not selected.

8.5.5 Storage

Open the Storage page, Figure 8-7 on page 155, to define the following LP characteristics:

- ▶ Central storage
- ▶ Expanded storage (which is no longer used by z/OS, but might be used by z/VM)

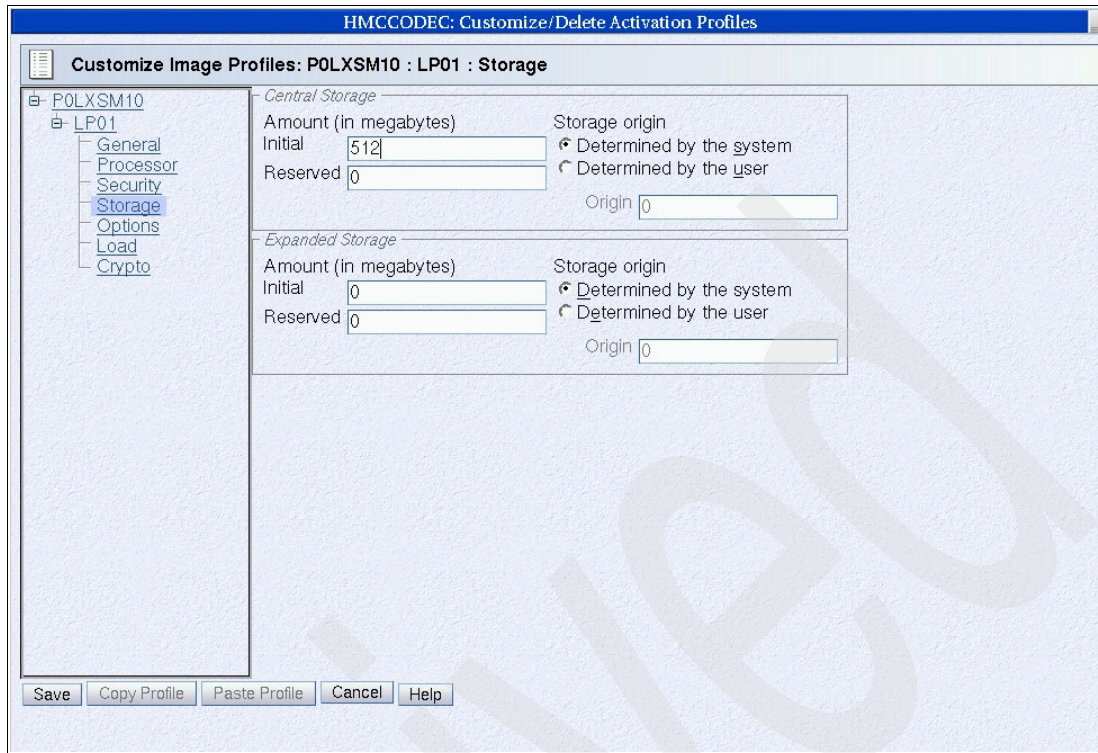


Figure 8-7 The Storage panel of an image profile

The main storage parameters include:

- ▶ **Initial:** Enter, in MB, the initial amount of main storage to be allocated to the logical partition at activation.
- ▶ **Reserved:** Enter, in MB, the amount of additional main storage requested for the logical partition. The reserved storage space is storage that can be dynamically brought online to the logical partition at some point after logical partition activation. Entering zero limits the main storage to the initial amount for the duration of the logical partition activation. Enter a value that is compatible with the storage granularity that is supported by your CPC.
- ▶ **Storage origin:** If **Determined by the user** is selected, enter, in MB, the central Storage origin for the logical partition. When the logical partition is activated, it is allocated at the origin that you specify here. Enter a value that is compatible with the Storage granularity supported by your CPC:
 - **Determined by the system:** Select this option if you want the system to allocate where the logical partition storage resides.
 - **Determined by the user:** Select this option if you want to allocate where the logical partition storage resides.

Expanded storage parameter descriptions are:

- ▶ **Initial:** Enter, in MB, the initial amount of expanded storage to be allocated to the logical partition at activation. Enter a value that is compatible with the Storage granularity that is supported by your CPC.
- ▶ **Reserved:** Enter, in MB, the amount of additional expanded storage that is requested for the logical partition. The reserved storage space is storage that might be dynamically brought online to the logical partition at some point after logical partition Activation. Entering zero limits expanded storage to the initial amount for the duration of the logical

partition activation. Enter a value that is compatible with the storage granularity that is supported by your CPC.

- ▶ Storage origin: Enter, in MB, the expanded storage origin for the LP. When the LP is activated, it is allocated at the origin you specify here. Enter a value that is compatible with the storage granularity that is supported by your CPC:
 - Determined by the system: Select this option if you want the system to allocate where the logical partition storage resides.
 - Determined by the user: Select this option if you want to allocate where the logical partition storage resides.

8.5.6 Options

Open the Options page, Figure 8-8, to define the following logical partition characteristics:

- ▶ Minimum input/output (I/O) priority
- ▶ Maximum input/output (I/O) priority
- ▶ Defined capacity

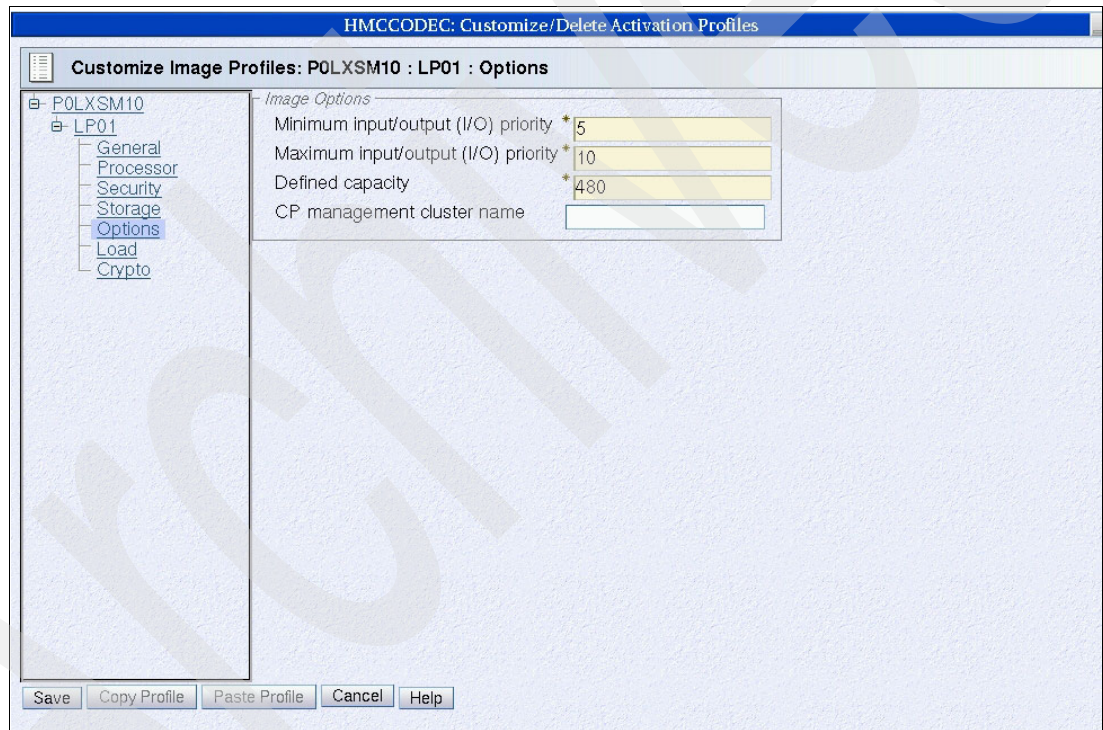


Figure 8-8 Options panel of an image profile

The options include the following fields:

- ▶ Minimum input/output (I/O) priority: Enter the minimum priority to be assigned to I/O requests from this logical partition.
- ▶ Maximum input/output (I/O) priority: Enter the maximum priority to be assigned to I/O requests from this logical partition.
- ▶ Defined capacity: Enter the upper bound in terms of millions of service units (MSUs) beyond which the rolling four-hour average CPU utilization cannot proceed.
- ▶ CP management cluster name: Enter the name of the Sysplex Cluster of which this logical partition will be made a member.

8.5.7 Load

Open the Load page, Figure 8-9, to define the following logical partition characteristics:

- ▶ Load during activation
- ▶ Load type
- ▶ Load address
- ▶ Load parameter
- ▶ Time-out value

The screenshot shows a window titled "HMC CODEC: Customize/Delete Activation Profiles". Inside, there's a tab labeled "Customize Image Profiles: P0LXSM10 : LP01 : Load". On the left, a tree view shows "P0LXSM10" expanded, with "LP01" selected, and "Load" highlighted under its sub-items. The main area contains several fields: "Load during activation" (unchecked checkbox), "Load type" (radio buttons for "Clear", "SCSI", and "SCSI dump", with "Clear" selected), "Load address" (text box with "0000"), "Load parameter" (text box), "Time-out value" (text box with "300", with a note "60 to 600 seconds"), "Worldwide port name" (text box with "0"), "Logical unit number" (text box with "0"), "Boot program selector" (text box with "0"), "Boot record logical block address" (text box with "0"), and "Operating system specific load parameters" (a large empty text area). There are also checkboxes for "Use dynamically changed address" and "Use dynamically changed parameter". At the bottom, there are buttons for "Save", "Copy Profile", "Paste Profile", "Cancel", and "Help".

Figure 8-9 Load panel of an image profile

The Load panel includes the following fields:

- ▶ **Load during activation:** Selecting this option allows initial program load of the operating system to occur automatically at logical partition activation. The default is not selected.
- ▶ **Load type:** Select **Clear**, **SCSI**, or **SCSI dump** to indicate whether to clear main storage during the load, do a SCSI IPL, or a SCSI dump.
- ▶ **Load address:** Enter the hex address of the I/O device that contains the operating system to be loaded automatically at logical partition activation.
- ▶ **Use dynamically changed address:** Select this option if you want to use the load address from a dynamically changed I/O configuration. This option and the Load address option are mutually exclusive.
- ▶ **Load parameter:** Enter a one-to-eight character optional IPL load parameter for loading an operating system on each activation of the logical partition. This option is useful for loading z/OS or z/VSE.
- ▶ **Use dynamically changed parameter:** Select this option if you want to use the load parameter from a dynamically changed I/O configuration. This option and the Load address option are mutually exclusive.
- ▶ **Time-out value:** Enter a time-out value in seconds to limit the amount of time for successful completion of the operating system load.

8.5.8 Crypto

Open the Crypto page (shown in Figure 8-10) to define the following LP characteristics:

- ▶ Control domain
- ▶ Usage domain
- ▶ Cryptographic candidate list
- ▶ Cryptographic online list

Note: Any changes to the Crypto Image Profile Page settings require a DEACTIVATE and ACTIVATE of the logical partition to have the change take effect. To verify that the active settings for the Cryptographic characteristics use the View LPAR Cryptographic Controls task that is available from the CPC Operational Customization tasks.

Index	Control Domain	Usage Domain	Crypto Number	Cryptographic Candidate List	Cryptographic Online List
0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	6	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
7	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	7	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
8	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	8	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
9	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	9	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
10	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	10	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
11	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	11	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
12	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	12	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
13	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	13	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
14	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	14	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
15	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	15	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Attention: You must install the 'IBM CP Assist for Cryptographic Functions' (CPACF) feature if a cryptographic candidate is selected from the list box; otherwise, some functions of Integrated Cryptograph Service Facility (ICSF) may fail.

Save Copy Profile Paste Profile Cancel Help

Figure 8-10 Crypto panel of an image profile

The crypto controls include:

- ▶ Control domain index: The Control Domain Index identifies the Crypto domain index(es) that can be administered from this logical partition when a partition is set up as the TCP/IP host for a TKE Workstation. The Control Domain Index selected must also be selected in the Usage Domain Index of the logical partitions to be managed by a TKE Workstation. If you are setting up the host TCP/IP in this logical partition for communicating with a TKE Workstation, the partition will be used as a path to this and the other domains' keys. Indicate all of the domains that you want to access, including this partition's own domain, from this partition as control domains.

Note: You can manage both master keys and operational keys from a TKE.

- ▶ Usage domain index: The Usage Domain Index identifies the cryptographic domain(s) that is assigned to the partition for all cryptographic coprocessors or accelerators that are

configured for the partition. The usage domain denotes where a specific partition's secure crypto data resides and where applications that are running on that partition will obtain the cryptographic data. If running z/OS, the usage domain index(es) selected must match the domain number(s) entered in the Options dataset when starting this partition's instance of ICSF.¹ As of z/OS 1.2 the usage domain specification in the Options dataset is only required if multiple usage domain index(es) are selected.

If running z/VM in a logical partition with guests, such as Linux or z/OS, a range of usage domain indices should be selected when assigning access to the cryptographic accelerator or coprocessor. A range allows more than one guest to have dedicated or shared access to the cryptographic queues. For further information, see the *z/VM CP Planning and Administration* and *z/VM Running Guest Operating Systems* documents.

The Usage Domain Index in combination with the Cryptographic Number selected in the Candidate List, must be unique across all partitions that are defined to the CPC. Although the Customize Image Profiles panel allows you to define duplicate combinations of Cryptographic numbers and Usage Domain Indexes, such logical partitions cannot be concurrently active. This is a useful option, for example, for backup configurations.

- **Cryptographic Candidate List:** The Cryptographic Candidate List identifies the Cryptographic numbers that are eligible to be accessed by this logical partition. Select from the list of numbers, from 0 to 15, that identify the coprocessor or accelerator to be accessed by this partition.

When the partition is activated, an error condition is not reported if a Cryptographic number that is selected in the Candidate list is not installed in the system. The Cryptographic number is ignored and the activation process continues. To plan ahead for a nondisruptive hardware install of any Crypto Express2 feature, the logical partition must be predefined with both the appropriate Usage Domain Index and Cryptographic number selected in its candidate list. It is possible to select all 16 index values (0-15) even before a Crypto Express2 feature is installed.

When a new Crypto Express2 feature is installed and its cryptographic numbers were previously selected in the Candidate list of an active partition, it can be configured on to the partition from the Support Element using the Configure On/Off option in the Crypto Service Operations task list. Selecting all of the values will not cause a problem if you have 16 or fewer partitions in your configurations that will use the Crypto Express2 feature. If you have more than 16 partitions that need to use coprocessors or accelerators, you must carefully assign the selection of cryptographic numbers across the partitions. Alternately, if you are running z/VM in one or more partitions, you might want to assign more than one Cryptographic number to each VM partition so that one or more than one type of Crypto card is available for VM guests to use, for example, you have 19 logical partitions defined and all require the use of Crypto Express2.

Some partitions will only need Crypto for acceleration of SSL handshakes using CEX2A, and others will need secure key operations from CEX2C. VM partitions might need access to several usage domains on several cards. Crypto Express2 is a single feature (or book) with two PCI-X Adapter Cards. These PCI-X Adapter Cards can be configured as accelerators or coprocessors. Each Crypto Express2 can be configured with two accelerators, or two coprocessors, or one accelerator and one coprocessor. Each of those PCI-X adapter cards will have a unique cryptographic number assigned. These numbers are assigned in sequential order during installation. A CEX2A or a CEX2C can be shared across 16 partitions. So, assume that the 19 partitions will share two CEX2C features and two CEX2A features.

It is very important to make the correct Crypto number assignments in the Cryptographic Candidate List for each of these logical partitions to avoid assignment conflicts.

¹ ICSF is a z/OS utility function that allows applications to interface with various cryptographic facilities.

Installing a Cryptographic Adapter requires the IBM CP Assist for Cryptographic Functions (CPACF) feature. See the *z/OS ICSF Application Programmer's Guide* and the *z/OS ICSF System Programmer's Guide* for complete information.

- **Cryptographic Online List:** The Cryptographic Online List identifies the Cryptographic numbers that are automatically brought online during logical partition activation. The Cryptographic numbers that are selected in the Online List must also be selected in the Candidate List. After partition activation, installed Cryptographic features that are in the partition Cryptographic Candidate list but not in the Cryptographic Online List are in a configured off state (Standby). They can be later configured on to the partition from the Support Element using the Configure On/Off option in the Crypto Service Operations task list. When the partition is activated, an error condition is not reported if the Cryptographic number that is selected in the Online list is not installed in the system. The Cryptographic number is ignored and the activation process continues. If a Cryptographic number that is selected in the Online list was configured off to the partition, it is automatically configured back on during the next partition activation.

Using the Image profile settings that we established in the previous subsections, when activated the logical partition is created as an ESA/390 mode partition that uses the COPT's time source, has one initial central processor, two reserved central processors, two reserved zIIPs, and two reserved zAAPs. Workload manager regulates the work that is running on these processors using the assigned minimum and maximum limits. Storage is allocated, crypto adapters are assigned, and no OS is loaded.

8.6 Activating a logical partition

To activate logical partition:

1. Locate the logical partition icon in the CPC Images Work Area.
2. From the Daily Tasks tasks list, Figure 8-11 on page 161, open the Activate task. This task activates the logical partition and, if automatic load is specified for a logical partition, automatically loads the logical partitions operating system as specified in the image profile for the logical partition.

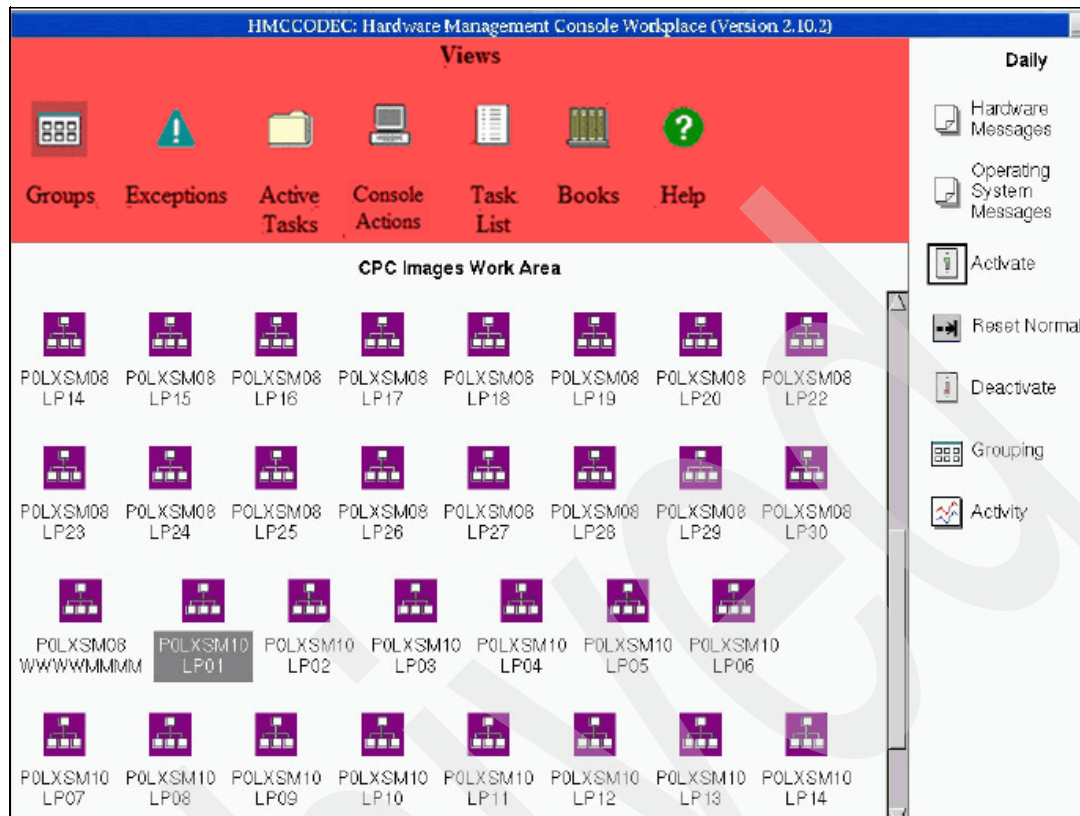


Figure 8-11 HMC Images workarea with LP01 selected for activation

Activation produces a conformation panel and a completion panel, as shown in Figure 8-12 and Figure 8-13 on page 162.

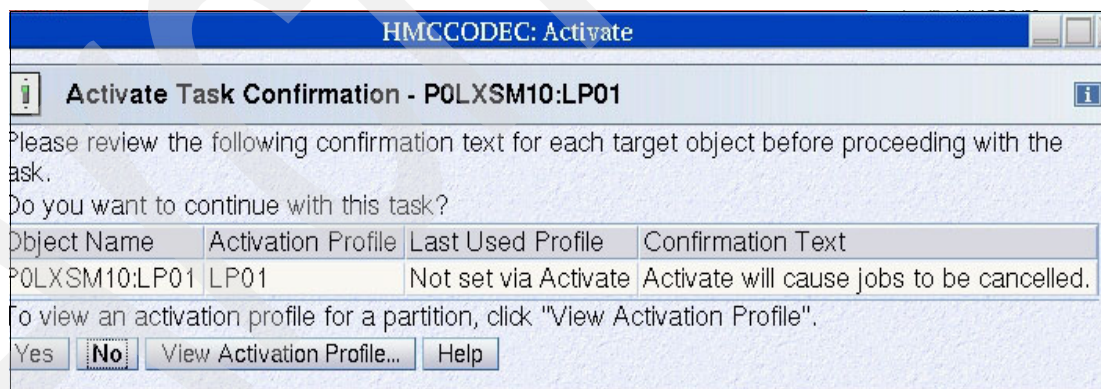


Figure 8-12 Activation confirmation panel

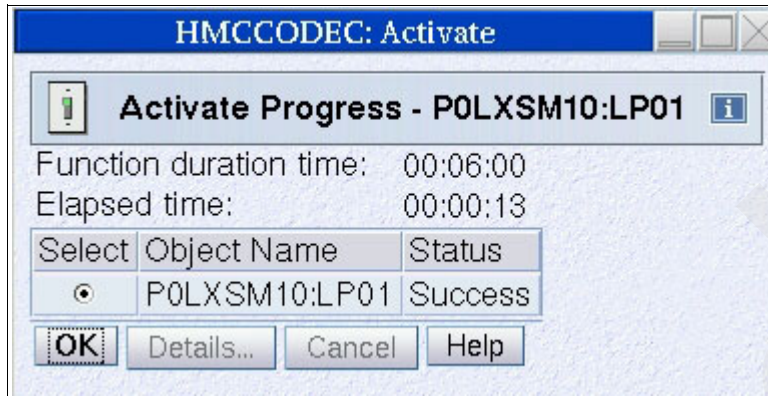


Figure 8-13 Activation completion panel

8.6.1 Initial program load

To perform a load on a logical partition or activate a load profile for a logical partition:

1. Locate the logical partition icon for a previously activated logical partition, and open the Customize/Delete Activation Profiles task that is available from the CPC Operational Tasks tasks list.
2. Select or, if necessary, customize or create a load profile for the logical partition.
3. Assign the load profile to the UP's activation profile, and exit the Customize/Delete Activation Profiles task. Save your changes.
4. In the CPC Image Work Area, locate the logical partition icon, and open the Activate task that is available from the Daily Tasks tasks list.

8.7 Deactivating the logical partition

To deactivate a logical partition:

Locate the logical partition icon, and from the Daily Tasks tasks list open the Deactivate task that is available. This task deactivates the logical partition and any operating system that is running in the logical partition.

Because deactivation is a disruptive operation that causes the data loss, confirmation of deactivation is required. The sequence of panels is shown in Figure 8-14 on page 163 through Figure 8-17 on page 164.

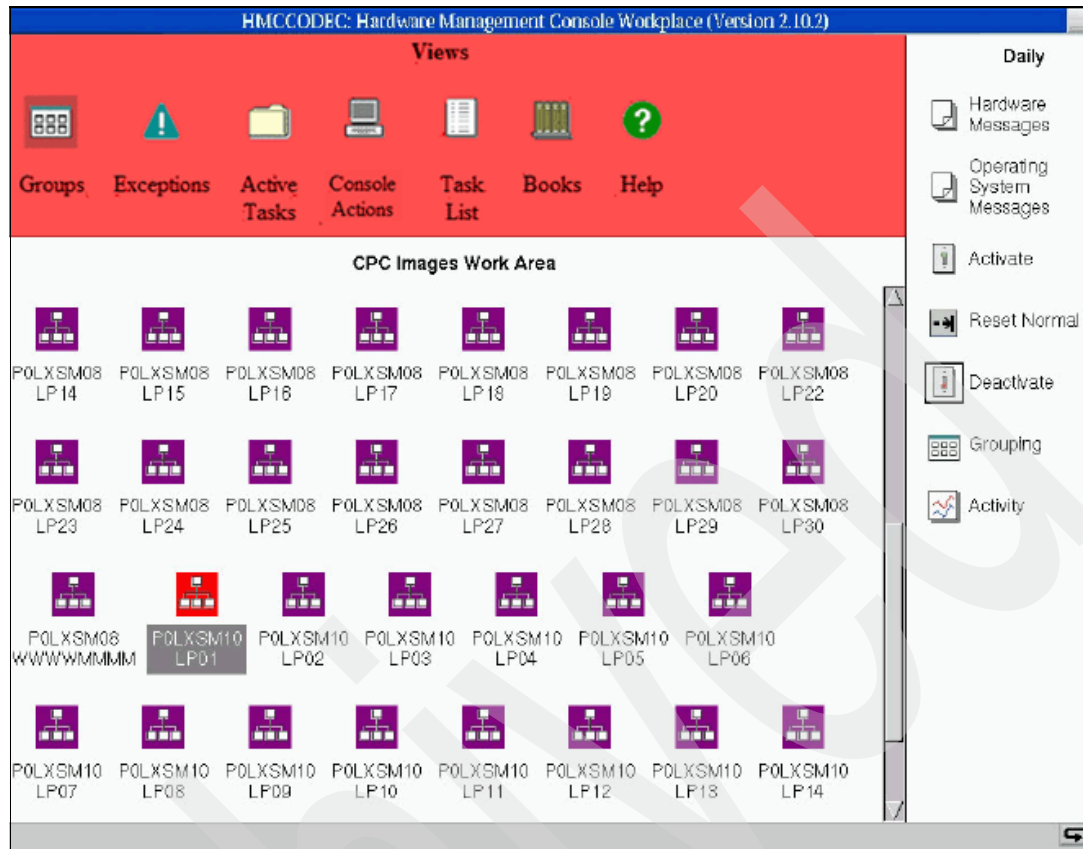


Figure 8-14 Images work area with LP01 selected for deactivation

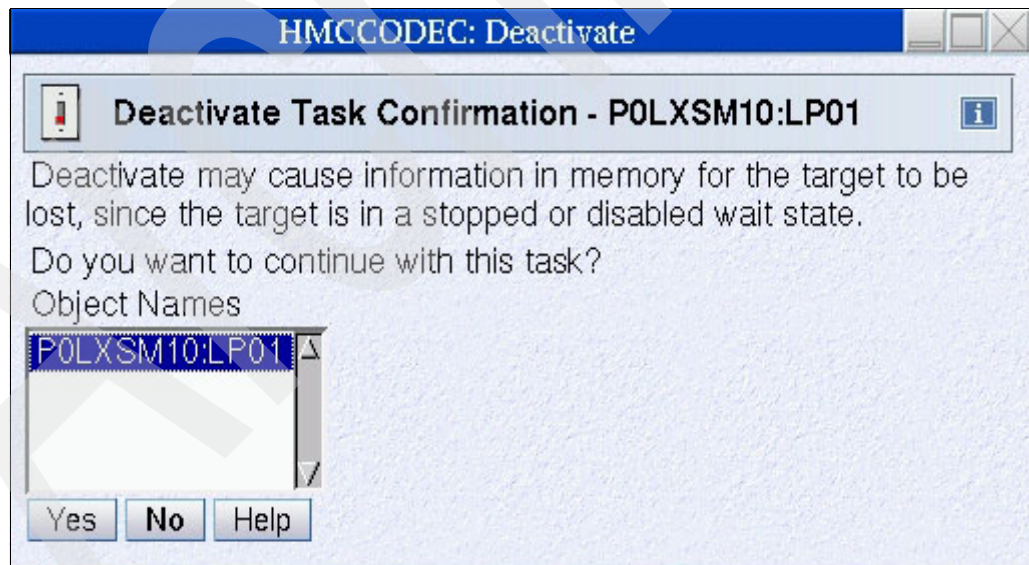


Figure 8-15 Deactivation confirmation panel

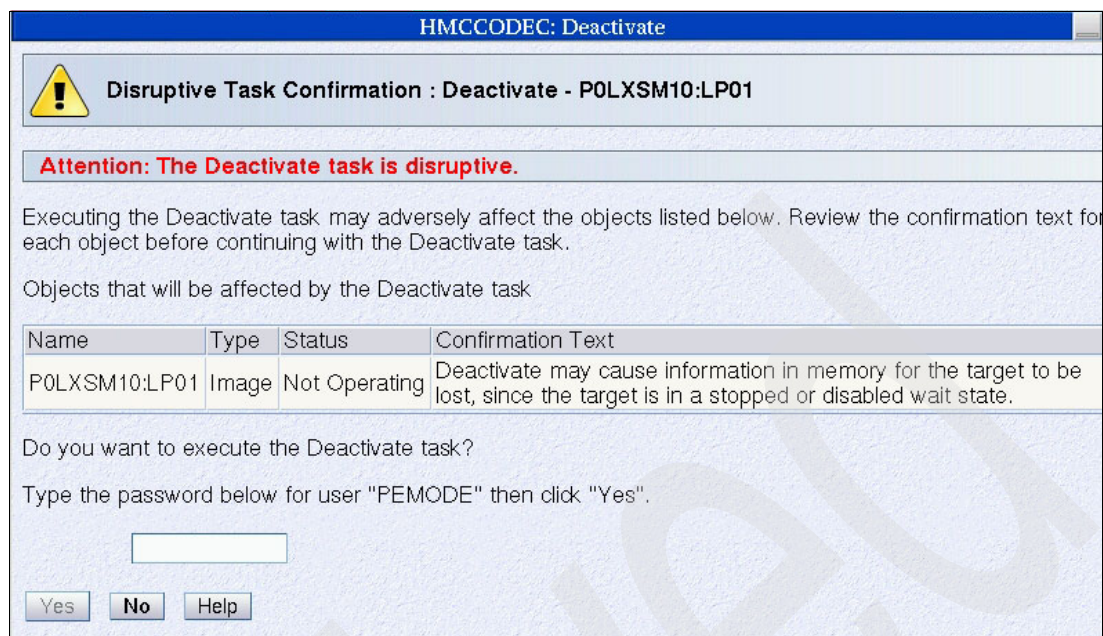


Figure 8-16 Second deactivation confirmation panel

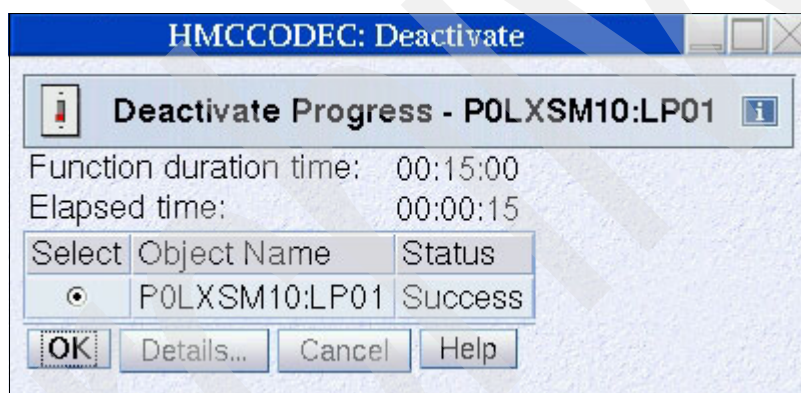


Figure 8-17 Deactivation completion panel

8.8 Dynamic LPAR change controls

In this section, we describe how to modify the configuration of an LPAR that is up and running. This is a very powerful set of functions that are used regularly by System Programmers on the HMC. Some of the things that we just described for initializing a partition can also be changed dynamically after the LPAR is up and running. In this section, we describe how to dynamically change the LPAR characteristics.

8.8.1 Change LPAR Controls panel

The Change Logical Partition Controls, Figure 8-18 on page 165, task allows you to select logical processor definitions to be changed dynamically on the system, in the image profile, or both. Dynamic changes take effect without reactivating the logical partition.

Changing the LPAR controls allows you to:

- Change the defined capacity value
- Enable/Disable Workload manager and set the related processing weight limits
- Change the processing weight
- Enable/Disable the capping of the processing weight

HMCCODEC: Change LPAR Controls

Change Logical Partition Controls - POLXSM10

Last reset profile attempted:
Input/output configuration data set (IOCDS): a1 eClipz1

CPs ICFs zAAPs IFLs zIIPs Processor Running Time

Logical Partitions with Central Processors

Logical Partition	Active	Defined Capacity	WLM	Current Weight	Initial Weight	Min Weight	Max Weight	Current Capping	Initial Capping	Number of Dedicated Processors	Number of Not dedicated Processors
LP01	Yes	0	<input type="checkbox"/>	90	90		190	No	<input type="checkbox"/>	0	1
LP02	No	0	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP03	No	0	<input type="checkbox"/>	0	105			No	<input type="checkbox"/>	0	1
LP04	No	6	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP05	No	0	<input checked="" type="checkbox"/>	0	20	10	30	No	<input type="checkbox"/>	0	1
LP06	No	0	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP07	No	0	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP08	No	0	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP09	No	0	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP10	No	0	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP11	No	0	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP12	No	0	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP13	No	0	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP14	No	0	<input type="checkbox"/>	0	10			No	<input type="checkbox"/>	0	1
LP15	No	0	<input type="checkbox"/>	0	0			No	<input type="checkbox"/>	3	0

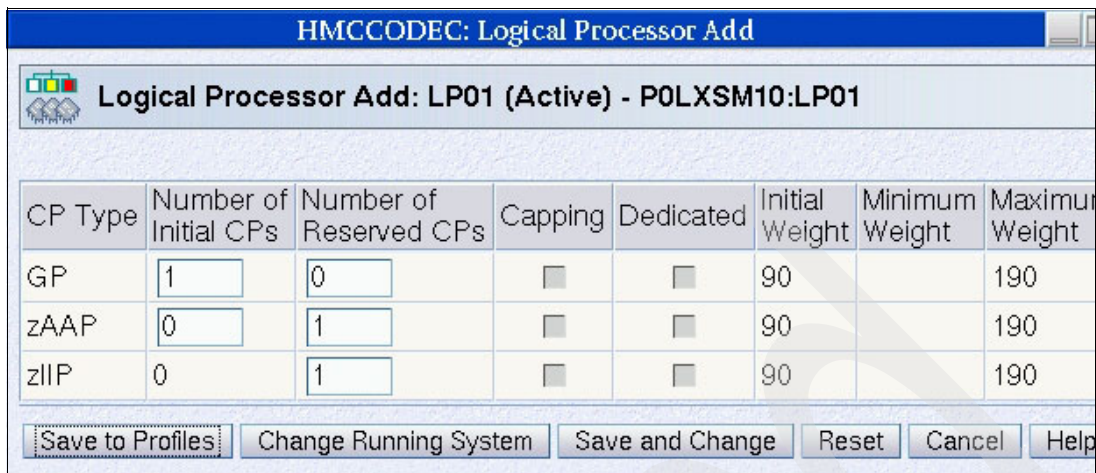
Save to Profiles Change Running System Save and Change Reset Cancel Help

Figure 8-18 Change LPAR Controls panel

8.8.2 Logical Processor Add panel

The Logical Processor Add task, shown in Figure 8-19 on page 166, allows you to:

- Increase the initial and reserved values for installed logical processor types.
- Add a reserved value and set weight and capping indicators for logical processor types that are not yet installed and have no reserved CPs defined.
- Increase the reserved value for logical processor types that are not yet installed and already have reserved CPs defined.



HMCCODEC: Logical Processor Add

Logical Processor Add: LP01 (Active) - P0LXSM10:LP01

CP Type	Number of Initial CPs	Number of Reserved CPs	Capping	Dedicated	Initial Weight	Minimum Weight	Maximum Weight
GP	1	0	<input type="checkbox"/>	<input type="checkbox"/>	90		190
zAAP	0	1	<input type="checkbox"/>	<input type="checkbox"/>	90		190
zIIP	0	1	<input type="checkbox"/>	<input type="checkbox"/>	90		190

Save to Profiles Change Running System Save and Change Reset Cancel Help

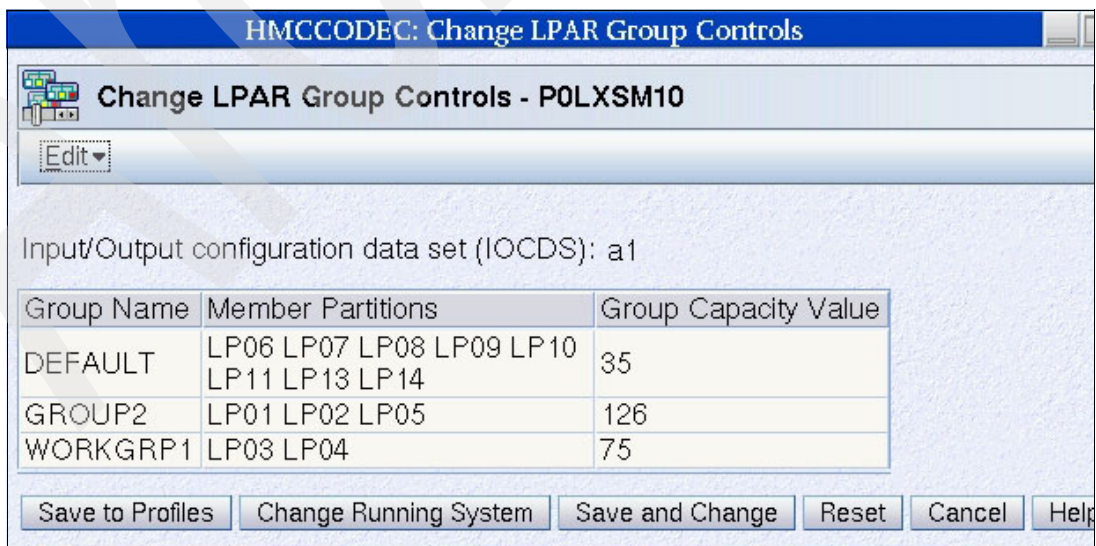
Figure 8-19 Logical Processor Add panel

The partition status (active/inactive) is indicated in the panel title, along with the logical partition name. If the logical partition is active, the current settings are displayed. If the logical partition is inactive, the settings in the image profile are displayed.

8.8.3 Change LPAR Group Controls panels

In the Change LPAR Group Controls panel, Figure 8-20, you can define LPAR group capacity limits, which allow you to specify one or more groups of LPARs on a server, each with its own capacity limit. This feature is designed to allow z/OS to manage the groups in such a way that the sum of the LPARs' CPU utilization within a group will not exceed the group's defined capacity. Each logical partition in a group can still optionally continue to define an individual logical partition capacity limit.

LPAR group capacity limits can help to provision a portion of a System server to a group of logical partitions, which allows the CPU resources to float more readily between those logical partitions, resulting in higher server utilization.



HMCCODEC: Change LPAR Group Controls

Change LPAR Group Controls - P0LXSM10

Edit

Input/Output configuration data set (IOCDs): a1

Group Name	Member Partitions	Group Capacity Value
DEFAULT	LP06 LP07 LP08 LP09 LP10 LP11 LP13 LP14	35
GROUP2	LP01 LP02 LP05	126
WORKGRP1	LP03 LP04	75

Save to Profiles Change Running System Save and Change Reset Cancel Help

Figure 8-20 Change LPAR Group Control pane

8.8.4 Change LPAR I/O Priority Queuing panel

Use the Change Logical Partition I/O priority queuing task, Figure 8-21, available from the CPC Operational Customization tasks list to set the minimum and maximum I/O priority Queuing values to be assigned to I/O requests from the logical partition. The operating system that is running in the logical partition must support I/O Priority Queuing for this feature to be effective.

HMCCODEC: Change LPAR I/O Priority Queuing

Change Logical Partition Input/Output (I/O) Priority Queuing - POLXSM10

Input/output configuration data set (IOCDS): a1
Global input/output (I/O) priority queuing: Disabled
Maximum global input/output (I/O) priority queuing value: 15

Logical Partition	Active	Minimum Input/Output (I/O) Priority	Maximum Input/Output (I/O) Priority
LP01	Yes	1	2
LP02	No	0	0
LP03	No	0	0
LP04	No	0	0
LP05	No	0	0
LP06	No	0	0
LP07	No	0	0
LP08	No	0	0
LP09	No	0	0
LP10	No	0	0
LP11	No	0	0
LP12	No	0	0
LP13	No	0	0
LP14	No	0	0
LP15	No	0	0
LP16	No	0	0

Figure 8-21 Change LPAR I/O Priority Queuing panel

8.9 Questions and discussions

These questions are intended to provide group discussions, led by an instructor.

1. How much of the information in this chapter is needed by Application Programmers?
2. Are all Systems Programmers expected to be completely familiar with HMC profiles and IOCDS preparations?
3. Partitioning can be complex. What percentage of mainframe installations actually use it?

4. How isolated is one partition from another partition? What are the data security implications?
5. With appropriate planning and manual operator intervention, memory can be moved from one partition to another. What are some of the complexities involved? How often is this done?

Automation

Automated management of System z hardware is accomplished through various APIs that run on the HMC. The HMC APIs allow the user to programmatically interface with the HMC's functionality through a number of languages, for example C, Java, and Rexx. In this chapter, we focus on client libraries that are written in Java, for example, there are two types of HMC automation APIs that are currently available, and both are based on industry standard protocols and data models:

- ▶ The Simple Network Management Protocol (SNMP) API
- ▶ The Common Information Management (CIM) API.

Support for the SNMP API is present on all System z servers, whereas the CIM API is only supported on the z10 and newer systems.

This chapter provides introductory information about SNMP, which is the basis for much of the automation that the HMC functions use. After reading this chapter the student should have a general understanding of SNMP architecture.

9.1 Simple network management protocol

SNMP is a transaction-oriented protocol that allows network elements to be queried directly. It is a simple protocol that allows management information for a network element to be inspected or altered by a System Administrator at network management station. SNMP is an IP network management protocol and is based on a manager-agent interaction. The SNMP manager communicates with its agents. Agents gather management data while the managers solicit this data and process it. An agent can also send unsolicited information, called a trap, to a managing system to inform it of an event that has taken place at the agent system, for example, an agent can send a trap of type **LinkDown** to the manager to inform it about the loss of a communication link with a particular device.

9.1.1 SNMP definitions

Discussions about SNMP can be confusing unless you understand the basic terminology. We suggest that you carefully review these definitions several times before you continue with the rest of the chapter:

SNMP agent	An implementation of a network management application that is resident on a managed system. Each node that is to be monitored or managed by an SNMP manager in a TCP/IP network, must have an SNMP agent resident. The agent receives requests to either retrieve or modify management information by referencing MIB objects. MIB objects are referenced by the agent whenever a valid request from an SNMP manager is received.
SNMP manager	Refers to a managing system that executes a managing application or suite of applications. These applications depend on MIB objects for information that resides on the managed systems.
SNMP subagent	The implementation of a network management application on a managed system, which interfaces with the SNMP agent for the purpose of expanding the number of MIB objects that an SNMP manager can access. SNMP agents have predefined MIB objects that they can access, which limits the managing application in regards to the type of information that it can request. The need to overcome this limitation brought about the introduction of subagents. A subagent allows the dynamic addition of other MIB objects without the need to change the agent. Whether a MIB object is referenced by the agent or the subagent is transparent to the managing system.
SNMP proxy agent	Acts on behalf of a managed system that is not reached directly by the managing system. A proxy agent is used when a managed system does not support SNMP or when a managed system supports SNMP but for other reasons it is more convenient to manage it indirectly, for instance, through the use of a proxy agent.
Management Information Base (MIB)	A logical database residing in the managed system that defines a set of MIB objects. A MIB is considered a logical database because actual data is not stored in it, but rather provides a view of the data that can be accessed on a managed system.
MIB object	A unit of managed information that specifically describes an aspect of a system, for example, CPU utilization, software name, hardware type, and more. A collection of related MIB objects is defined as a

	Management Information Base (MIB). A MIB object is sometimes called a MIB variable.
Instance	Refers to a particular representation of a MIB object. The MIB object that it represents can be thought of as a template for which one or more instances can be defined, depending on the type of MIB object. Actual values can only be assigned to instances of a MIB object.
SNMP community	An administrative relationship between an SNMP agent and one or more SNMP managers. Each community consists of a community name, an object access specification, and a list of SNMP managers' IP addresses. A community is used by an SNMP agent to determine which requests are to be honored.
Object Identifier (OID)	A dotted decimal string that represents a single MIB object.

9.1.2 The SNMP architecture

The SNMP architectural model is a collection of network management stations and network elements, such as gateways, routers, bridges, and hosts. These elements act as servers and contain management agents that perform the network management functions that the network management stations request.

The network management stations act as clients. They run the management applications that monitor and control network elements. SNMP provides a means of communicating between the network management stations and the agents in the network elements to send and receive information about network resources. This information can be status information, counters, identifiers, and other types of information.

The SNMP manager polls the agents for error and statistical data. The performance of the network is dependent upon what the polling interval is set at. The physical and logical characteristics of network objects make up a collection of information called a Management Information Base. The individual pieces of information that comprise a MIB are called MIB objects, and they reside on the agent system. These objects can be accessed and changed by the agent at the manager's request. Unsolicited data, called traps, can also be sent from the agent to the manager under certain conditions.

9.1.3 Goals of the SNMP architecture

The SNMP architecture explicitly minimizes the number and complexity of management functions that are realized by the management agent itself. This goal is attractive because among other benefits, it allows for:

- ▶ Reduced costs in developing management agent software to support the Protocol
- ▶ Few restrictions on the form and complexity of management tools
- ▶ Simplified, easier to implement management functions

A second goal of the protocol is that the functionality can be extended to accommodate additional, possibly unanticipated, aspects of network management. A third goal is that the architecture be, as much as possible, independent of the architecture and mechanisms of particular hosts or gateways.

9.1.4 SNMP model

The components that make up the SNMP model are:

- ▶ At least one network element to be managed (agent system) containing an agent.

- At least one network managing station (NMS) that contains one or more network management applications.
- A network management protocol for use by the NMS and the agent system to exchange network management information.
- At least one MIB that defines the information to be managed on the agent system.

Figure 9-1 illustrates the SNMP model.

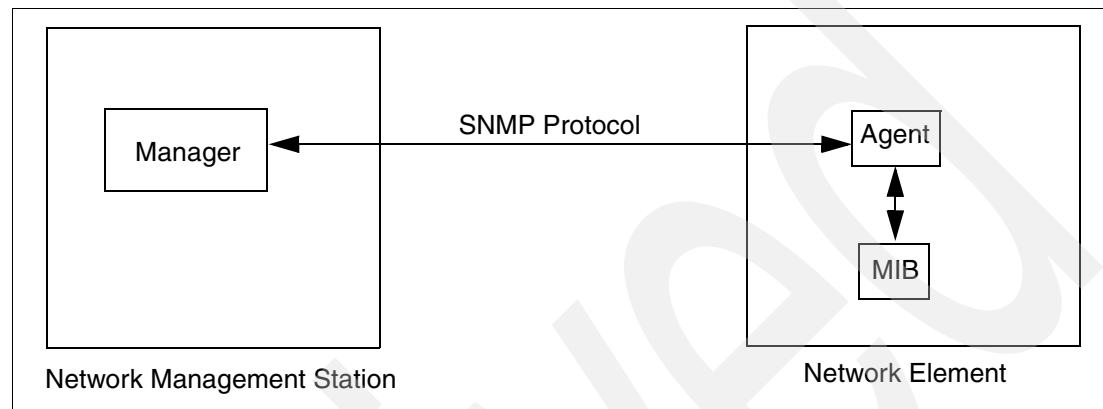


Figure 9-1 The SNMP model

9.1.5 User datagram protocol

The communication of management information among management entities occurs in SNMP through the exchange of protocol messages, each of which is entirely and independently represented within a single user datagram protocol (UDP) using the Basic Encoding Rules (BER) of ASN.1. These protocol messages are referred to as Protocol Data Units (PDU).

Consistent with the goal of minimizing complexity of the management agent, the exchange of SNMP messages requires a simple datagram service. For this reason, the preferred transport service for SNMP is the UDP, although the mechanisms of SNMP are generally suitable for use with a wide variety of transport services.

As a transport layer protocol, UDP uses the Internet Protocol (IP) as the underlying protocol. Two inherent characteristics of UDP provide for its simplicity. One of them is that UDP is unreliable, meaning that the UDP does not guarantee that messages are not lost, duplicated, delayed, or sent in a different order. UDP is also a connectionless protocol, because the only process that is involved is data transfer. However, UDP does provide a certain level of data integrity validation through checksum operations. UDP also provides application layer addressing because it can route messages to multiple destinations within a given host. Figure 9-2 on page 173 shows where SNMP and UDP operate within the TCP/IP protocol stack.

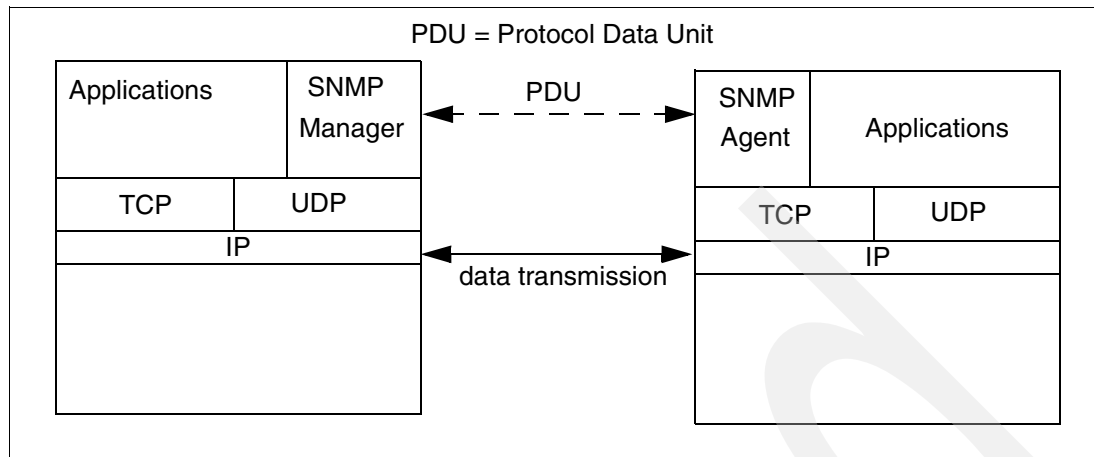


Figure 9-2 SNMP in the TCP/IP protocol stack

9.1.6 Asynchronous request/response protocol

Managing systems generate SNMP requests, and agent systems generate responses to these requests. After a request message is sent, SNMP does not need to wait for a response. SNMP can send other messages or realize other activities. These attributes make SNMP an asynchronous request/response protocol.

An agent system can also generate SNMP messages (traps) without a prior request from the managing system. The purpose of a trap message is to inform the managing system of an extraordinary event that occurred at the agent system. It must be noted that all request/response transactions are subject to the time delays that are inherent to all networks. The typical SNMP request/response primitives take place in the following manner:

1. Manager polls agent with a REQUEST for information.
2. Agent supplies information, which is defined in a MIB, in the form of a RESPONSE.

Figure 9-3 illustrates two time sequence diagrams. The top diagram shows a typical SNMP request/response interaction, while the bottom diagram shows a typical SNMP trap sequence.

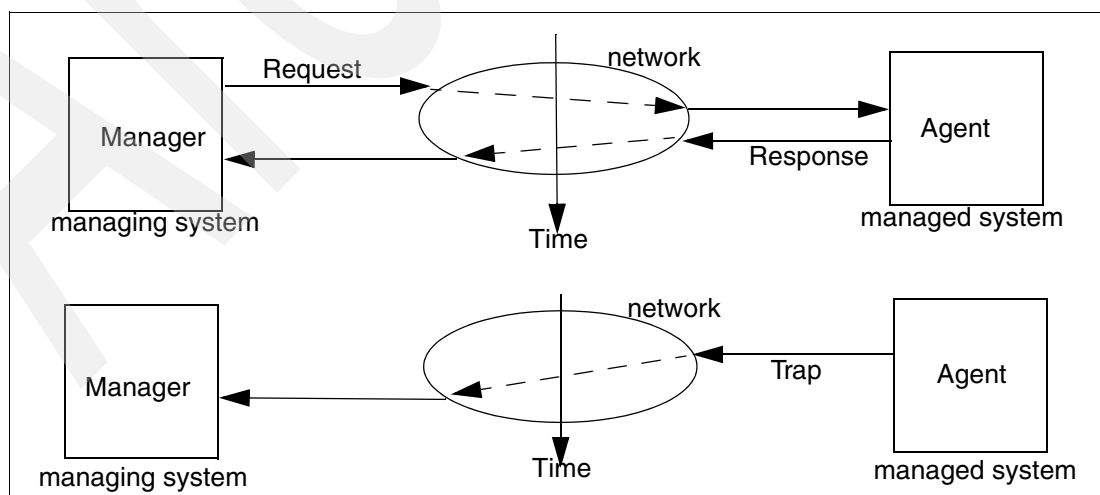


Figure 9-3 Asynchronous request/response protocol

9.1.7 SNMP agent

The SNMP agent has the following two responsibilities:

- ▶ To gather error and statistical data that MIB objects define
- ▶ To react to changes in certain MIB variables that are made by a managing application

In summary, the following steps describe the interactions that take place in an SNMP-managed network:

1. The SNMP agent gathers vital information about its respective device and networks.
2. The SNMP manager polls each agent for MIB information and can display this information at the SNMP manager station. In this manner, a Network Administrator can manage the network from a management station.

An agent also can send unsolicited data to the SNMP manager in the form of a trap. A trap is generally a network condition that an SNMP agent detects that requires the immediate attention of the Network Administrator.

9.1.8 SNMP subagent

A subagent extends the set of MIB objects that an SNMP agent provides. With a subagent it is possible to define MIB variables that are useful and specific to a particular environment, and then register them with the SNMP agent:

1. Requests for the variables that are received by the SNMP agent are passed to the process that is acting as a subagent.
2. The subagent then returns an appropriate answer to the SNMP agent.
3. The SNMP agent eventually sends an SNMP response with the answer back to the network managing station that initiated the request.
4. The network management station has no knowledge that the SNMP agent calls on other processes to obtain an answer. From the viewpoint of the managing application, the agent is the only network management application on the managed system.

9.1.9 SNMP manager

An SNMP manager refers to a network management station that runs a network management protocol and network management applications. SNMP is the network management protocol that provides the mechanism for management. Several different network management applications exist that can be used, such as IBM Director and Pandora Free Monitoring System. The network management application provides the policy to be used for management.

The network management applications rely on Management Information Base objects for information regarding the managed system, also called the agent system. Management systems generate requests for this MIB information, and an SNMP agent on the managed system responds to these requests. A request can either be the retrieval or modification of a MIB variable.

The agent system makes network and system information available to other systems by accessing the MIB objects and allowing configuration, performance, and problem management data to be managed by the SNMP manager, for example, a network manager can access the system description of a particular agent system by using the network management application to gain access to the agent system's **sysDescr** MIB object. To do

this, the managing application builds a message that requests a MIB object called **sysDescr**. This request is sent to the agent system where the agent decodes the message and then retrieves the information that is related to the **sysDescr** MIB object. The agent constructs a response with this information and sends it back to the managing application. When the application decodes the response, the SNMP manager can then display the agent system's description information to the user. Figure 9-4 shows the relationships among the SNMP entities.

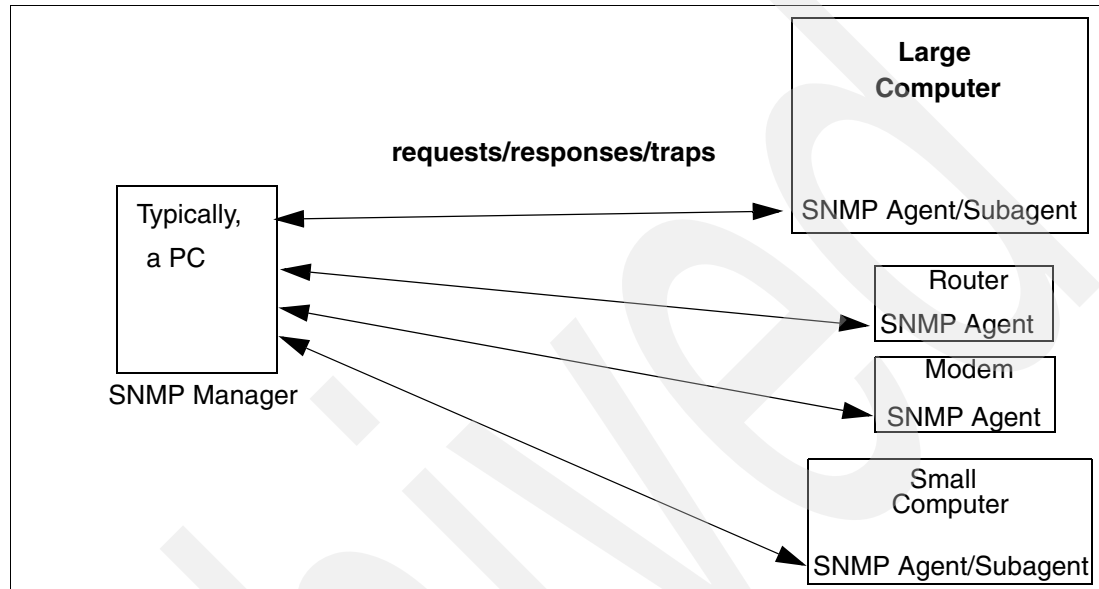


Figure 9-4 SNMP manager/agent/subagent relationship

9.1.10 Understanding Management Information Bases

The physical and logical characteristics of a system are what make up a collection of information that can be managed through SNMP. The individual pieces of information make up MIB objects. A Management Information Base is comprised of MIB objects that reside on the agent system, where they can be accessed and changed by the agent at the manager's request.

MIBs are classified as:

Standard MIB

All devices that support SNMP are also required to support a standard set of common managed object definitions of which a MIB is composed. Using the standard MIB object definition, MIB-II, you can monitor and control SNMP-managed devices.

Enterprise-specific MIB

SNMP permits vendors to define MIB extensions or enterprise-specific MIBs, specifically for controlling their products. An enterprise-specific MIB must follow certain definition standards, just as other MIBs must, to ensure that the information that they contain can be accessed and modified by SNMP agents.

Experimental MIB

Generally, new ideas and activities that are related to the Internet result in the definition of MIB objects. An experimental MIB is comprised of such objects. This approach offers the advantage that all new ideas must be proven while under experiment before they can be proposed for standardization.

9.1.11 Representation of management information

Since SNMP is used to manage a broad range of MIB objects, each and every one of these objects must be uniquely identified to provide unambiguous management capabilities. In this section, we briefly discuss the guiding mechanisms through which MIB objects are uniquely identified for management purposes and by which MIBs are structured.

Abstract syntax notation.1 (ASN.1)

ASN.1 is a formal language that the ISO originated that defines information that is exchanged by protocols, in the form of an abstract syntax, meaning that data is defined without regard to a specific machine architecture. ASN.1 is very useful because it does not allow any ambiguities in the information that it represents. ASN.1 defines managed objects and the PDUs that are exchanged by the protocols that manage those objects. ASN.1 provides two representations of the information that is defined by it:

- ▶ One representation can be read by humans
- ▶ The other representation is an encoded version of the same information, which the communications protocols use

Each managed object is described using an ASN.1 notation called OBJECT-TYPE. An OBJECT-TYPE definition consists of five fields, shown in Example 9-1, which describe a MIB object called **hrSystemUptime** (from the Host group in the MIB-II).

Example 9-1 hrSystemUptime OBJECT-TYPE

```
hrSystemUptime OBJECT-TYPE
SYNTAX TimeTicks
ACCESS read-only
STATUS mandatory
DESCRIPTION
The amount of time since this host was last
initialized. Note that this is different from
sysUpTime in MIB-II (RFC 1213) because sysUpTime is
the uptime of the network management portion of the
system.
 ::= { hrSystem 1 }
```

Here are descriptions of each of the fields that define an OBJECT-TYPE:

- ▶ OBJECT DESCRIPTOR (Name) is a textual name for the OBJECT-TYPE, for example, **hrSystemUptime** and **sysUpTime** are OBJECT DESCRIPTORS or names for an OBJECT-TYPE.
- ▶ SYNTAX defines the data type that is associated with the OBJECT-TYPE. ASN.1 constructs that define this structure, although the full generality of ASN.1 is not permitted. The ASN.1 type **ObjectSyntax** defines three categories of object syntax:
 - Simple
 - Application-wide
 - Simply constructed

INTEGER, OCTET STRING, NetworkAddress, Counter, Gauge, TimeTicks, SEQUENCE, and SEQUENCE OF are all examples of these types of syntax.

- ▶ ACCESS defines the level of access that is permitted for the managed object. It can be one of these levels:
 - Read-only Object instances can only be read, not set.
 - Read-write Object instances can be read, or set.

- Write-only Object instances can only be set, not read.
- Not-accessible Object instances cannot be read or set.
- ▶ STATUS defines the managed object's implementation requirement in terms of one of the following statuses:
 - Mandatory: A managed node must implement this object.
 - Optional: A managed node can implement this object.
 - Obsolete: A managed node need not implement this object any more.
- ▶ DESCRIPTION is a textual description of the semantics of the OBJECT-TYPE. In the case of non-enterprise-specific MIB implementations, take care to ensure that instances of the managed object fulfill their description because the MIB is intended for use in multivendor environments; therefore, it is vital that objects have consistent meanings across all machines.

Structure of management information

SGMP adopted the convention of using a well-defined subset of the ASN.1 language. SNMP continues and extends this tradition by utilizing a moderately more complex subset of ASN.1 for describing managed objects and for describing the Protocol Data Units (PDUs) that are used for managing those objects. In addition, the desire to ease eventual transition to OSI-based network management protocols led to the definition in the ASN.1 language of an Internet standard Structure of Management Information (SMI) and Management Information Base. For the sake of simplicity, SNMP uses only a subset of the Basic Encoding Rules of ASN.1.

The SMI is defined through ASN.1 and comprises a set of rules that describe management information in the form of MIB objects. The Internet standard RFC1155 documents the SMI in detail. Based on the SMI, management objects can be uniquely identified through the using an OBJECT IDENTIFIER that represents a specific object's name, that is, an OBJECT DESCRIPTOR. An OBJECT IDENTIFIER can be used for purposes other than naming managed object types, for example, each international standard has an OBJECT IDENTIFIER that is assigned to it for the purpose of identification. In short, OBJECT IDENTIFIERS are a means for identifying some object, regardless of the semantics that are associated with the object. An example is a network object or a standards document.

MIB naming conventions

MIB objects are logically organized in a hierarchy called a tree structure. Each MIB object has a label that is derived from its location in the tree structure. A label is a pairing of a brief textual description and an integer. An OBJECT IDENTIFIER is a sequence of non-negative integers that traverse a global tree for the purpose of identifying an object. The tree consists of a root that branches to connect to a number of labeled nodes, also called subordinates. Each node can, in turn, have children of its own, which are labeled. In this case, we might term the node a subtree or intermediate node. If a node does not have children, it is called a leaf node.

A fully qualified OBJECT IDENTIFIER for a particular MIB object contains all nodes, starting at the root and traversing the tree to an arbitrary level of depth until the desired leaf object is reached. The nodes are concatenated and separated by periods, in a format known as ASN.1 notation, for example, the mib-2 subtree is iso.org.dod.internet.mgmt.mib-2, which is concisely written in ASN.1 notation as 1.3.6.1.2.1.

Figure 9-5 on page 178 illustrates the MIB tree structure.

Note: Do not confuse the ASN.1 notation that OBJECT IDENTIFIERS use with the format that is used to identify IP addresses. Although both notations use periods in their format, they are not the same thing. The OBJECT IDENTIFIER for the sysContact object contained in the MIB-II is 1.3.6.1.2.1.1.4, as shown in Figure 9-5. Follow the labels from the root down to the leaf object.

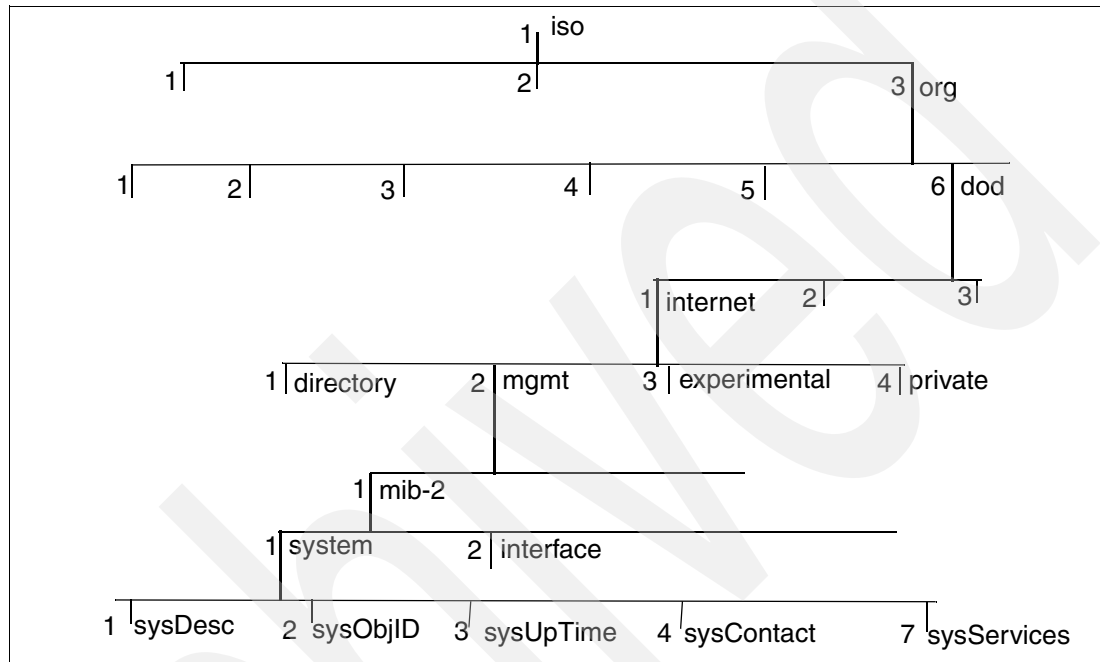


Figure 9-5 A view of the MIB tree structure

The standard MIB-II is registered in the mib-2(1) subtree. Experimental MIBs are registered in the experimental(3) subtree. Enterprise-specific MIBs are registered in the private(4) subtree. Each enterprise is assigned a number. IBM is assigned the enterprise number 2; therefore, all IBM enterprise-specific MIB objects have an OBJECT IDENTIFIER that starts with 1.3.6.1.4.1.2, corresponding to the tree structure iso.org.dod.internet.private.enterprises.ibm. Figure 9-6 on page 179 shows where IBM objects reside within the global tree structure.

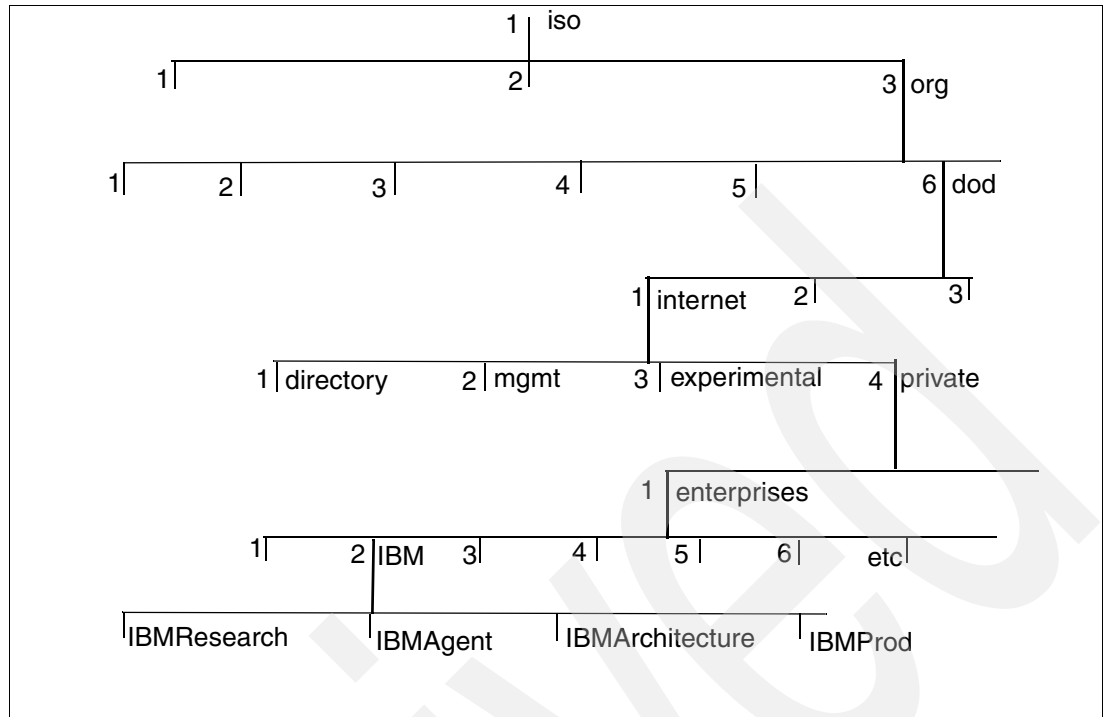


Figure 9-6 MIB tree structure for IBM objects

9.1.12 Identification of object instances

The names for all OBJECT-TYPES in the MIB are defined explicitly either in the Internet standard MIB or in other documents that conform to the naming conventions of the SMI. The SMI requires that conformant management protocols define mechanisms for identifying individual instances of those OBJECT-TYPES for a particular network element. It is these instances of MIB objects that are actually managed by a management protocol, thus it is important to understand what an instance is.

To understand what an instance of a MIB object is, it is helpful to view a MIB object as an abstract definition or template. A MIB object instance derives from the object's template; therefore, it retains all of the object's properties, but also has a particular value that is assigned to it that conforms to the MIB object's OBJECT-TYPE definitions. In a way, an instance can be viewed as a snapshot of a particular value of a MIB object.

Each instance of an OBJECT-TYPE that is defined in the MIB is identified in SNMP operations by a unique name, which is called its variable name. The term variable refers to an instance of a managed object. In general, the name of an SNMP variable is an OBJECT IDENTIFIER of the following form, where x is the name of an OBJECT-TYPE that is defined in the MIB, and y is an OBJECT IDENTIFIER suffix that, in a way specific to the named OBJECT-TYPE, identifies the desired instance:

x.y

The rules for determining the value of the suffix (y in the previous example) depend on the type of object. Basically, two types of objects are involved in a MIB:

- ▶ Tabular objects
- ▶ Scalar objects

We discuss both object types and the manner in which the suffix value is assigned in the next two sections.

Tabular objects

Tabular objects form part of a grouping in the form of a table. Tables are useful for managing multiple variables that share common properties but can have different values within a given device, for example, a personal computer usually has several different software entities that are installed on it. As shown in Figure 9-7, the **hrSWInstalledTable** object in the Host Resources MIB (RFC1514) defines an object called **hrSWInstalledName**, which is used to obtain a textual description of the software that is installed on a given system. **hrSWInstalledName** is a tabular object because it is part of a table, **hrSWInstalledTable**, which is also a tabular object itself. The different objects that make up the **hrSWInstalledTable**, such as **hrSWInstalledIndex**, **hrSWInstalledName** and others, make up the columns of that table. Through using values that are defined by **hrSWInstalledIndex**, which assigns a unique index number to each piece of software that is installed on the personal computer, it is possible to identify the instances of **hrSWInstalledName**. The instances of the table objects make up the rows of the table.

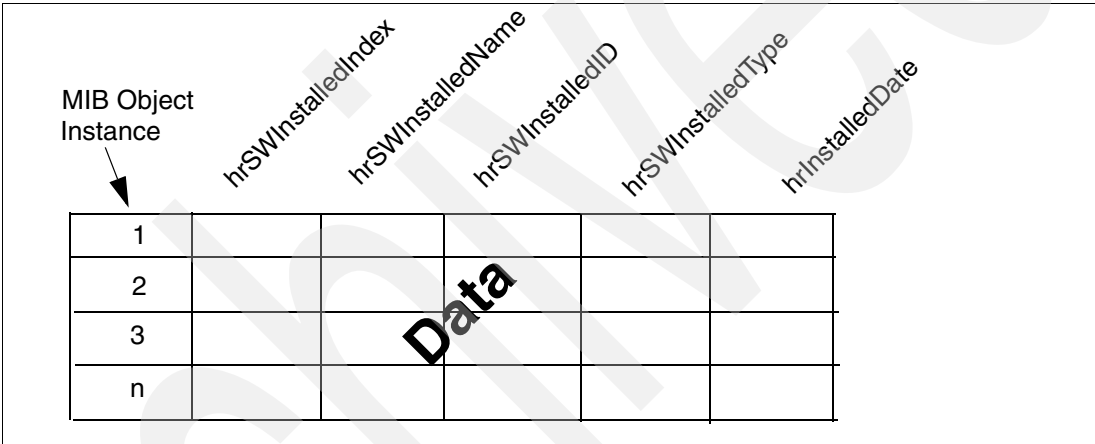


Figure 9-7 Graphical representation of the data

In the manner shown in Figure 9-7, it is possible to uniquely identify each software entity that is installed on the personal computer because each one is identified by the same object but by a different instance. Each instance then is the suffix for the OBJECT IDENTIFIER of each piece of software. To show this, the instance of **hrSWInstalledName** that is associated with the first software entity is:

hrSWInstalledName.1 or 1.3.6.1.2.1.25.6.3.1.2.1

The suffix then is the same as the instance. In this example, it is 1.

By using OBJECT IDENTIFIERS to identify the different instances, it is possible to create a lexicographic ordering over all of the object variables. This naming strategy admits the fullest exploitation of the SNMP GETNEXT operation because it assigns names for related variables so as to be contiguous in the lexicographical ordering of all variable names that are known in the MIB.

Scalar (non-tabular) Objects

Scalar objects do not form part of a table, and there is only one instance of these objects in a given device. Likewise, the only OBJECT IDENTIFIER suffix, y, for the OBJECT-TYPE, is zero, for example, the OBJECT IDENTIFIER for OBJECT-TYPE **sysUpTime** is simply:

sysUpTime.0 or 1.3.6.1.2.1.1.3.0

9.1.13 SNMP operations

To be consistent with its simplicity objective, SNMP contains few commands to execute its operations. SNMP supports two commands that managing systems can use to retrieve information from a managed system and one command to store a value into a managed system. All other operations are considered to be side-effects of these three commands.

As an example, SNMP does not contain an explicit reboot command. However, this action might be invoked by simply setting a parameter that indicates the number of seconds until system reboot. In addition to these commands, SNMP supports an event-driven mechanism that alerts managing stations of the occurrence of extraordinary events at a managed system.

The approach that SNMP is based on for network management makes it a simple, stable, and flexible protocol because it can accommodate new operations as side-effects of the same SNMP commands acting on new MIB variables, which do not requiring SNMP to be modified.

SNMP also specifies that if a single SNMP message specifies operations on multiple variables, the operations are either performed on all variables or on none of them. No operation is performed if any of the variables are in error.

Each SNMP operation is defined in a particular PDU:

GET	A request that originated by a managing application to retrieve an instance of one or more MIB objects. The specified instance is retrieved for each variable in the request, provided that community profile authentication was successful.
GETNEXT	A request that originated by a managing application to retrieve the next valid instance following the specified instance of one or more MIB objects, provided that community profile authentication was successful.
SET	A request that originated by a managing application to store a specific value for one or more MIB variables. All variables must be updated simultaneously or none of them.
GET-RESPONSE	Provides response data that is originated by an agent application and is sent back to the originator of a GET, GETNEXT, or SET request.
TRAP	An unsolicited message that originated by an agent application, which is sent to one or more managing systems within the correct community to alert them of the occurrence of an event. Traps include the following types: coldStart (0), warmStart (1), linkDown (2), linkUp (3), authenticationFailure (4), egpNeighborLoss (5), enterpriseSpecific (6)

9.1.14 Table traversal

An important use of the GETNEXT operation is the traversal of conceptual tables of information within the MIB. With table traversal it is possible to view all MIB objects as though they were organized in a table.

Let us assume that an SNMP application extracted the table index and software type for each entry in the installed software table (**hrSWInstalledTable**) of a particular network element. Suppose that this installed software table has the three entries in Table 9-1 on page 182, which shows MIB entries.

Table 9-1 MIB entries

hrSWInstalledIndex	hrSWInstalledType
1.3.6.1.2.1.25.6.3.1.1.3 4	(application)
1.3.6.1.2.1.25.6.3.1.1.1 2	(operatingSystem)
1.3.6.1.2.1.25.6.3.1.1.2 1	(unknown)

The management station sends to the SNMP agent a GETNEXT request that contains the indicated OBJECT IDENTIFIER values as the requested variable names:

```
PDU ==> GetNext Request (hrSWInstalledIndex,hrSWInstalledType )
```

The SNMP agent's GET-RESPONSE will contain:

```
PDU ==> GetResponse (( hrSWInstalledIndex.1 = 1),
                      ( hrSWInstalledType.1 = 2))
```

The management station continues with:

```
GetNextRequest ( hrSWInstalledIndex.1, hrSWInstalledType.1)
```

The SNMP agent responds:

```
GetResponse (( hrSWInstalledIndex.2 = 2)
              ( hrSWInstalledType.2 = 1))
```

The management station continues with:

```
GetNextRequest ( hrSWInstalledIndex.2, hrSWInstalledType.2)
```

The SNMP agent responds:

```
GetResponse (( hrSWInstalledIndex.3 = 3)
              ( hrSWInstalledType.3 = 4))
```

The management station continues with:

```
GetNextRequest ( hrSWInstalledIndex.3, hrSWInstalledType.3)
```

Because there are no further entries in the table, the SNMP agent returns those objects that are next in the lexicographical ordering of the known object names. This response signals the end of the routing table to the management station.

Note: For practical purposes, on an HMC this method of object discovery is very inefficient, and traversing the entire MIB tree is not feasible; instead, perform Get requests on provided group objects, which we describe in the next section.

9.2 SNMP on the HMC

SNMP is the protocol on which one of the HMC APIs is built. Libraries that IBM writes in C, Java, and Rexx act as the SNMP management station and abstracts many details of SNMP from the Programmer.

Figure 9-8 on page 183 illustrates the high-level architecture of the SNMP API. We focus on the Java interface. We assume that you have some background with the Java language and details of Java syntax. We do not cover running Java code.

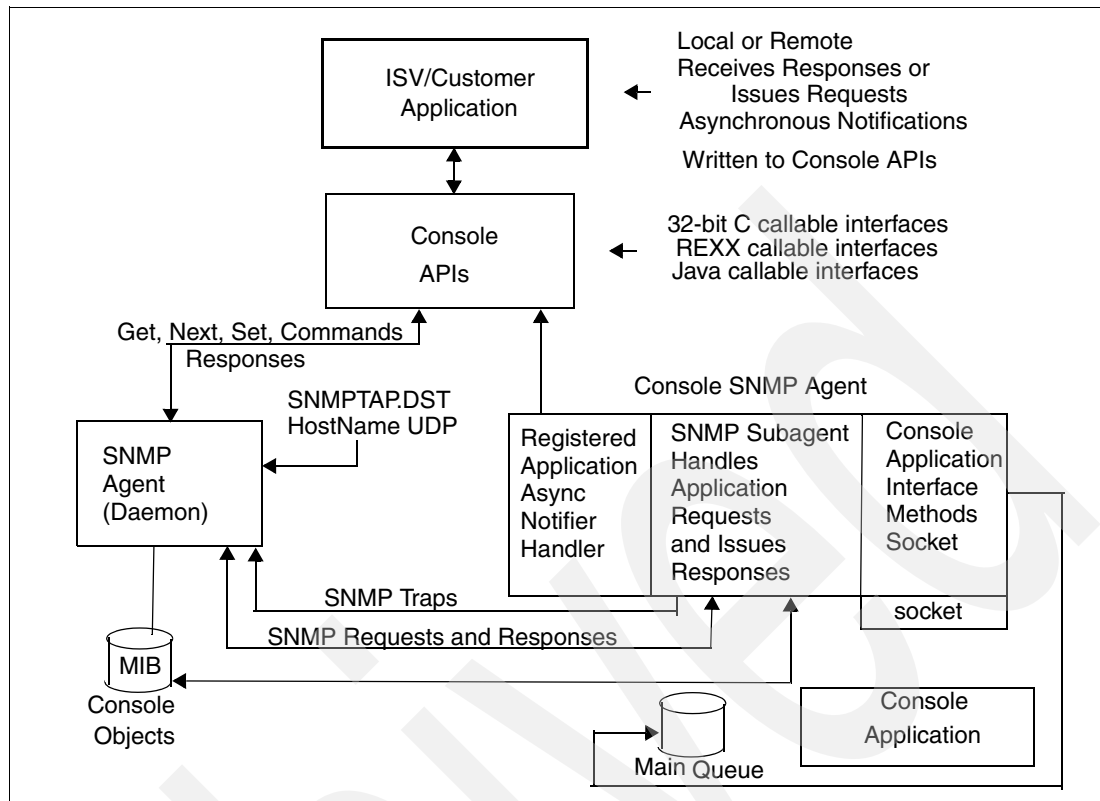


Figure 9-8 Console data exchange and commands API

9.2.1 Object identifier conventions

All of the objects that the Console application manages follow the same object identifier naming scheme. The naming scheme that the Console uses breaks the object identifiers into four distinct portions:

`prefix.attribute.group.object`

The meanings and options for each of these portions are:

► Prefix

This portion of the object identifier must be one of the following items:

- 1.3.6.1.4.1.2.6.42.0

An attribute of the Console object or the Console object itself.

- 1.3.6.1.4.1.2.6.42.1

An attribute of the Defined CPCs group object or the Defined CPCs group object itself.

- 1.3.6.1.4.1.2.6.42.1.0

An attribute of a Defined CPC object or a Defined CPC object itself.

- 1.3.6.1.4.1.2.6.42.2

An attribute of the CPC Images group object or the CPC Images group object itself.

- 1.3.6.1.4.1.2.6.42.2.0

An attribute of a CPC Image object or a CPC Image object itself.

- 1.3.6.1.4.1.2.6.42.3
An attribute of a user-defined group object or a user-defined group object itself.
- 1.3.6.1.4.1.2.6.42.3.0
An attribute of an object contained within a user-defined group object or an object contained within a user-defined group object itself.
- 1.3.6.1.4.1.2.6.42.4
A Console application command object.
- 1.3.6.1.4.1.2.6.42.5.0
An attribute of a Reset Activation Profile or a Reset Activation Profile object itself.
- 1.3.6.1.4.1.2.6.42.6.0
An attribute of a Image Activation Profile or a Image Activation Profile object itself.
- 1.3.6.1.4.1.2.6.42.7.0
An attribute of a Load Activation Profile or a Load Activation Profile object itself.
- 1.3.6.1.4.1.2.6.42.8.0
An attribute of a Group Activation Profile or a Group Activation Profile object itself.
- 1.3.6.1.4.1.2.6.42.9.0
An attribute of a Capacity Record object or a Capacity Record object itself.
- 1.3.6.1.4.1.2.6.42.10
An attribute of the Managed z/VM Virtual Machines group object or the Managed z/VM Virtual Machines group object itself.
- 1.3.6.1.4.1.2.6.42.10.0
An attribute of a z/VM Virtual Machine object or a z/VM Virtual Machine object itself.

► Attribute

This portion of the object identifier is used when specifying an object identifier for an attribute of an object. It is optional and when not specified results in an object identifier for the object itself.

Note: For a full list of attributes, see the document *System z Application Programming Interfaces* (IBM document order number SB10-7030-10).

► Group

This portion of the object identifier uniquely specifies which user-defined group this object identifier pertains to. It is optional and should only be used for the following object identifiers:

- User-defined groups
- User-defined group attributes
- Objects contained within user-defined groups
- Attributes of objects contained within user-defined groups
- Reset Activation Profile, Image Activation Profile, and Load Activation Profile objects (in this case the group value is used to identify the CPC object that the activation profile pertains to)
- Attributes of Reset Activation Profile, Image Activation Profile, and Load Activation Profile objects (in this case the group value is used to identify the CPC object that the

activation profile attribute pertains to). This value is generated using the name attribute of the CPC object

► Object

This portion of the object identifier uniquely specifies which object within a group this object identifier pertains to. It is optional and should only be used for the following object identifiers:

- Objects that are contained within a group.
- Attributes of objects that are contained within a group.
- Reset Activation Profile, Image Activation Profile, and Load Activation Profile objects.
- Attributes of Reset Activation Profile, Image Activation Profile, and Load Activation Profile objects.

This value is generated using the name attribute of the object.

As an example, suppose we want to build an object ID to retrieve the name of the CPC with the identifier 695366068. The prefix for CPC objects is 1.3.6.1.4.1.2.6.42.1.0, the attribute suffix for object names is 1.0, and the group portion of the object ID is irrelevant, so the full object ID would be 1.3.6.1.4.1.2.6.42.1.0.1.0.695366068. If we were to perform a GET request with this identifier, we would receive an OCTETSTRING with the name of the CPC.

It is inconvenient to have to look up these values when writing a program; therefore, all used prefixes and attributes are available as constant fields in the SNMP API. Convenience methods are also included for building object identifiers rather than manually concatenating strings.

9.2.2 The SNMP Java API

The Java API is contained within the package `com.ibm.hwmca.api`. This package is part of the `HWMCAAPI.JAR` jar file, which is located in the `D:\TOOLKIT` directory of the Hardware Management Console for Version 2.9.0 or earlier. The most up-to-date copy of this file is available on the IBM Resource Link facility at:

<http://www.ibm.com/servers/resourceLink>

Log into Resource Link, click **Services**, and then click **API**.

Figure 9-9 on page 186 is a sample program that simply connects to the HMC, issues a **get request** on the name of the console, and prints out the response.

```

import com.ibm.hwmca.api.*;

public class HwmcaTest {
    public static String community = "community";
    public static String ipaddr = "1.2.3.4";
    public static void main(String[] args) {
        APIConnection conn = null;
        try {
            conn = new APIConnection(community,ipaddr);

            int eventMask = APIEvent.HWMCA_EVENT_ALL_EVENTS;
            conn.Initialize(eventMask);
            System.out.println("Connection initialized");

            String oid = APIConnection.BuildAttributeId(
                APIConnection.HWMCA_CONSOLE_ID,
                APIConnection.HWMCA_NAME_SUFFIX);
            SNMPData getData = conn.Get(oid);
            assert(getData.getType() == SNMPData.TYPE_OCTETSTRING);
            System.out.println(getData.toStringNoNullTerm());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Figure 9-9 Java sample program

APIConnection

The `APIConnection` class maintains a connection to the HMC in addition to issuing SNMP **Get**, **Set**, and **GetNext** requests. It can also be used to selectively listen for Trap requests that the console generates.

`APIConnection` can either be initialized with the default constructor, and the IP address and community set with `setAddress()` and `setCommunity()`, or the host name and community can be passed to the constructor.

The `Initialize` method initializes the connection to the HMC and takes a single argument, which is a bitmask that matches the bit patterns of various events. To construct this bitmask, bitwise OR together selected static constant fields of the `APIEvent` class, which are prefixed with `HWMCA_EVENT`, for example, to listen for unacceptable status events and hardware message events, pass the value:

```
APIEvent.HWMCA_EVENT_EXCEPTION_STATE | APIEvent.HWMCA_EVENT_HARDWARE_MESSAGE
```

Requests are performed using the **Get**, **GetNext**, **Set**, **WaitEvent**, and **Command** methods of `APIConnection`. The first three perform SNMP requests of the same name, **WaitEvent** performs Trap requests, and **Command** allows more complex management commands to be performed by encoding special OIDs for Set requests.

Get

An SNMP **Get** request is coded as:

```
SNMPData Get(java.lang.String oid)
```

GetNext

An SNMP **GetNext** request is coded as:

```
SNMPData[] GetNext(java.lang.String oid)
```

Figure 9-10 is an example of using **GetNext**, where the entire MIB tree is traversed starting at a given OID. As we mentioned before, this is a very inefficient way of traversing the MIB tree; instead, **Get** requests must be performed on specific OIDs that return lists of other OIDs.

```
public void SNMPWalk(APIConnection conn, String walkOID) {
    SNMPData[] getNextData;
    do {
        getNextData = conn.GetNext(walkOID);
        if ((getNextData != null) &&
            (getNextData[0].getType() == SNMPData.TYPE_OBJECTID) &&
            (getNextData[0].toString().startsWith(argv[2]))) {
            printData(getNextData, getNextData[0].toString());
            walkOID = getNextData[0].toString();
        } else {
            walkOID = null;
        } // endif
    } while (walkOID != null);
}
```

Figure 9-10 Java **GetNext** example

Set

An SNMP **Set** request is coded as:

```
void Set(java.lang.String oid, SNMPData data)
```

WaitEvent

An SNMP **WaitEvent** request is coded as:

```
APIEvent WaitEvent()
```

WaitEvent waits for events that the target machine sends. Which events are received depends on the bitmask that is passed to **APIConnecton.Initialize()**.

Command

An SNMP **Command** request is coded as:

```
void Command(java.lang.String target, java.lang.String cmd, SNMPData[] parms, [
byte[] correlator ])
```

Command optionally takes an array of bytes called the correlator that is unique to the request and is used to correlate the response from the command with the call, which might be needed because commands are dispatched asynchronously.

The **Command** API uses special OIDs to represent each command that the HMC supports. These OIDs are used as the targets for **Set** requests. Each parameter for the command is encoded in ASN.1 data types, depending on what the command uses, and the arguments are concatenated as an ASN.1 Octet String. This string is used as the value for the **Set** request. In the case of optional arguments that are not used, ASN.1 null encodings must be used as placeholders because the arguments are positional.

Here is an example. Consider the **Activate** command, which takes a Managed Object, an optional activation profile name, and an option to force activation. The OID that represents

Activate is 1.3.6.1.4.1.2.6.42.4.1. So to perform an **Activate** on a specific Image, you must get the OID of the desired image (perhaps by doing a **Get** on the "Group Contents" attribute of the "Images Group" object: 1.3.6.1.4.1.2.6.42.2.23.0). Next, encode the activation profile name as an ASN.1 Octet String, or use null (0x00) if none is needed, and encode the force parameter as an ASN.1 Integer or use null as a placeholder. Then you concatenate these as one Octet String, and use it as the value of a **Set** request on 1.3.6.1.4.1.2.6.42.4.1.

After the application determines that the command request is successfully delivered to the Console, it must wait for one or more **APICommandResponseEvent** event notification object(s) for this command request, which is accomplished through using `APIConnection.WaitEvent()`. All applications are implicitly registered for this event type. The data that the **APICommandResponseEvent** event notification will contain, accessible through the `getEventData` method, is:

- ▶ An `HWMCA_TYPE_OBJECTID` that specifies the object identifier of the command for which this command response event was generated.
- ▶ An `HWMCA_TYPE_INTEGER` that specifies the return code value to be used to determine the success or failure of the command request that is associated with this command response event.
- ▶ An `HWMCA_TYPE_INTEGER` that specifies whether or not this is the last `HWMCA_EVENT_COMMAND_RESPONSE` event that will be issued for this command. A value of `HWMCA_TRUE` indicates that this event is the last, while a value of `HWMCA_FALSE` indicates that more `HWMCA_EVENT_COMMAND_RESPONSE` events are forthcoming.
- ▶ An `HWMCA_TYPE_OCTETSTRING` that specifies the name of the object that the event pertains to.
- ▶ An `HWMCA_TYPE_OCTETSTRING` that specifies the command correlator. This field is only present if the command was invoked with a correlator argument.

BuildId, BuildAttributeId

The statement to build an object identifier, with the necessary parameters, can be:

```
public static java.lang.String BuildId(
    java.lang.String prefix,
    java.lang.String attribute,
    java.lang.String group,
    java.lang.String object)
```

The parameters used are:

- ▶ **Prefix:** The prefix string to be used for the object identifier to be built. It can be one of these:
 - `APIConnection.HWMCA_CONSOLE_ID`
 - `APIConnection.HWMCA_CFG_CPC_GROUP_ID`
 - `APIConnection.HWMCA_CFG_CPC_ID`
 - `APIConnection.HWMCA_CPC_IMAGE_GROUP_ID`
 - `APIConnection.HWMCA_CPC_IMAGE_ID`
 - `APIConnection.HWMCA_GROUPS_GROUP_ID`

You can specify the prefix string as null, in which case an ID for the given attribute and group or object is generated.

- ▶ **Attribute:** The attribute suffix string to be used for the object identifier to be built, which can be specified as null, when building an ID for an object itself, as opposed to an attribute object ID.

- ▶ **Group:** The group name to be used for building the object identifier. You can specify the group name as null when building an ID for a predefined group or an object from a predefined group.
- ▶ **Object:** The object name to be used for building the object identifier. You can specify the object name as null when building an ID for a group object.

The statement to build an attribute ID is:

```
public static java.lang.String
    BuildAttributeId(java.lang.String oid,
                    java.lang.String attribute)
```

The parameters are:

- ▶ **OID:** The object identifier for the object for which the attribute identifier is to be built.
- ▶ **Attribute:** The attribute suffix string to be used for the attribute identifier to be built.

These methods build OIDs to be used with SNMP commands through concatenation. All available OID fragments are available as static const fields on `APIConnection`, and a full listing of them is available in the System z API reference.

9.3 Common Information Model

The Common Information Model (CIM) is a server management architecture that is controlled by the Distributed Management Task Force (DMTF), which is a collaborative industry standards organization. The DMTF provides a number of CIM-related specifications, including those that describe:

- ▶ An object-oriented representation of manageable objects and inter-object associations. This representation is normally expressed in Managed Object Format (MOF), which is a C++ class-like representation, but there is an equivalent XML representation too.
- ▶ An object-oriented framework that includes objects with properties and methods that define the common functionality of all systems. Management applications can be written against this framework that could theoretically be used to manage systems in a platform independent way. Vendors can also extend this framework with platform-specific functionality.
- ▶ An API that allows a management client to manipulate CIM objects (create, modify, call methods, and so on).
- ▶ The transmission of CIM requests in XML format over HTTP.

9.3.1 What is CIM

CIM is the information model that is used in the Web-Based Enterprise Management (WBEM) suite of management technologies. As a standard, CIM has several components:

- ▶ The meta-schema that describes what all schemas look like
- ▶ A language that is used to represent schemas (MOF)
- ▶ Schemas to represent common technologies, such as printers, disks, and servers
- ▶ Additional schemas that represent vendor-specific extensions to the common schema

The CIM schemas give an object-oriented view of managed information, and they are organized into a hierarchy of classes. Classes define properties and methods of objects, and classes can be related through special classes called Associations. The standard graphical representation of CIM is a variant of the Unified Modeling Language (UML).

CIM itself does not specify how operations are done on objects or details of the transport layer; instead, standards in WBEM define CIM operations over HTTP and the XML representation of CIM objects that are used for transport (CIM-XML).

Figure 9-11 on page 191 shows the elements of CIM.

CIM versus SNMP

CIM, as a newer standard, has advantages over SNMP for automated management of technologies. For one, CIM is fully object-oriented, which makes it easier to integrate with current enterprise software, making it easier to understand for programmers who are already familiar with object-oriented design.

The most important advantage CIM has over SNMP is increased interoperability. SNMP has no standardized way to represent computer system objects, so SNMP clients must be built differently depending on how the vendor represents objects in the MIB. CIM, in contrast, has an extensive object framework (the *schema*), which is supplemented with detailed specifications for the framework (*profiles*). This framework means that a client can be written to manage objects in a heterogeneous environment, with CIM servers from different vendors, as long as they are written to use the common functionality that CIM defines.

However, because a common model cannot represent all possible behaviors of hardware, it is likely that a vendor will want to extend the common functionality, for example, IBM provides System z-specific functionality through subclasses prefixed with IBMZ. In this case, to use the extended functionality, a client must have code that is specific to that product.

9.3.2 Managed Object Format

MOF is the standard language that is used to represent classes in the Common Information Model. The syntax of MOF is similar to the definitions of classes in C++, although there is additional syntax for CIM-specific details, such as qualifiers. Also, there are no private class elements because MOF represents the public interface of a class.

Class declaration

A class in MOF consists of:

- ▶ Qualifiers
- ▶ A name
- ▶ The names of any parent classes, properties, and methods

Properties and methods can also have qualifiers, which we explain in a further subsection.

Figure 9-11 on page 191 is a complete example of a CIM class in MOF format, perhaps for a caffeine-obsessed work environment. Directly after Figure 9-11 on page 191, we explain the elements.


```
[ Description ("A Coffee Maker") ]
class IACMM_CoffeeMaker : CIM_LogicalDevice
{
    [Key,
        Description("The device's brand")]
    string Manufacture;
    [Key,
        Description("The device's model")]
    string Model;
    [Key,
        Description("The device's serial number")]
    string SerialNumber;
    [Description("Amount of coffee left in Ounces"), Max(80)]
    uint32 CoffeeLeft=0;
    [Description("The device's current operating state."),
        Values(0, 1, 2),
        ValueMap("Off", "Brewing", "Warming")]
    uint16 OperatingState;
    [Description("Refill the coffee maker")]
    uint32 Refill();
};
```

Figure 9-11 Elements of CIM

Namespaces

In CIM, all classes must be prefixed with an identifier that is unique to the vendor, followed by an underscore. In our example, the CoffeeMaker class is in the namespace of the hypothetical IACMM or International Association of Coffee Maker Makers. CIM classes that relate to IBM System z have the prefix IBMZ. Standard classes that the DMTF provides in the CIM schema have the prefix CIM.

Data types

Table 9-2 contains a list of data types that are used for properties, method argument, and return types.

Table 9-2 Data type properties

Data type	Properties
uint8, uint16, uint32, uint64	An 8, 16, 32, or 64 bit unsigned integer
sint8, sint16, sint32, sint64	An 8, 16, 32, or 64 bit signed integer
string	A UCS-2 encoded string
boolean	A Boolean value (true or false)
real32, real64	A 32 or 64 bit IEEE-754 encoded floating point number
datetime	A string containing a date and time
classname ref	A reference to the class classname
char16	A single UCS-2 character

Properties

A property in MOF is written as:

```
datatype PropertyName [= DefaultValue];
```

Datatype is one of the data types listed in Table 9-2 on page 191. Optionally, you can include a default value for the property as shown, where DefaultValue is a literal or constant.

Methods

A method in MOF is written as:

```
returntype MethodName([IN] datatype parameter1, [IN] datatype parameter2, ...  
, [IN] datatype parameterN);
```

IN is an example of a qualifier and in particular is the qualifier applied to method parameter to indicate that the parameter is a value to be passed to the method. OUT can also be specified if values are returned through that parameter.

Qualifiers

Qualifiers are metadata about any MOF elements, including parameters, classes, methods, and method parameters but not including qualifiers themselves. They specify extra information about said elements, such as if they are modifiable or not. There are a large number of qualifiers that are specified in the CIM schema, and these are implicitly applied to all relevant elements with their default values, which are usually false.

A comma-delimited list of qualifiers enclosed in square brackets ([]) is given before any of the previously named MOF elements, for example, this sample defines a writable string property with a maximum length of 64 called ObjectName:

```
[ Write (true), MaxLen (64), Description ("The object's name") ] string  
ObjectName;
```

Boolean qualifiers can be specified without an argument, in which the argument is implicitly True, for example, instead of [Write(true)], you can specify just [Write].

Qualifiers also have a defined scope or list of MOF elements that they are applicable, for example, In, the method parameter qualifier, is only scoped to method parameters. Write is scoped to properties, and Static, the static method qualifier, is only scoped to methods.

A full list of qualifiers is available in the CIM specification.

While it is preferable to use one of the qualifiers that are specified in CIM, custom qualifiers can also be created. The only constraint is that custom qualifiers cannot override any standard CIM qualifiers.

Two frequently used qualifiers are the Values and ValueMap qualifiers that, when used together, enable properties to be value mapped or limited to a set of strings that are represented by integers internally, which is somewhat equivalent to an enum in C or C++. In the CoffeeMaker example above, the Values qualifier of the OperatingState property limits its values to 0, 1, and 2, which are mapped by the ValueMap qualifier to human-friendly values of Off, Brewing, and Warming. The value to string mapping is determined by the order of the qualifier arguments.

Inheritance

CIM classes can subclass other classes, and in fact most CIM objects have ManagedElement as their base class. User-specified classes are expected to refine already existing CIM classes through subclassing because it improves interoperability.

In MOF, a class' superclass is specified after a colon after the subclass' name, for example:

```
class Keyboard : UserDevice
```

The Override qualifier can be applied to elements in the subclass to override elements in the superclass. The name of the element to override is given as an argument. However, elements can only override elements of the same type, for example, a property can only override another property.

Instances

Special MOF syntax is used to denote instances of classes. The keyword `instance of` is followed by the name of the class and then a list of property values is given in a block, for example, in the `CoffeeMaker` example, constructing an instance with the coffee maker empty would look like Example 9-2.

Example 9-2 Instance construction for example CoffeeMaker

```
instance of IACM CoffeeMaker {  
    Manufacturer = "CoffeeCo";  
    Model = "8265";  
    SerialNumber = "48AZ6649";  
};
```

Keys

Keys are a way for instances of classes to be identified in the same namespace. One or more properties can be specified as a key in a class by using the `Key` qualifier. In our `CoffeeMaker` example, you might initially think that the device serial number uniquely identifies instances of the class; however, there is no guarantee that coffee makers from different manufacturers, or even different models from the same manufacturer, might not have the same serial number. A better way to uniquely-identify coffee makers is with the combination of the manufacturer, model, and serial numbers. Thus, in our example, the `Manufacturer`, `Model`, and `SerialNumber` properties are all given the `Key` qualifier, which means that a different value in any one of these three properties indicates a different instance.

Key property values must always be static and non-null. If a key property value changes, the original instance must be destroyed and a new one created.

Object paths

Object paths in CIM are the internal representation for references to objects, but they are also used when specifying reference literals, for example, in default values for properties and instance representations. Object paths are made by comparing the class name with the fields of the class qualified with Keys and their values, for example, the value of a reference to a `CoffeeMaker` object could be:

```
IACMM CoffeeMaker.Manufacturer="CoffeeCo",Model="8265",SerialNumber="48AZ6649"
```

Associations

Associations are a special type of class in CIM that represent relationships between two or more objects. Figure 9-12 on page 194 illustrates a CIM association. Each association has a name and relates two (or more) classes. This example shows that classes `CIM_System` and `CIM_LogicalDevice` are related by an association named `IBMZ_SystemDevice`. The diamond at the left end of the association connector indicates that this is an aggregating association. It is possible that the same two classes can be related by more than one association. It is also possible that a class can be related to more than one class by the same association.

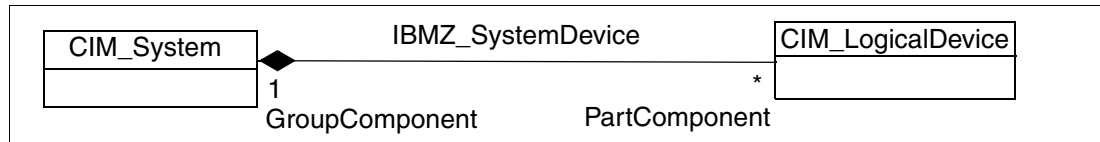


Figure 9-12 CIM association

For each class that participates in the association, there are two attributes: the *cardinality*, which specifies any limits on the number of times that instances of the class can participate in the association, and the *role*, which identifies the role that the class plays in the association. In the example, the CIM_System class plays the role of GroupComponent in the association. The cardinality of 1 indicates that each instance of this class can participate in this association exactly one time. In other words, each CIM_LogicalDevice instance must be associated to one and only one CIM_System instance. The CIM_LogicalDevice class plays the role of PartComponent in the association. The asterisk indicates that there is no limit on the number of CIM_LogicalDevice instances, which means that each CIM_System can aggregate zero or more instances of CIM_LogicalDevice instances.

Aggregations are a stronger form of associations, where an object on one side of the relationship is considered to have ownership of the other, and the child cannot belong to another object.

Compositions are a further refinement of this concept. In a Composition, the lifetime of the child object is simultaneous with the lifetime of the parent object, that is, when the parent is destroyed all children are destroyed too.

To specify that a class is an association, it should have the Association qualifier, for example, in our CoffeeMaker example, let us say the coffee maker has two pots, which are represented by IACMM_CoffeePot that cannot be used with other coffee makers. In MOF, we would specify this situation as:

```

[Association, Aggregation] class IACMM_CoffeeMakerPot
{
    [Description("The coffee maker"), Key, Aggregate] IACMM_CoffeeMaker ref Maker;
    [Description("The pot belonging to the coffee maker"), Key, Max(2)]
        IACMM_CoffeePot ref Pot;
}

```

Max(2) specifies the cardinality of the reference, aggregation specifies that the class is an aggregation, and the Aggregate qualifier specifies which role in the association is the parent.

Indications

CIM indications are instances of specialized CIM classes that are asynchronously broadcast to registered listeners. They describe an event that occurred. Indication instances do not persist for any amount of time in the CIM server, which is a fact that has numerous implications that include:

- ▶ Indications do not have key properties.
- ▶ Indications do not participate in associations.
- ▶ There are no methods, intrinsic or extrinsic, that can be invoked on an indication, for example, you cannot ask the CIM server to enumerate members of an indication class.

Indications fall into three broad categories:

- ▶ Life cycle indications are broadcast when instances of certain classes are created or deleted. They are instances of classes CIM_InstCreation or CIM_InstDeletion. As an

example, a client could register with a CIM server to be notified when instances of the CoffeeMaker class are created or destroyed.

- Modification indications are broadcast when certain properties of certain classes are updated. They are instances of class CIM_InstModification, for example, a client could register with a CIM server to be notified when the OperationalStatus property value changes for any CoffeeMaker instance. The indication identifies the instance that changes and contains the original and new property values.
- Alert indications are broadcast when some other kind of event occurs. They are instances of class CIM_AlertIndication, for example, a client could register with a CIM server to be notified if the coffee maker overheats.

Implementations are not required to support any type of indications. The implementation documentation must describe which, if any, indications are generated.

9.4 The CIM schema

In the previous sections, we described the CIM *meta-schema*, which is the description of the actual representation of the class in CIM. Now we can discuss the actual CIM schema.

The CIM schema is broken up into different sections, each representing a different segment of technology from user management to networking. However, at the base is the core schema, which contains general purpose base classes that most of the other schemas inherit from.

Figure 9-13 on page 196 illustrates the classes that make up the CIM_PhysicalElements and CIM_LogicalDevices part of the core CIM schema. The diagram is from the CIM schema published by the DMTF, with some simplification. As we can see from the key, red lines are associations, green lines with diamonds are aggregations, and green lines with diamonds and dots are compositions. Associations can also have numbers, ranges, or stars next to them, and they indicate the number of objects that can be at either end of the relationship, for example, the association between CIM_System and CIM_PhysicalElement, CIM_SystemPackaging has stars on either side, which means that the relationship between the two is many-to-many.

Page 4 : PhysicalElements & LogicalDevices (including StorageExtents)

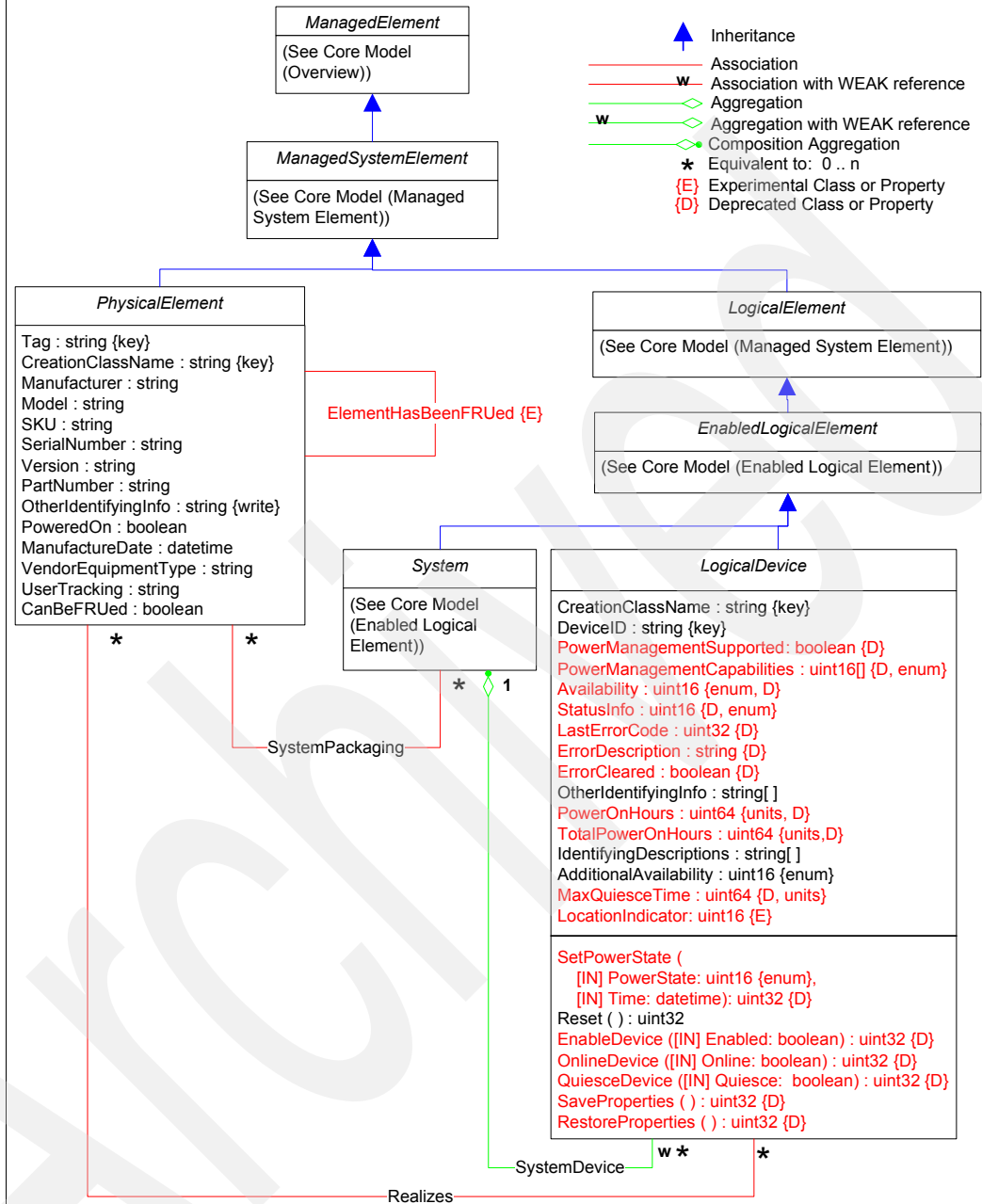


Figure 9-13 Core schema

In Figure 9-13, the text in red represents properties and classes that are either experimental or deprecated, depending on whether they are marked with D or E. Avoid deprecated properties and classes, and use experimental properties and classes with caution.

This part of the schema introduces several important classes that are subclassed heavily. CIM_PhysicalElement, for example, represents a physical piece of hardware. LogicalDevice, on the other hand, represents a more abstract device that can encompass several CIM_PhysicalElements and which might also belong to other CIM_LogicalDevices. This

relationship is represented by the CIM_Realizes association. CIM_LogicalDevices must also belong to a single System object, as shown by the CIM_SystemDevice aggregation, for example, a processor is an abstract concept that embodies the calculation engine(s) of a computer system. Because this is a logical construct, the CIM representation of a processor (class CIM_Processor) is defined in the Schema as a subclass of CIM_LogicalDevice. Physically, a processor is a chip, so a processor is most likely realized by an instance of class CIM_Chip, which is a component of a CIM_Card, which might be installed in a CIM_Slot in a CIM_Rack. All of these classes fall under the physical side of the schema.

The complete CIM schema is on the DMTF Web site at:

<http://dmtf.org>

9.4.1 Profiles

Profiles are specifications that define the properties of CIM classes. Profiles are essentially what allow interoperability between CIM implementations because they require that implementations of CIM classes behave in certain standard ways. On the DMTF Web site there are many different profiles that define the operations of CIM classes in different management domains. There are, for example, profiles for hardware elements, such as CPUs, fans, and software services.

Vendors can choose which profiles to support in their CIM implementations.

9.5 System z-specific schema

The implementation of CIM on the HMC supports a number of classes and extends a few for specific System z functionality. A complete guide to CIM on System z can be found by following the Library link on the HMC or in the IBM ResourceLink function. In this section, we discuss the different groups of classes that are used in the System z CIM implementation. We also cover, in detail, the model that is used to represent computer systems, including systems and images.

The broad groups of classes that are used on the HMC are:

- Computer system-related classes

In CIM, a computer system object is primarily the aggregation point for all of the things that make up a computer system. Some things that one might normally expect to be included in a computer system object are represented elsewhere, for example, the current trend in CIM class design is to dissociate the methods that operate on an object from the object itself. They are instead organized into interface-like Service objects, which we discuss in more detail later.

- Service-related classes

In CIM, a service class represents a set of one or more related methods (and sometimes properties) that perform a service to or on behalf of a related object. By dissociating the functionality from the object, you create the possibility of reusing the service definition in an interface-like manner for multiple classes of objects. An example of a service-related class is the CIM BootService class, which can start or stop an associated CPC or image using the StartService() and StopService() methods.

- Setting data-related classes

Setting data is the generic CIM concept that is used to describe collections of configuration-like data. They are used in the HMC model in various ways to describe Activation Profiles and temporary capacity records, for example, an IBMZ_ProcessorSettingData class controls various operational aspects of processors in pools, such as the pool ID and the weight.

- Logical device-related classes.

These classes represent logical devices, such as network ports and PUs.

There are several classes that are used to represent computer classes. Figure 9-14 illustrates the class hierarchy.

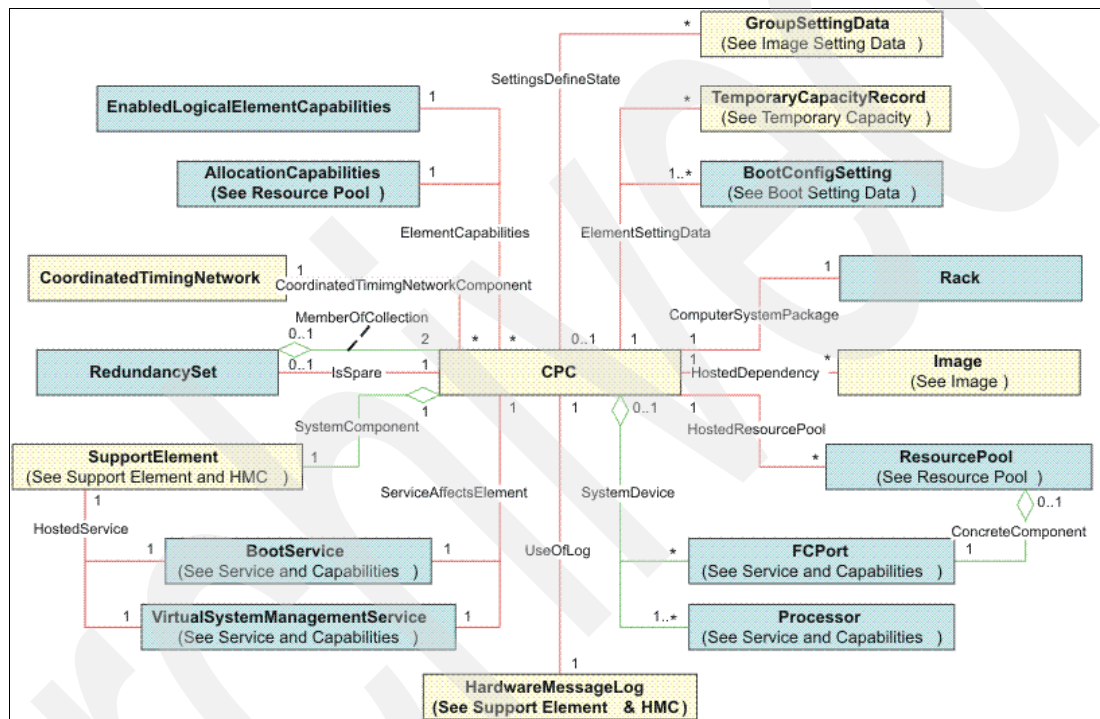


Figure 9-14 Computer system classes

In Figure 9-14, there are several classes that are specific to IBM in this hierarchy:

- Class IBMZ_CPC represents a CPC. In CIM nomenclature, a CPC IMled in LPAR mode performs the role of a host computer system. Class IBMZ_Image represents an image, either an LPAR or a Coupling Facility. It performs the CIM role of a virtual computer system.
- Class IBMZ_SupportElement represents a Support Element that is associated with a CPC. It performs the CIM role of a service processor. SE objects are included in the model because they more accurately portray the owner of a system setting data (Activation Profiles) and the provider of services. It is not intended that a CIM client can directly manage an SE.
- Class IBMZ_HardwareManagementConsole represents an HMC. In CIM terms, the HMC is a management client. Because CIM is focused on representing managed objects rather than clients, an HMC does not have a defined logical role. However, you can view the hardware on which the HMC application is running as a manageable object, so the HMC is represented at that level.

Figure 9-15 shows the relationship between CPC instances and other classes. The representations of associations are as in previous diagrams.

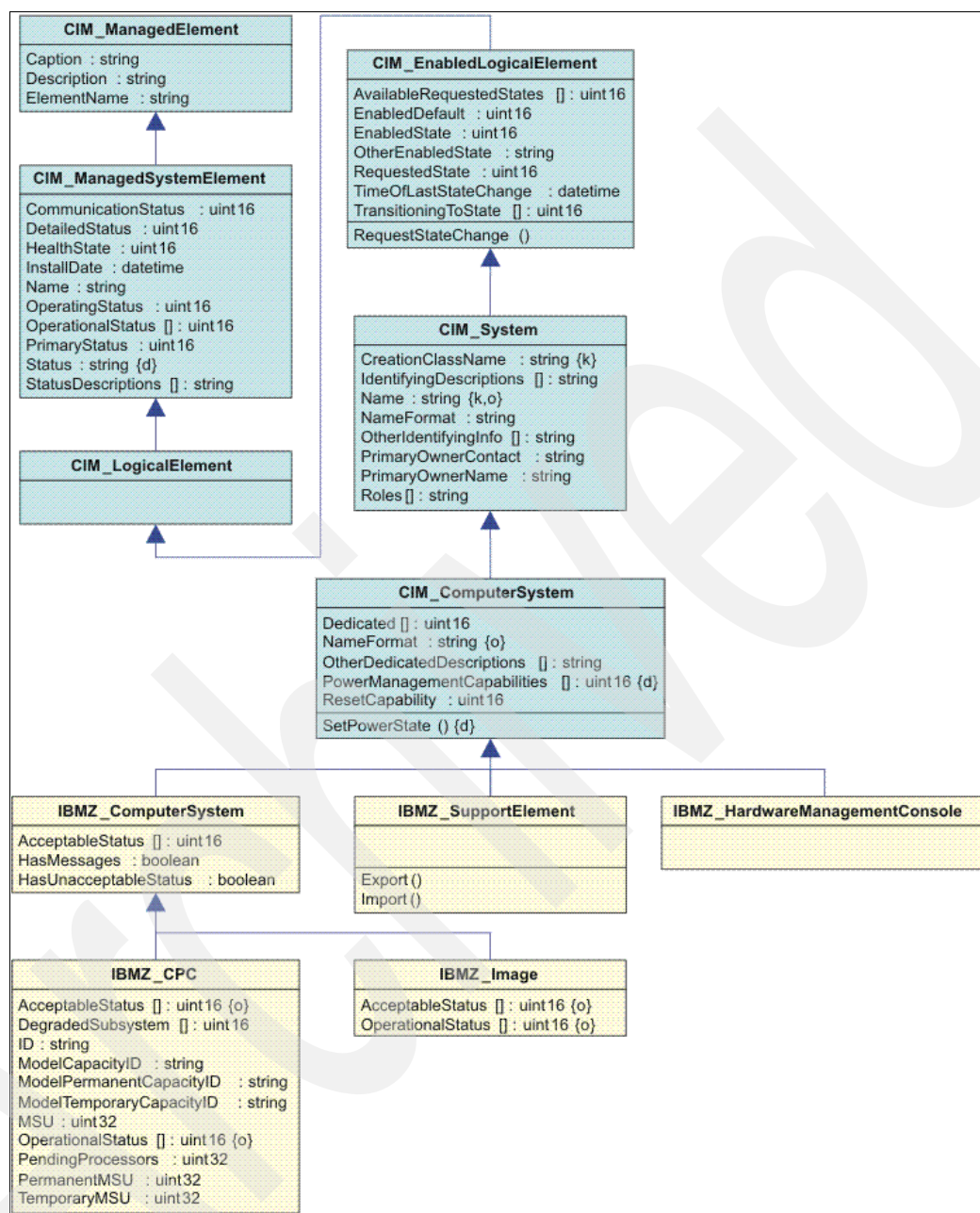


Figure 9-15 A cpc instance relationship

The physical enclosure is modeled using an instance of CIM_Rack, which is associated using CIM_ComputerSystemPackage to its CPC. The enablement capabilities of a CPC are described by an instance of CIM_EnabledLogicalElementCapabilities that is associated to each CPC using CIM_ElementCapabilities.

The Fibre Channel port allocation capabilities of a CPC are described by an instance of CIM_AllocationCapabilities that is associated to each CPC through CIM_ElementCapabilities.

One or more boot configurations (Reset Activation Profiles) are rooted in CIM_BootConfigSetting instances, which are associated to the CPC through CIM_ElementSettingData.

One of the boot configurations that are associated with a CPC will have the CIM_ElementSettingData.IsNext property set to 1. This setting indicates the configuration that will be automatically used when the CPC is next enabled (powered on). One of the configurations will have the CIM_ElementSettingData.IsDefault property set to 1, which identifies the system-defined default boot configuration. If the CPC was booted at least once, one of the associated configurations will have the CIM_ElementSettingData.IsCurrent property set to 1, indicating the last used boot configuration. A single CIM_ElementSettingData instance might have more than one of these attributes present, which indicates that the referenced boot configuration plays multiple roles.

Zero or more virtual computer system (image) instances are associated through CIM_HostedDependency to the hosting CPC. Zero or more real-time Image group settings are associated through CIM_SettingsDefineState to the hosting CPC.

A CPC contains zero or more host CIM_FcPort instances (Fibre Channel PCHIDs) as indicated by CIM_SystemDevice associations. Each FC port is aggregated into a CIM_ResourcePool, as indicated by a CIM_ConcreteComponent association. The resource pool is hosted by a CPC, as indicated by the CIM_HostedResourcePool association.

A CPC contains one or more CIM_Processor instances, (PUs) as indicated by CIM_SystemDevice associations. Zero or more temporary capacity records are associated through CIM_ElementSettingData to a CPC.

The members of an IBMZ_CoordinatedTimingNetwork are identified by IBMZ_CoordinatedTimingNetworkComponent associations to instances of IBMZ_CPC. If a backup time server is configured, two of these members are also associated to an instance of CIM_RedundancySet through CIM_MemberOfCollection. One of these will additionally be associated to the CIM_RedundancySet through CIM_IsSpare, identifying the spare time server. The current time server is the other member of the redundancy set.

The CPC's hardware message log is represented by an instance of IBMZ_HardwareMessageLog that is associated to the CPC through CIM_UseOfLog. A CPC is related to its IBMZ_SupportElement through a CIM_SystemComponent association.

A CPC is affected by two services: CIM_BootService and CIM_VirtualSystemManagementService, each associated to the CPC through CIM_ServiceAffectsElement. A Support Element hosts the CIM_BootService and CIM_VirtualSystemManagementService instances, as indicated by the CIM_HostedService associations.

CIM infrastructure

The mechanisms for performing operations on CIM objects are part of the WBEM set of technologies specified by the DMTF. CIM clients' communicate with CIM servers over a TCP/IP network connection, as suggested in Figure 9-16, using HTTP to make requests.

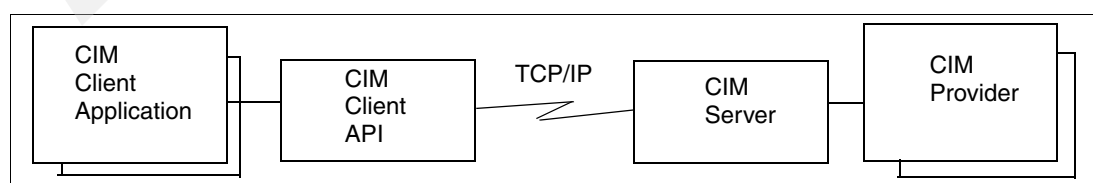


Figure 9-16 CIM infrastructure

Typically, CIM client Application Developers use one of the CIM client API packages that are generally available. The OpenPegasus CIM server includes C and C++ client APIs:

<http://www.openpegasus.org>

A Java CIM client is available at:

<http://sblim.wiki.sourceforge.net>

A search of the Internet will yield other CIM client API packages. On the server side, vendor Developers write plug-in provider modules that process incoming CIM requests and return results to the CIM client. Asynchronous event notifications (indications) can also flow from the CIM server to registered CIM clients.

9.5.1 The Java CIM client API

Client software that accesses CIM through the HMC can be written using the Java CIM client of SBLIM (pronounced “sublime”), which is a collection of system management software that implements WBEM. You can download version 2 of the SBLIM Java client from:

<http://sblim.wiki.sourceforge.net/CIMClient>

Example 9-3 is a source code listing for a simple program that connects to the HMC and prints to the console all CIM keys and their values for all instances of IBMZ_Image.

Example 9-3 Source code listing for a program

```
import java.util.Locale;
import javax.cim.CIMInstance;
import javax.cim.CIMObjectPath;
import javax.cim.CIMProperty;
import javax.security.auth.Subject;
import javax.webm.CloseableIterator;
import javax.webm.client.PasswordCredential;
import javax.webm.client.UserPrincipal;
import javax.webm.client.WBEMClient;
import javax.webm.client.WBEMClientFactory;

public class CIMTest {
    public static String hostName = "9.60.15.40";
    public static String protocol = "https";
    public static String port = "5989";
    public static String user = "sysprog";
    public static String password = "password";
    public static String namespace = "root/ibmz";

    public static void main(String[] args) {
        WBEMClient client = WBEMClientFactory.getClient("CIM-XML");
        CIMObjectPath path = new CIMObjectPath(protocol, hostName, port, null, null, null);
        Subject subject = new Subject();
        UserPrincipal principal = new UserPrincipal(user);
        subject.getPrincipals().add(principal);
        PasswordCredential credentials = new PasswordCredential(password);
        subject.getPrivateCredentials().add(credentials);
        // Connect to the CIM server.
        try {
            client.initialize(path, subject, Locale.getAvailableLocales());
            // Construct a CIMObjectPath for the class IBMZ_Image.
            CIMObjectPath imagespath = new CIMObjectPath("IBMZ_Image", namespace);
            // Iterate over the instances of this class.
```

```

        CloseableIterator iter = client.enumerateInstances(
            imagesPath, false, false, false, null).
        while (iter.hasNext()) {
            CIMInstance inst = CIMInstance(iter.next());
            CIMProperty[] keys = inst.getKeys();
            for (int i=0; i < keys.length; i++ {
                System.out.println(
                    "Key name" + keys[i].getName() + ", "
                    + "Key value: " + keys[i].getValue() + "\n");
            }
            System.out.println();
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.exit(-1);
    }
}

```

As you can see in Example 9-3 on page 201, `javax.wbem.WBEMClient` is the central class that is used in the SBLIM library. It sets up the connection to the HMC, and it is used to make CIM requests. However, `WBEMClient` is just an interface, and it uses the factory pattern for instantiation. Thus, to instantiate it, use the `getClient()` method of the `javax.wbem.client.WBEMClientFactory` class.

There are several important methods of `WBEMClient` that we explain in detail later. All of these methods throw `WBEMException`. Several of these methods return `javax.wbem.CloseableIterator`, which is a subinterface of `Iterator` with the additional `close()` method, which, as expected, closes the connection that the iterator uses.

Note the difference between methods with and without the `Names` suffix. Methods that end in `Names` return iterators that iterate over CIM object paths, whereas the methods that do not end in `Names` iterate over actual objects.

The following method either enumerates through the CIM instances that are associated to a given CIM instance or through the classes that are associated to a given class.

```

CloseableIterator associatorNames(CIMObjectPath pObjectName, String
    pAssociationClass, String pResultClass, String pRole, String pResultRole)

```

If `pAssociationClass` is null, all associated classes or instances are returned; otherwise, only the ones that are associated with the given association name are returned. Additional characteristics of this method are:

- ▶ `pObjectName` is the CIM path to the instance or class.
- ▶ `pResultClass` must be either null or a string with a CIM class name, and it filters the results to objects of the given class.
- ▶ `pRole` filters the results to associations where the source object has a given role that is specified as a string, for example, the `CIM_SystemDevice` association has two roles, `GroupComponent` and `PartComponent`. To only retrieve associated objects where the source is that object's `GroupComponent`, you pass `GroupComponent` for `pRole`. For no filtering, pass a null value.
- ▶ `pResultRole` is the same as `pRole` but for the target object.

The following method is similar to `associatorNames`, but it returns a `CloseableIterator` with full CIM instances, represented by the `CIMInstance` class or by CIM classes that are represented by `CIMClass`, if the source path is a class:

```
CloseableIterator associators(CIMObjectPath pObjectName, String
    pAssociatorClass, String pResultClass, String pRole, String pResultRole,
    boolean pIncludeQualifiers, boolean pIncludeClassOrigin,
    String[] pPropertyList)
```

Additional parameters for this method are:

- ▶ `pIncludeQualifiers`, which if true includes qualifiers in the returned objects.
- ▶ `pIncludeClassOrigin`, which is true, includes the `CLASSORIGIN` attribute on relevant elements in returned objects. This attribute indicates the class in which the element was defined.
- ▶ `pPropertyList`, which is a list of properties that specifies which properties are included on the returned objects. If this parameter is null, all properties are included. If it is an empty array, no properties are included.

The following method enumerates the names of instances for a given CIM class:

```
CloseableIterator enumerateInstanceNames(CIMObjectPath pPathP)
```

The following method enumerates `CIMInstance` objects for a given CIM class:

```
CloseableIterator enumerateInstances(CIMObjectPath pPath, boolean dDeeep,
    boolean pPropagated, boolean pIncludeClassOrigin, String[] pPropertyList)
```

The additional parameters are:

- ▶ `dDeeep`, which if true includes instances of subclasses for the class as well.
- ▶ `pPropagated`, which if true only includes properties, methods, and references that are defined or overridden in this class.
- ▶ `pIncludeClassOrigin`, which is the same as in the `associators` method that we described earlier.
- ▶ `pPropertyList`, which is the same as in the `associators` method that we described earlier.

The following method retrieves a single `CIMInstance` with the object path `pName`. Its arguments are the same as the previous method:

```
CIMInstance getInstance(CIMObjectPath pName, boolean pPropagated,
    boolean pIncludeClassOrigin, String[] pPropertyList)
```

The following method invokes the method given in `pMethodName` on the CIM instance given in `pName`:

```
Object invokeMethod(CIMObjectPath pName, String pMethodName,
    CIMArgument[] pInputArguments, CIMArgument[] pOutputArguments)
```

The parameters are:

- ▶ `pInputArguments` is an array of `CIMArguments` that are the arguments to the target method.
- ▶ `pOutputArguments` is an array of `CIMArguments` where the output is stored. The members of this array must not be initialized.

The following method modifies the an instance, denoted as `ci` in this reference:

```
void modifyInstance(CIMInstance ci, String[] propertyList)
```

The target instance, denoted as ci, must have properties set, and the properties that are to be changed are passed as an array of Strings in propertyList.

The following method returns a CloseableIterator of CIMObjectPaths of the associations that refer to the object that is specified by pObjectName:

```
CloseableIterator referenceNames(CIMObjectPath pObjectName,  
    String pResultClass, String pRole)
```

If pObjectName is a class, the iterator will be of association classes that refer to that class. pResultClass and pRole filter the results similarly to the associatorNames method.

The following method is similar to the last one, but returns a CloseableIterator of association CIMObjects:

```
CloseableIterator references(CIMObjectPath pObjectName, String pResultClass,  
    String pRole, boolean pIncludeQualifiers, boolean pIncludeClassOrigin,  
    String[] pPropertyList)
```

If the source path is for a class, it returns association CIMClasses. The remaining arguments are similar to those of the associators method.

The difference between the associator[Names] and reference[Names] methods is that the former directly returns objects that are associated to a given object, whereas the latter returns the actual association objects that have the given object fulfilling a role.

These are less commonly used methods that are not explained in detail. For a full reference, see the SBLIM Java API documentation at:

<http://sblim.sourceforge.net/cim-client2-doc/>

- ▶ void close()
- ▶ void createClass(CIMClass pClass)
- ▶ CIMObjectPath_createInstance(CIMInstance pInstance)
- ▶ void deleteClass(CIMObjectPath pPath)
- ▶ void deleteInstance(CIMObjectPath pPath)
- ▶ void deleteQualifierType(CIMObjectPath pPath)
- ▶ CloseableIterator enumerateClasses(CIMObjectPath pPath, boolean pDeep, boolean pPropagated, boolean pIncludeQualifiers, boolean pIncludeClassOrigin)
- ▶ CloseableIterator enumerateClassNames(CIMObjectPath pPath, boolean pDeep)
- ▶ CloseableIterator enumerateQualifierTypes(CIMObjectPath pPath)
- ▶ CloseableIterator execQuery(CIMObjectPath pPath, String pQuery, String pQueryLanguage)
- ▶ CIMClass getClass(CIMObjectPath pName, boolean pPropagated, boolean pIncludeQualifiers, boolean pIncludeClassOrigin, String[] pPropertyList)
- ▶ String getProperty(String pKey)
- ▶ CIMQualifierType getQualifierType(CIMObjectPath pName)
- ▶ void initialize(CIMObjectPath pName, Subject pSubject, Locale[] pLocales)
- ▶ void modifyClass(CIMClass pClass)
- ▶ void setLocales(Locale[] pLocales)

- ▶ `void setProperty(String pKey, String pValue)`
- ▶ `void setQualifierType(CIMQualifierType, pQualifierType)`

9.6 Questions and discussions

1. The SNMP design for reporting seems primitive and confusing. Why do you think this came about?
2. Java seems to mix well with SNMP. Were they designed for this?
3. Why are automation and SNMP grouped together?
4. How much of this material does the average System z Systems Programmer see? An Advanced Systems Programmer?

Archived

Problem analysis

In this chapter, we discuss aspects of problem analysis:

- ▶ Overview of the issues
- ▶ Reliability, availability, and serviceability
- ▶ Autonomic Computing
- ▶ Problem Analysis
- ▶ First Failure Data Capture (FFDC)
- ▶ Trace
- ▶ Active Resource Monitoring

After reading this chapter you should have a better understanding of the highly-structured problem analysis and handling that is involved in maintaining System z firmware and hardware.

10.1 Overview of problem analysis

There is always the chance that entities within a system will breakdown. Throughout the history of computing, measures were always taken to prevent problems from arising, but it is not possible to prevent every problem. A more feasible goal is to gain the capability to fix a problem as quickly as possible and not allow the problem to interfere with the operations that your system is running.

Problem analysis within the Hardware Management Console (HMC) focuses on taking problem solving approaches to a higher level. Within the HMC, an attempt is made to facilitate the involvement of the user and keep a Knowledge Database so that when a problem arises it is handled quickly and efficiently.

In this chapter, we introduce the concepts of reliability, availability, and serviceability (RAS) in System z terminology. We then expand on the serviceability aspect. After this chapter you will be able to:

- ▶ Understand the importance of RAS to the System z family of processors.
- ▶ Understand the major components of serviceability (the S in RAS) that reside on the Hardware Management Console and Support Element (SE).
- ▶ Understand the life cycle of an error that occurs in the mainframe.

10.2 RAS

One of the key features of the System z family of processors is their reliability. In fact, the 'z' in System z stands for *zero down time*. The three components of RAS (reliability, availability, and serviceability) work together to achieve this goal.

10.2.1 Reliability

The hardware components and coding put into the IBM mainframe are made for long term reliable usage. This keeps errors at a minimum and allows for the declaration that the mainframe could run with zero downtime and without data corruption.

To ensure that these promises are kept much work is put into making sure that the mainframe is built with sturdy hardware and that the code that runs the mainframe is thoroughly tested and of the highest quality. The IBM policy is to only release a product to market if the quality of this product is superior to the current generation's product quality. This policy ensures that the IBM products have continuous quality improvement.

Although much effort goes into preventing errors from occurring in the structure of the mainframe, expecting errors not to occur is unrealistic. Built into the reliability model is a set of steps that facilitate minimizing the impact an error makes on a client.

When errors are detected the system attempts to solve the problem through a series of tests:

1. First we check if a permanent malfunction in the code or hardware caused the problem.
2. Then we ascertain if the failure was intermittent by retrying the task.
3. If the task works after the retry, no impact is made to the system, and it continues to run; however, if we find that there is actually a malfunction in the hardware or the code, we move to fix the error.

10.2.2 Availability

IBM mainframes are built with more components, such as processors and memory, than the client requests when purchasing the system. We cover this concept in detail in Chapter 14, “Capacity on demand” on page 293. Also, redundant hardware is included in key areas, such as power components, error correcting bits in memory, and RAID DASD. Because of the existence of extra hardware in the system, we can isolate faulty parts and repair them without impairing the system’s ability to run, which is known as *Fencing*, and it further supports the mainframe’s goal of zero downtime.

Sometimes Fencing does not solve the problem. From this stage in the error cycle, many possibilities exist that extend into the serviceability component of the RAS model. One feature is that much of the hardware is designed so that it can be safely removed and installed into a running system without requiring the system to be powered down. In System z terms, we call this *Concurrent Repair*.

10.2.3 Serviceability

A considerable amount of design thought is needed to ensure that a system can be serviced in a reasonable way and that most service actions can occur without taking the system down completely. Furthermore, considering how seldom these systems fail, training for service actions that are seldom performed on “real” systems is a major part of this effort. Here are a few goals:

- ▶ All errors must be detected.
- ▶ The error must be contained so that it does not affect the customer’s data.
- ▶ The error must be isolated to the failing component (part of this is done by the problem analysis component).
- ▶ The error must be repaired with the least amount of impact to the customer’s operation as possible.

Basically, the reliability of the system is so good that many systems never experience any problems that require servicing. Because of this, the serviceability package is designed to remove most of the decisions out of the hands of the user or service personnel and instead use automation as much as possible.

10.3 Autonomic computing

Problem analysis within the HMC takes an approach that tries to achieve a high level of autonomic computing. To fully understand this approach you must understand what autonomic computing is and how it plays a role in problem analysis. IBM actually coined the term autonomic computing to describe its initiatives to reduce the complexity of managing the ever growing complexity of computer systems. Problem analysis goes a long way towards reducing the complexity (from a user’s perspective) of managing errors that occur on a system.

Autonomic computing creates systems that can manage themselves without needing Information Technology (IT) professionals to monitor and maintain the systems. The name derives from the autonomic function of the human central nervous system. Autonomic controls use motor neurons to send indirect messages to organs at a sub-conscious level. These messages regulate temperature, breathing, and heart rate without conscious thought. The implications for computing are immediately evident: a network of organized, smart

computing components that provide what is needed, when its needed, without a conscious mental or even physical effort.

This new paradigm shifts the fundamental definition of the technology age from computing to defining by data. Access to data from multiple, distributed sources in addition to traditional centralized storage devices allows you to transparently access information when and where you need it. At the same time, this new view of computing necessitates changing the industry's focus on processing speed and storage to developing distributed networks that are largely self-managing and self-diagnostic.

This new computer paradigm means that the design and implementation of computer systems—software, storage, and support—must exhibit these basic fundamentals from a user perspective:

- ▶ Flexible: The system can sift data using a platform and device-agnostic approach.
- ▶ Accessible: The nature of the autonomic system is that it is always on.
- ▶ Transparent: The system performs its tasks and adapts to your needs without dragging you into the intricacies of its workings.

10.4 Problem analysis from start to finish

From a System Administrator's perspective, the internals of how problem analysis works is not much of a concern; instead, they are mostly concerned with the results. We discuss these results later on in 10.5.4, "Notification" on page 219, but basically, the customer is notified about four things:

- ▶ What happened
- ▶ What was the impact of the problem
- ▶ What the customer should do about the problem
- ▶ What is the impact the repair will have

So now we have an understanding of why we have problem analysis and what problem analysis tries to achieve. Now we can walk through how problem analysis works. It is important when studying problem analysis to keep in mind the principles of autonomic computing that we learned earlier in 10.3, "Autonomic computing" on page 209.

10.4.1 Problem analysis framework

The major objective of the problem analysis framework is to provide a way that events are captured, analyzed, and reported. As events progress through the system, there exists transitional states that the events go through on the way to becoming problems. A typical process is to:

1. Capture the event.
2. Analyze the event.
3. Wait for additional events to be captured and analyzed.
4. Analyze the collection of events as a whole.
5. Determine the primary cause of the failure.
6. Report the results to one or more interested parties.

The problem analysis framework provides three classes to represent the failure as it progresses through problem analysis:

- ▶ The first is a problem analysis event, which represents a situation that was detected that might require additional analysis to be performed.

- The second class is an incident, which represents failures that were analyzed but could be waiting for some other event to occur.
- The third class is a Problem. This class represents failures that are service action points, that is, something on the system that must be repaired.

To move the failure objects through the framework, control and analysis classes are needed. Figure 10-1 shows the flow of failure objects through the framework and the interaction between the components.

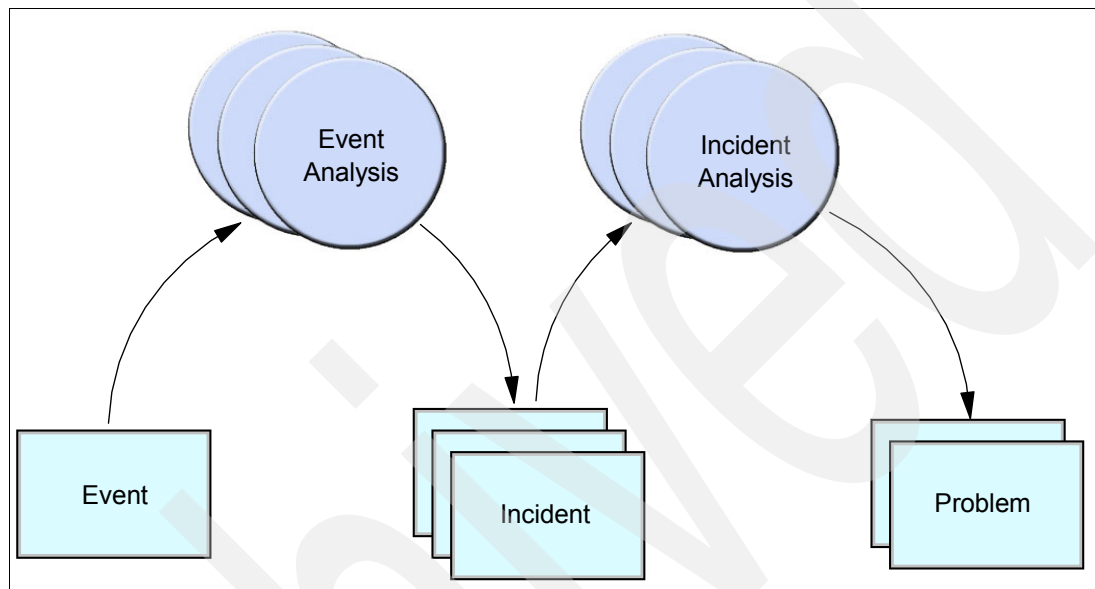


Figure 10-1 Problem creation

Events are posted to the problem analysis framework by the *event generators*. Event generators are code in the various subsystems that are responsible for detecting potential problems in the system and then gathering and creating the input to the analysis system, for example, there are event generators for the power subsystem, channel subsystem, processor subsystem, and so on. Analysis routines (AR) for each system are registered with the event manager to listen for events that originate with that subsystem:

1. When the event manager receives the event, the registered analysis routine is called.
2. The analysis routine then does what is appropriate for that event, for example, suppose a CPU performed an arithmetic calculation incorrectly:
 - a. The problem is recognized by the hardware and an event is created.
 - b. The event is then analyzed by the analysis routine that is responsible for CPU failures.
 - c. The analysis routine determines the information that is necessary for notifying the customer, calling home the problem (if appropriate), and information needed to correlate the incident with other incidents.
3. An incident is then created containing the results of the analysis.

There are situations when a subsystem might retry a long running function if any failures are detected. If the retry is initiated, the pending failure events must be discarded, but if no retry is attempted, the analysis should continue in the normal manner. To accommodate this retry a problem *analysis suspend* function is provided, which triggers the event manager to hold (suspend) posting of all events for the duration of the suspend function.

Two types of ending events are provided:

- ▶ The first is a resume: Any events that were held are posted for processing.
- ▶ The second is a clear: This event clears all pending events and allows the posting of any new events which might be received.

A timer is associated with the suspend function to provide restarting of the event posting in the absence of an ending event.

As an event generator detects some condition that it wants analyzed, it flows through the problem analysis framework, as shown in the following process, which we also illustrate in Figure 10-2 on page 213:

1. It creates a problem analysis event and posts it to the problem analysis event manager.
2. The event manager then calls all of the registered listeners for this type of event, which is the primary analysis routines for the event.
3. The AR then creates an incident, sets the appropriate time delay for the incident, and adds the incident to the incident pool.
4. After the time expires, the event manager is notified.
5. The event manager then generates a second problem analysis event that calls a different analysis routine. This analysis routine looks at ALL of the Incidents in the incident pool and determines that it must create a *problem*.
6. The AR determines whether to open a new problem or not depending on whether a duplicate problem is already opened.
7. The problem manager posts an open event until the Problem Analysis framework is finished. All listeners now must do something with the open problem, and that particular action is outside of the scope of the problem analysis framework. Examples of listeners include: customer notify, call home, Hardware message, pager notification, R/V, and so forth.

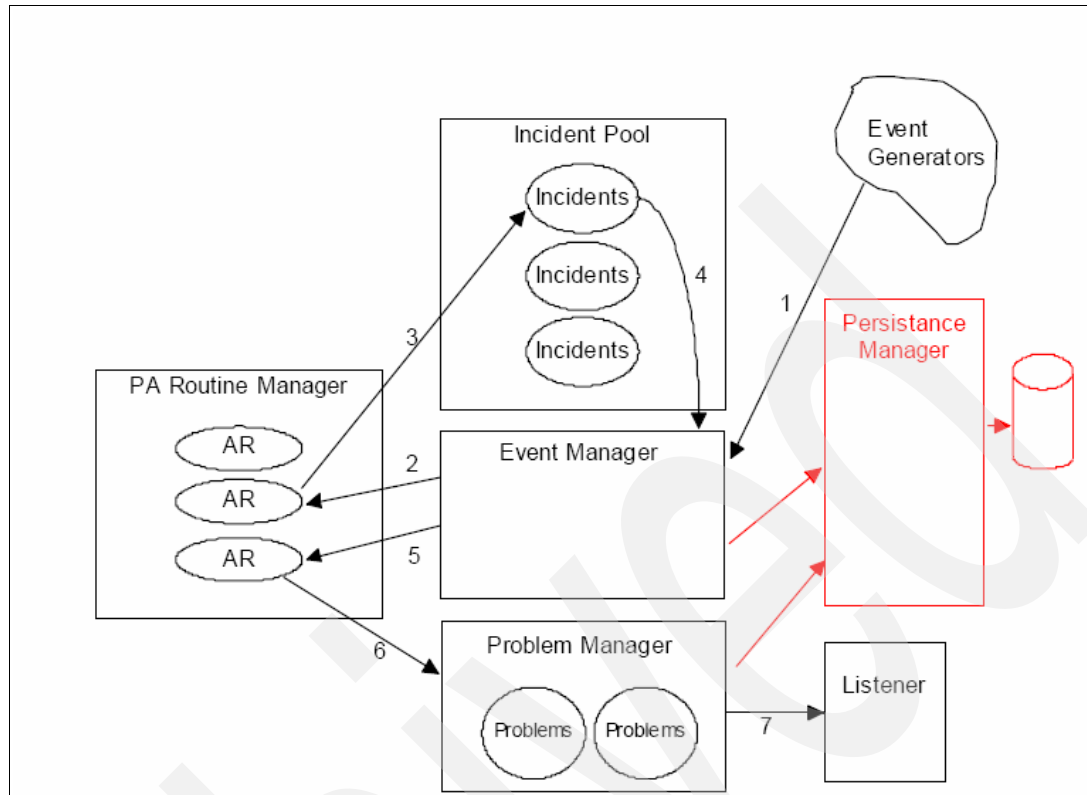


Figure 10-2 Problem analysis framework flow diagram of a hardware event

10.4.2 Analysis phase

After an error is recorded and problem analysis is notified. The initial analysis routine (AR) is run (see arrows 1 and 2 in Figure 10-2). The type of error determines which analysis routine to run.

When the initial analysis routine is run, an incident is created and added to an incident pool (see arrow 3). The purpose of the pool is to hold all of the incidents that occur during the PA analysis time window. The time window is to ensure that all subsystems that are downstream from the event have adequate time to detect and report the event, for example, suppose a power spike in an I/O cage occurs. The power subsystem and the I/O cage both detect and report this incident. In addition, the CEC might also detect that the I/O card timed out and report another incident. The purpose of the time pool is to collect all of these incidents so that they are grouped together to open a single problem.

There are several subsystems that are involved with the CPC. Ideally, for every problem that occurs, only one incident is reported from a single subsystem. However, this is not always the case, for example, when a channel between a CPC and an I/O cage fails, both the CPC and the I/O cage see the problem and therefore both report the incident. Problems such as this cannot rely on a single subsystem to report the incident because a power failure might have occurred on either the CPC or the I/O cage that prevents that subsystem from reporting the incident. The analysis phase is necessary to determine the most accurate source of the problem for such occurrences when there are multiple reports.

The base line path is that the analysis routine sets a timer that closes the window at the end of the time. When the timeout occurs, the pool is closed, so no new incidents can be added to it. All other incidents that are in the pool will at this time have their timers canceled, and all of

the incidents that are in the pool are then grouped together and processed as a group. After the timeout occurs, any second level subsystem analysis is performed. This action allows each subsystem to alter the collection of incidents that are in the pool, for example, it can combine two incidents into one new incident.

In addition to setting the timeout values, the initial AR assigns a priority to each incident. Each subsystem is assigned a value in the range of priorities that are used by the system-level AR to determine the proper problem to open. There is a general mapping, but each subsystem has the ability to alter the value.

After all of the secondary ARs are complete, a system-level AR is run. This routine determines what incidents to leave in the pool based on the system perspective. For the SE, the algorithm is to sort the incidents in the pool based on the priority that is assigned to it. The default behavior is that if two incidents have the same priority, the incident that occurred first is higher priority.

There are exceptions to the standard behavior of the analysis phase. Although these exceptions are fairly uncommon, they are required to determine the most accurate cause of failure.

One way that the base analysis flow can be altered is that the initial AR can specify that the incident requires that the full time specified for the incident must be waited for. This specification influences the flow when the incident is not the initial incident in the pool, for example, if the first incident is a standard incident with a timeout value of 45 seconds associated with it and a second incident is added to the pool that has a required wait of 60 seconds, the pool does not close when the first timer expires but when 60 seconds elapses since the addition of the second timer. This behavior is referred to as “full timer wait” within problem analysis. To prevent an endless wait situation, only the first full timer wait incident is waited on. Any additional full timer wait incidents are canceled when the first full timer wait times out. Currently, the only use of this flow is when timeouts are supplied in the logs.

For each subsystem, there are predefined default timeout values that are based upon the average or most common problem for the subsystem. In special cases, a much larger timeout is required for a particular problem. These specific problems can specify an alternative timeout value. The new value is not treated any differently than a normal incident, for example, if an incident is a normal incident and has a timeout value of 45 seconds associated with it and a second incident is added that has a specified timeout of 60 seconds from the default of 30 seconds, the incidents still stop waiting at 45 seconds from receiving the first incident.

Another exception to the default behavior of the Analysis phase is done by identifying the incident as one that must always be reported regardless of the other incidents. These incidents are always made into a separate problem, and the analysis phase continues to create a problem with the highest priority incident, for example, capacity on demand allows a customer to purchase the use of additional processors when needed during peak work-loads, such as tax preparers around April 15th. When the purchased time-frame expires, these additional processors are deactivated. An incident is reported prior to this time to alert the customer of the deactivation and is reported independently of any other incident. A capacity on demand (CoD) license expiring is an incident that is always reported, regardless of any other incident (see left-side of Figure 10-3 on page 216).

Similarly, an incident can be marked to report all incidents of the same priority (see the Flag field in Figure 10-3 on page 216). Load failures (failures that occurs during IPL), that are the highest priority in the incident pool, are marked to report all other load failures of the same priority that are in the pool. Load incidents are considered independent of one another and therefore system analysis generates a separate problem for each Load failure, for example, if

a number of load failures are detected along with a channel path failure. The most likely cause for the load failures is the channel path failure. A channel problem is opened. However, if only load failures are detected, system analysis knows that there is no probable link between multiple Load failures. Therefore, all of the load failures are independently reported, and a separate problem is opened (see right-side of Figure 10-3 on page 216).

10.4.3 Analysis routines

In this section, we discuss two types of analysis routines.

First-level analysis routine

The design pattern that was chosen for the HMC was to have a multiple-level analysis process. First-level analysis routines look at one error from a specific subsystem at a time. These routines determine if there is a condition that indicates a problem or not. If there is a problem, the first-level analysis routine creates a problem and passes that, with the analysis control information, in an incident to the next level of analysis. This process is repeated for the subsystem and then for the system. The end result is some number of problems being opened with the Problem Manager (see arrow 6 in Figure 10-2 on page 213).

The analysis routine also must determine what is to be reported to the customer. Any number of problem statements can be added to the problem to inform the customer of the problem description, what corrective actions can be performed and what the impact of repairing this problem is. Also the priority of the incident is set so that proper system-level analysis can be supported.

The timeout value is read out of the configuration file that is being held by the Event Manager and set into the incident. It then adds the incident, which contains the problem, to the incident pool that the identifies system. At this point, the first-level analysis routine is complete.

All the timers are canceled for the remaining incidents in the incident pool so that the spurious timeout events are not posted for this pool. Any secondary events that the first level analysis routine associated the incident are then posted with the Event Manager. After all of the secondary events are posted, a System Analysis Ready Event is posted to the event manager. The event has the original incident pool (the “analysis” incident pool) attached to it.

System analysis routine

The final analysis routine for the HMC is the system analysis routine (this is step 5 in Figure 10-2 on page 213). This routine determines which incidents become problems. For the HMC, the process is straight forward because the incidents and problems are set up by the preceding analysis routines.

The first step of this routine is to sort the incidents. The default implementation that is provided with the Common Service Incident is to sort based on the priority of the incident followed by the time of the incident, if the priorities are equal.

The second step performed allows the highest priority incident a chance to swap places with any of the other incidents in the pool. This swapping allows special case incidents to be instantiated by the analysis routines, which can alter the sort algorithm without having any special case code in the system analysis routine. If any call to swap results in a true, the passed incident becomes the highest priority incident and the processing of other incidents in the pool stops.

The third step is to pass the list of incidents to the top priority incident and ask if any of the other incidents should also have problems opened for them. One use of this with the System

z machines is that if the top priority incident is an IPL failure, we want to also report any other IPL failures that are in the analysis pool.

The top priority incident and any incidents deemed as associates will have problems opened for them. At this point the system analysis routine is complete. Figure 10-3 shows an overview of two examples of flows through the analysis routines.

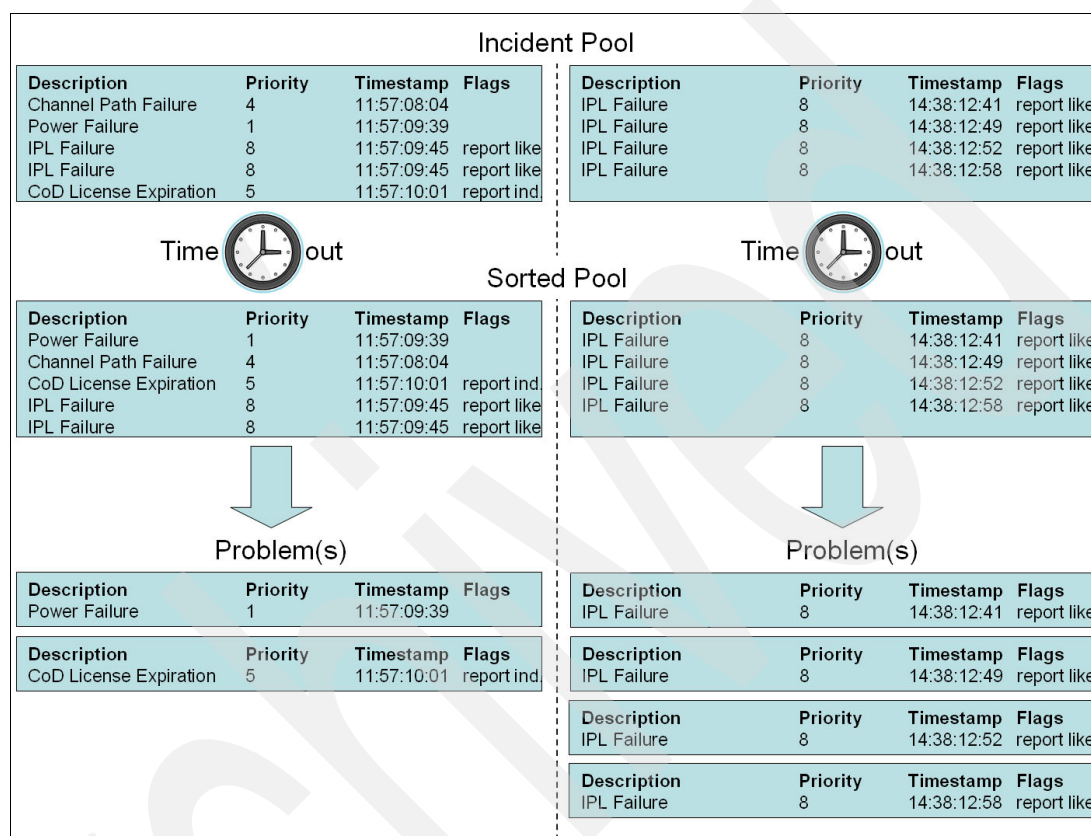


Figure 10-3 Analysis routines

10.5 First failure data capture

In an ideal world, code is developed in such a way that it is 100% reliable and does exactly what it is supposed to do under all conditions. Unfortunately, the systems today are so complex and have so many layers of code running on them that unexpected conditions sometimes cause unexpected results. Because the System z systems are designed for 99.999% of availability, the code is designed to handle the unexpected. One method to mitigate the impact of unexpected code behavior is the concept of First Failure Data Capture (FFDC).

The HMC's implementation is an important concept that involves the following elements:

- ▶ Collection of the right data at the right time
- ▶ Predetermined set of data are collected for all problems:
 - Log file
 - Machine configuration data
 - System activity information
 - Platform information

- Problem-specific data collection:
 - Additional files
 - Program can be run to do collection
 - Can be modified by the IBM product engineering group (PE)

The first failure data capture concept is that at the occurrence of any failure in the system, FFDC gathers and saves enough data for later analysis to assist in analyzing the source of the problem. A good FFDC implementation, in most cases, delivers a sufficient set of data for Developers to work with and solve the problem. It is important to get the right set of data when the problem occurs because we cannot expect the customer to allow the problem to be re-created for the purpose of additional data collection. FFDC is also used in development and testing. FFDC gives Developers a snapshot of the data that is necessary to debug any problem that is found during testing. This snapshot gives the FFDC a cost benefit by allowing Developers to focus on the problem that was discovered rather than requiring the test organization to recreate the problem.

10.5.1 Data collection

While data collection is very important, error detection and error reporting are more important. Steps are taken to ensure that the data collection does not interfere with the error detection and subsequent error reporting processes, for instance, there is a time-out of one hour for the data collection process, and if all the data collection code does not finish in one hour, the data is not collected. Even with this time limit, when there are many errors occurring over a short period of time the FFDC collection process can get behind because each collection takes time. Steps are being taken to address this and determine when to stop FFDC collection.

There is also the possibility that something is incorrect with the data collection process itself, so problems collecting FFDC data (missing data) are ignored because the original problem may be causing the data collection problem. By ignoring problems collecting FFDC data, we prevent infinite loops from occurring.

Because the systems are complex and consist of many subsystems and components, a hardware or firmware failure often results in a flood of information being recorded on the SE/HMC. This data consists of log data and subsystem or component-specific private data, which is usually only of interest to the subsystem or component developer. If all the recorded data was collected for every failure, the call home data set would take many hours to transmit to IBM. Since this is not desirable, a means to select what is transmitted to IBM is needed. The idea is not to collect everything possible, but to collect only that data necessary for analysis of the particular problem. The HMC collects more data than necessary to prevent the condition of not having the right data.

The component that does this data collection is the FFDC collection code. This code works with the problem analysis code to determine what data is to be collected, where it is to be saved, and what is called home to IBM.

10.5.2 The phases of first failure data capture

The three phases to the FFDC collection process are:

- Data generation
- Data store
- Data cleanup

Data generation

In most cases, the data to be collected is already available, but there are certain cases where a routine must be run to collect or generate necessary data. In phase one, programs that generate the FFDC data are called to collect the data that is requested to be called home. If multiple events result in a single-problem generation, only the data generation programs for the primary reference code are called. All data that is already available for all of the reference codes that are associated with the problem is included with the FFDC data.

Data store

During the data store phase, a snapshot of the data that was identified for this particular problem is copied into a staging area called Virtual RETAIN (VR). RETAIN is the back end portal that IBM currently uses for storing information about problems with customers' systems. There are two types of files that are collected:

- ▶ **Primary files:** Files that are defined to be included in the call home data.
- ▶ **Secondary files:** Files that are only copied in the VR directory. The secondary data is data that was identified to be useful in analyzing obscure conditions, but is not normally needed.

A snapshot of the data is saved because the data can change and sometimes disappear as the system continues to operate.

Data cleanup

In this final phase, data cleanup, programs are called to allow the FFDC data to be cleaned up so that the hard drive does not get full.

10.5.3 Trace

Written in the code that runs the HMC, developers placed trace calls to assist in problem analysis. Trace calls allow the HMC code developer to isolate and debug a code problem or a difficult hardware error. Trace is very detailed logging of the system events that is the lowest level of Developer debug data.

There are various levels and types of traces that help a Developer get closer to the source of the problem. Log files are created from these trace calls and they are very important for problem analysis. The way the traces work allows Developers to record what they think is important for the code and consequently when something goes wrong what the source is.

There is a shared memory buffer for trace calls; hence, not every trace call is saved. Within the code, a Developer can judge the level of importance certain errors have. For the sake of performance efficiency, the size of trace calls must be controlled because the memory buffer is shared. Trace is always running, but the specifications that the Developer gives it is what controls what the trace records.

In another effort to ensure good performance, traces are recorded on three separate buffers, each with a different scope:

- ▶ **System traces:** Are recorded for the shortest amount of time but have the largest scope.
- ▶ **Process traces:** Are recorded for a longer period of time but have a smaller scope.
- ▶ **Thread traces:** Cover the longest period of time but have the smallest scope.

Figure 10-4 on page 219 shows an example of the three trace buffers.

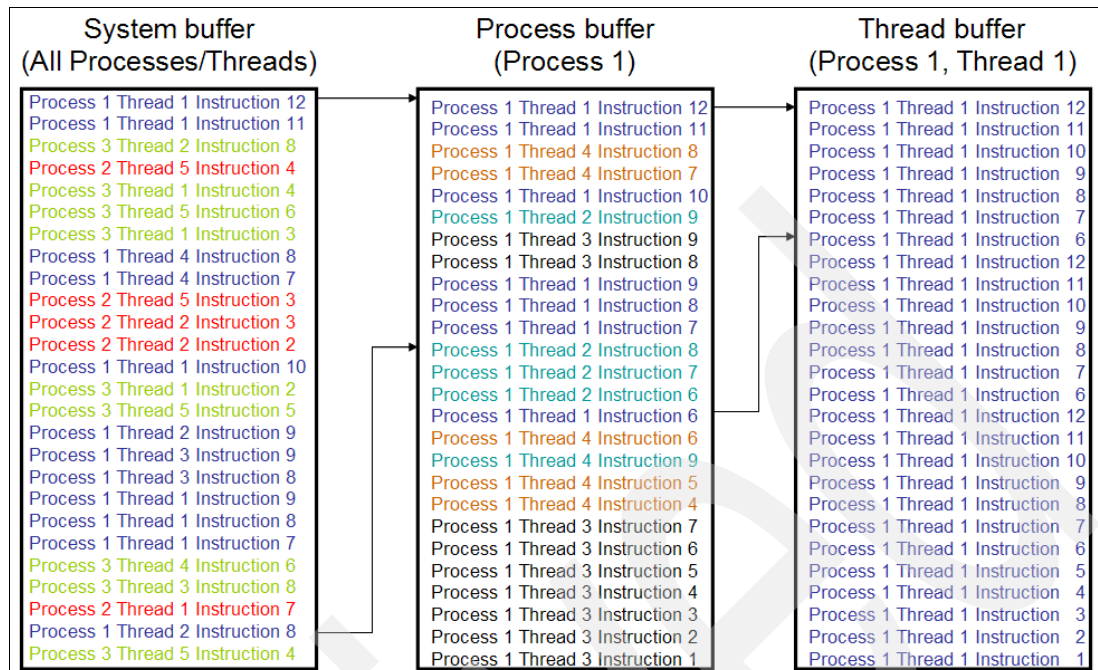


Figure 10-4 Trace buffers

10.5.4 Notification

After the data is collected, notification of the problem is made. Notification has two components. First the customer must be notified, and then IBM service must be notified using a call home operation. The customer is notified locally by creating an attention on the managed object, which causes the hardware messages icon to blink. The user then can view details about the problem. From here the user can also request service for the problem if it was not already called home. In addition to local notification, the system can be set up to notify the customer by a number of other methods, such as e-mail, pager, or an SNMP trap to a higher-level system management system.

Problem analysis gives the customer four types of information for each problem that it detects:

- ▶ What happened at a high level
- ▶ How did it affect the customer's operations
- ▶ What the customer should do about the problem
- ▶ What impact on the customer's operation the repair will have

There are a couple of reasons why the results of what happened are kept at a high level. One reason is that understanding of the low-level details of a particular problem does not really help the typical user and could in fact be misinterpreted if the information is too low level. So the message to the customer is something like "a problem was detected with the power subsystem of system XYZ" instead of a more detailed message, such as "the internal 1.8 volt current checker on the Bulk Power Distributor detected an overcurrent condition on the DCA of the CEC cage of system XYZ".

Because of all of the redundancy and recovery actions that are designed into the system, it is quite possible that a particular problem had no affect on the customer's operations. If that is the case, problem analysis tells the customer (one caveat to this type of notification is that if the condition is totally recovered with no loss of resources and no repair is required, the message is suppressed entirely). Some problems do cause some level of impact to the customer, such as:

- ▶ Loss of redundant hardware (that is, the next failure will cause a loss of resources)
- ▶ Loss of spare hardware
- ▶ Some percentage loss of a particular type of resource (memory, processors, and so on.)

When these problems occur, the system is running in a degraded state, and jobs that are running in a particular partition were aborted, or worse case, the system is no longer operating.

In most cases, the only thing that a customer must do is allow the system to be serviced. But there are types of problems that the customer does need to react to. An example of this is if the ambient temperature of the room was detected to be too hot. In this case, problem analysis tells the customer that they must ensure that the ambient temperature returns to normal operating conditions.

Most repair actions can be done concurrently with customer operations. Some other repair scenarios require some or all of the system resources to be quiesced. This information is important for the customer to know because if the repair requires that the entire system is shut down (a very rare case) and the system is currently running, the customer might opt to delay the repair until a convenient time.

10.6 Problem Analysis initialization

One thing that we have not discussed is what happens if an error is detected on the system when the problem analysis application is not communicating with the system, which can happen in two conditions:

- ▶ When communication is lost between the console where problem analysis is running (the SE) and the system. In this case, the system buffers the events locally until communications are re-established.
- ▶ If the HMC (where problem analysis is running) must be re-initialized. During initialization of the HMC, errors can be presented to problem analysis before it is fully initialized. Problem analysis handles this by saving the notifications of the logs and waiting until problem analysis is fully initialized before processing them. After problem analysis is fully operational, all of the errors that were saved are passed into problem analysis as one group. Problem analysis then applies the analysis rules to all of the errors as a whole. No attempt is made to separate the errors based on time.

When problem analysis receives the first log, an information log is taken, which is shown in Entry 1 in Figure 10-5 on page 221.

2E000000	10/26/2005 07:19:29.190[-4.0]	I	LIC_WRMSTCHK	
1B0A0000	10/26/2005 07:19:26.490[-4.0]	E	PA input	C0EA0000 00000000 00000000 00
C0EA0100	10/26/2005 07:19:26.440[-4.0]	I	OTS started	C0EA0000 00000000
E0C70100	10/26/2005 07:19:25.840[-4.0]	E	TCP/IP Srv	E0C71772 00000002 00000000 00
E0C70100	10/26/2005 07:19:25.780[-4.0]	E	TCP/IP Srv	E0C71771 00000002 00000000 00
E0C70100	10/26/2005 07:17:39.950[-4.0]	E	NetBIOS Srv	E0C7138A 00000002 00000000 00
0B500000	10/26/2005 07:17:16.850[-4.0]	I	PA PEND STAR	PAPENDINIT ← Entry 1
E1550000	10/26/2005 07:17:16.660[-4.0]	I	PersistObj	E1550172 745E77A5
E1550000	10/26/2005 07:17:16.590[-4.0]	E	UpdateRootPw	E15509A2 2D1DE452
E1030000	10/26/2005 07:17:16.490[-4.0]	E	UpgradeDataR	E103405D DD10E8E0
FFFF0000	10/26/2005 07:17:16.000[-4.0]	I	Log Info Log	

Figure 10-5 Event log summary

When problem analysis is finally initialized, a log entry, such as Entry 2, is made, as shown in Figure 10-6. At this point problem analysis is fully operational.

B1100000	10/26/2005 07:22:02.230[-4.0]	I	SHCustReport	Problem 1
B5200000	10/26/2005 07:22:01.950[-4.0]	I	Vrt1_Retain	Problem 1
14600000	10/26/2005 07:21:50.730[-4.0]	I	PA END	
B1400000	10/26/2005 07:21:50.610[-4.0]	I	PA Results	E103405D DD10E8E0 null PN 1
D1100000	10/26/2005 07:21:50.530[-4.0]	I	SHProbOpen	Problem 1
E0000000	10/26/2005 07:21:36.290[-4.0]	I	LIC_WRMSTCMP	
E0000000	10/26/2005 07:21:36.270[-4.0]	I	LIC_WRMSTCHK	
E0000000	10/26/2005 07:21:36.250[-4.0]	I	LIC_WRMSTCMP	
E0000000	10/26/2005 07:21:36.220[-4.0]	I	LIC_WRMSTCHK	
E0000000	10/26/2005 07:21:36.210[-4.0]	I	LIC_WRMSTCMP	
R5000000	10/26/2005 07:21:35.690[-4.0]	T	PA PEND END	PAPENDEND
14600000	10/26/2005 07:21:35.170[-4.0]	I	PA START	
B0200000	10/26/2005 07:21:35.080[-4.0]	I	PA RESUME	← Entry 3
B0203000	10/26/2005 07:21:35.050[-4.0]	E	RTfunc end	HOM wait
14600000	10/26/2005 07:21:34.840[-4.0]	I	PA INIT DONE	← Entry 2
B0201000	10/26/2005 07:21:34.700[-4.0]	E	RTfunc start	HOM wait
10300000	10/26/2005 07:21:32.960[-4.0]	E	UpgradeDataR	E1034064 0FD27107
10300000	10/26/2005 07:21:32.310[-4.0]	E	UpgradeDataR	E103405D 2C7449DF
00033000	10/26/2005 07:21:32.180[-4.0]	I	SPINIT	

Figure 10-6 Event log summary 2

When problem analysis resumes, it logs the event IDs of the logs that it held. The log is shown as Entry 3 in Figure 10-6.

10.7 Questions and discussions

1. What is special or unique about this problem reporting and handling design?
2. What is the link between the HMC and this problem handling design?
3. If System z machines seldom fail, why is such an elaborate problem reporting and handling scheme needed?

Archived

Change management

Code update is the subsystem that provides the ability for the System z firmware to be updated. In order for System z machines to maintain the 99.999% availability, updates to the firmware cannot cause downtime. The goal of code update is to update the firmware without requiring a reboot.

After completing this chapter, you will be able to:

- ▶ Explain the Alternate Service Element and its underlying technology
- ▶ Explain the difference between a disruptive and concurrent update
- ▶ Explain how fixes are applied to a CPC
- ▶ Explain how mainframe availability is enhanced due to code update functionality
- ▶ Perform a concurrent driver update

11.1 Overview

High on that list of critical business requirements today is the need for IT infrastructures to better provide continuity of business operations. Mission-critical application availability directly correlates to successful business operations. In today's on-demand business environment, downtime, whether planned or unplanned, is not only unwelcome—it is costly. Downtime statistics are staggering and range from tens of thousands of dollars to multiple millions of dollars per hour of downtime. While the financial impact is significant, the damage can extend well beyond the financial realm into key areas of customer loyalty, market competitiveness, and regulatory compliance.

Downtime costs will quite likely continue to rise and become more visible as enterprises increase their reliance upon IT solutions. If your site is not up or responsive when your customers want it, they are more likely to go somewhere else in the time that it takes to click a mouse. The fact is that frequent system outages in today's highly competitive marketplace can negatively impact your business. Furthermore, focusing on application availability means introducing design points that address as many of the potential causes of downtime as possible, as shown in Figure 11-1.

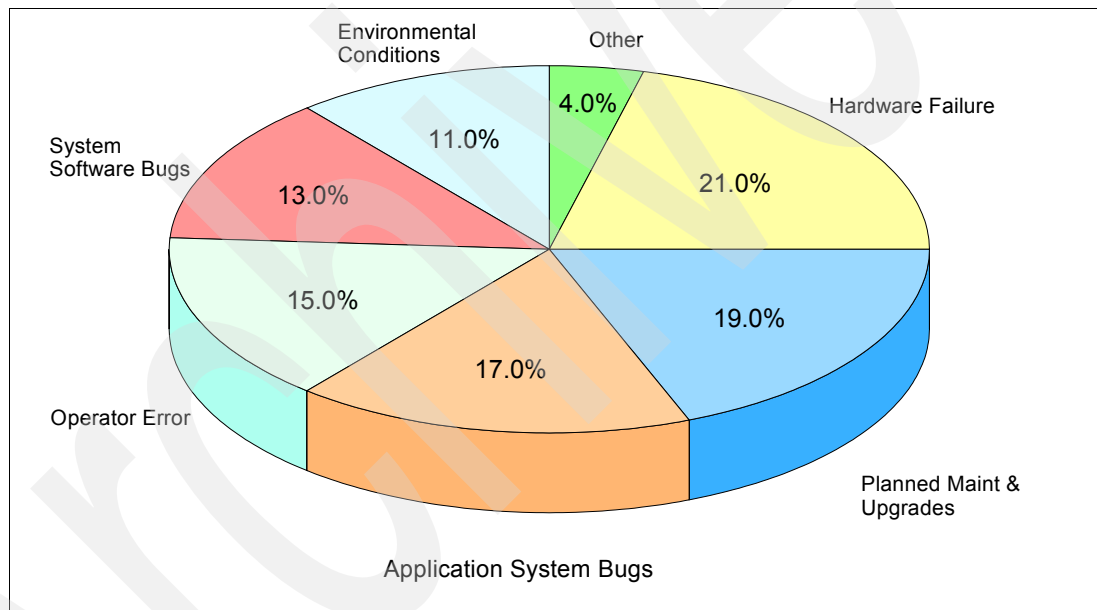


Figure 11-1 Downtime distribution

As noted in Figure 11-1, planned maintenance and upgrades are attributed to 19% of data center outages in a recent survey.

11.1.1 System z code maintenance and upgrade environment

One of the key planned maintenance and upgrades areas of design focus for the System z is in firmware upgrades and maintenance. Software and firmware changes are inherent in any product. IBM is committed to continually improving their products by delivering enhancements that provide IBM System z customers with higher application productivity, a smaller carbon footprint, continual advancements in RAS, and so forth.

The System z firmware environment that the Firmware Change Management process has to address is large and complex. System z Firmware is called Licensed Internal Code (LIC). In

the System z there are many microprocessors that have associated RAM, each containing Licensed Internal Code for function flexibility and adaptability.

The terminology can be a little confusing. The Licensed Internal Code (LIC) we are discussing is also known as microcode or firmware. These terms, LIC, microcode, and firmware are often used interchangeably. One can debate whether this terminology is correct, especially the use of microcode in this context; however, this usage is common and we use it throughout this chapter.

The microcode manages and controls the hardware for processing, communicating, error handling and recovery, configuring, environmental monitoring, service notification, and service execution. It is critical that the microcode be:

- ▶ Of the highest possible quality
- ▶ Flexible to changes to support the hardware (and itself)
- ▶ Monitored closely in the field
- ▶ Serviceable in a rapid and nondisruptive manner

11.1.2 Change management

Managing these enhancements and fixes in this large complex code environment is called change management. *Change management* is the process in which you manage changes to the firmware components, for example, changes in source code that are involved in any on-going development life cycle. It involves the process in which a change request is received, prioritized, assigned, worked upon, and closed. In maintenance projects, change requests, such as defects, escalations, or enhancement requests, are the triggers for any feature change.

They are filed either from within the organization, such as defects filed by the internal testing team or externally by customers. Change requests are triaged by the service and development teams based on priority, determination, and ultimately assigned to the components development plan. Estimations are done to determine the time and resources that are required for resolution. When these steps are complete, execution begins to address the change request.

11.1.3 System z code change request process

There is a rigorous process for managing change requests across eight million lines of code. When a change request is conceived it is documented and assigned in a change tracking tool. To ensure that no change requests are ever lost this tool is integrated into the development process, and the only way a change gets released is if it is contained in the change tracking tool. In the change request is a description of the fix, what code modules the Developer changed, and the test results. The Developer then requests approval to release the fix, which is packaged as an MicroCode Fix (MCF).

A fix, or MCF, contains the code-related files that implement the change. After the MCF is built the Developer tests it again. When complete, the MCF is sent to the IBM Product Engineering (PE) group. PE bundles many MCFs into a MicroCode Level (MCL). A MCL addresses one or more requests for a given EC stream. PE then collects MCLs from all of the firmware areas to form a bundle. A bundle is a collection of MCLs, normally with fixes for multiple firmware areas. When a bundle completes all testing, it is released to RETAIN, which is an IBM distribution vehicle. Service Representatives and customers are made aware of the availability of the fixes by communications from product engineering and the service support center and by information about the IBM Resource Link facility.

11.1.4 Engineering change stream overview

The System z delivers a new technology release about every 18 to 24 months (that is, z900, z990, z10, and so forth). The firmware that supports each new System z is labeled with an engineering change (EC) number. Each firmware subsystem is also assigned one or more EC numbers. The ECs are how the system keeps track of the firmware release and content over time, for example, the HMC contains two EC streams: one for HMCs operating system (OS) and OS-related code and one for the HMC firmware. For the Support Element, there is an EC stream for the Support Element operating system, one for the Support Element firmware, one for the LPAR hypervisor, one for channel code, and so forth. There are over 20 engineering change streams that represent all of the firmware of a System z. All of the EC streams put together represent one driver. A new driver is released with each new System z release (approximately every 18 to 24 months).

There are separate EC streams for each firmware subsystem because it makes it easier to control the release of fix levels (MCLs), for example, all LPAR hypervisor fix levels (MCLs) go against one EC stream, all HMC OS fixes go into another EC stream, and so forth. Within each firmware subsystem's EC stream fix levels are applied to update the firmware between driver releases. MCLs must be activated in a sequential fashion. To activate MCL 005 in a stream, you must also activate MCLs 001, 002, 003, and 004. This mandatory sequential MCL process ensures that each fix that IBM releases and a customer installs was tested by IBM.

11.1.5 Fix release and availability

Remote Technical Assistance Information Network (RETAIN) is an IBM controlled Microcode Fix database, among other RETAIN functions. RETAIN makes available, to System z systems world-wide, all fixes that are associated with all ECs. Using the System z HMC, the systems in the field can then retrieve the MCLs using call home functions. In addition, MCLs can be distributed by DVD, known as an AROM. Normally the retrieve from RETAIN is automatic.

11.1.6 System z change management

Updates for System z Licensed Internal Code are controlled and managed from the System z HMC and SE. In addition, each HMC and each SE has Licensed Internal Code of its own that is also subject to periodic updates from IBM. On systems with multiple HMCs, one of the HMCs is configured as a LIC Change Management focal point. The HMC that is configured as the LIC Change Management focal point monitors all System z LIC firmware levels, updates, backups, retrieval, and concurrent/disruptive application of updates without any customer effort or involvement. The HMC automatically retrieves and distributes Licensed Internal Code updates for all System z areas. The Hardware Management Console automatically retrieves LIC from IBM and distributes the LIC updates to all the HMCs and SEs for processing.

11.1.7 Retrieve, activate, and install LIC updates

The customer is given the ability to plan, schedule, distribute, synchronize, install, activate, test, backout, and track all code changes for their System z system. These action are handled by the *code update* component of the HMC and SE. There are four discrete and mandatory steps that the IBM Customer Engineer (CE) executes to complete the code update process:

Retrieve	The HMC retrieve command fetches MCLs from RETAIN or DVD. Some number of MCLs can be packaged into a combined package called a System Update Level (SUL). This command copies the MCL
-----------------	--

	files to the HMC or SE hard drive and unpacks the MCLs, so that the component MCF files are created.
Install	The HMC or SE install command places the MCL into a state where it can be activated by an activate command. An MCL must be in a received state to be installed.
Activate	The HMC or SE activate command causes a system re-initialization, which includes applying installed changes.
Accept	The HMC or SE accept command cancels removability by erasing the MCF files that are associated with the MCL being accepted. The MCL must be in the activated state.

When the install step requires a system re-initialization to be performed it is considered a disruptive upgrade. One of the contributors to downtime during planned outages is Licensed Internal Code updates that are performed in support of new features and functions. When properly configured, the System z is designed to support activating a selected new LIC level concurrently (that is, the ability to deploy firmware updates on a running system without rebooting partitions or disturbing the applications). The mechanics to make this work are substantially more complicated than disruptive mechanisms.

In developing a change or fix, the Developer must consider both the content of what is being fixed and whether the fix can be activated concurrently. Concurrent LIC upgrades require a new level of sophistication and discipline in developing, testing, and releasing changes. While this is a significant amount of added Developer effort, it provides the System z with upgrade capabilities that are leading edge in the industry. It also provides System z customers with less down time and business impact due to updates and upgrades.

More recently, a way to update and install fix streams in one step was introduced, called a single-step internal code change that provides a way for an IBM representative to upgrade the system in one quick step, which saves time. This action provides a simple method for an IBM representative to handle changes without customer interaction, thus decreasing the potential for operator errors.

Concurrent driver upgrade

The degree of continuous reliable operation was improved with every new System z release by using error prevention, error detection, error recovery, and other methods that contribute to avoiding unplanned interruptions. Planned downtime, the time needed to upgrade a System z to the next firmware driver for new functions, required a system reset to take effect. The concurrent driver upgrade (CDU) feature is provided so that customers can add new functions without downtime. Remember that a driver is, essentially, a complete new package of firmware for the whole system. Installing a new driver, without disrupting system operation, is a major technological feat.

11.2 Alternate Support Elements

The alternate SE is a redundant piece of hardware. If the primary SE fails, the alternate SE takes over until the primary SE is replaced, which requires no interaction by the user and no loss of system availability. We now look at how the alternate SE accomplishes this, and how the code for the SEs is updated.

11.2.1 Mirroring

In the computer science industry, a mirror is an exact copy of a data set. A useful feature of mirroring is that a backup exists if it is desired. Mirroring is a technology in use with the Support Elements. The mirroring support function copies appropriate data from a CPC's primary Support Element to its alternate Support Element. By regularly mirroring primary Support Element data, the alternate Support Element can be switched to function as the primary Support Element in real time.

Ordinarily, Support Element data is mirrored automatically each day, but the user can mirror Support Element data immediately if desired. When critical changes are made to the machine, such as internal code changes, new LPAR data, profile changes, or other major alterations, the mirroring functions ensure that switching to the alternate Support Element, for whatever reason, can be done safely from that point on.

11.2.2 Switching

Switching is the process of taking the primary Support Element and alternate Support Element and swapping their roles. The system automatically attempts a switchover for the following conditions:

- ▶ Primary Support Element has a serious hardware problem
- ▶ Primary Support Element detects a CPC status check
- ▶ Alternate Support Element detects a loss of communications to the primary Support Element over both the service network and the customer's LAN

When a manual switchover is started, the system checks that all internal code level information is the same on both Support Elements and that the CPC is activated. The code level is a way to determine if two separate platforms of code are the same version. If the Licensed Internal Code levels are the same on the two Support Elements, the switch can be made concurrently. The necessary files are passed between the Support Elements, and the new primary Support Element is rebooted. If a disruptive switch is necessary, the CPC is powered off before completing the switch. There are several conditions that prevent a switchover:

- ▶ Mirroring task in progress
- ▶ Internal code update in progress
- ▶ Hard disk restore in progress
- ▶ Engineering change in progress
- ▶ Concurrent upgrade engineering changes preload condition

11.3 Strategy for applying fixes

The fixes for a component on a System z server are arranged in an engineering change stream. System z servers currently have about 30 fix streams. Within each fix stream the fixes are sequential, which means that to activate, say, fix number 3 of the fix stream, you must first activate fix number 1 and then fix number 2.

11.3.1 Backing up the code on your platform

The data that is stored on the SE can be backed up and copied to removable media. Any files that changed since the last load of a driver on the SE are compressed and then copied to the

DVD-RAM. The backup is run whenever Licensed Internal Code changes are installed on the SE.

Retrieve, activate, and accepting fixes

New fixes are installed on the SE in three steps:

- ▶ **Retrieve step:** This is the first step where the user retrieves the latest fixes, either over the network or from a removable media, such as a DVD.
- ▶ **Install and activate step:** The user activates the fixes retrieved onto the SE. At this point, the fixes can be run for a while, typically a few weeks. If the user does not like the fixes, the user can remove the fixes and revert back to the previous code by running the remove and activate task.
- ▶ **Accept step:** If the user likes the fixes and everything runs well, the accept step is then run, wherein the fixes are finalized onto the SE. After finalized with the accept step, the fixes cannot be removed.

Mirroring fixes to the alternate Support Element

After fixes are activated on the SE, it is recommended that they are mirrored to the alternate SE. By regularly mirroring primary Support Element data, you can help ensure that the alternate Support Element functions the same as the primary Support Element in case a switch to the alternate Support Element is needed. Support element data is mirrored automatically each day, but the user can opt to mirror the Support Element data immediately at any time.

Single step internal code changes

Using single step internal code change the user can perform all of the necessary functions for managing fixes in one step. Figure 11-2 on page 230 shows a single step internal code changes task. The single step internal code changes task also has other more advanced functionality that includes:

- ▶ **Verifying the system environment.** In the case of the Support Element, it verifies that the most recent alternate Support Element mirroring operation was successful and that a Service Required state does not exist.
- ▶ **Performing a backup of critical data.**
- ▶ **Accepting all previously activated internal code changes.**
- ▶ **Retrieving internal code changes from the IBM Service Support System.**
- ▶ **Connecting to the IBM Service Support System and download any internal code change hold status for pending internal code changes.**
- ▶ **Connecting to the IBM Service Support System to see if the status of any existing internal code changes has changed from nondisruptive to disruptive.**
- ▶ **Installing and activating the internal code changes.**
- ▶ **Triggering the alternate Support Element mirroring operation.**
- ▶ **Transmitting system availability data to the remote support system.**

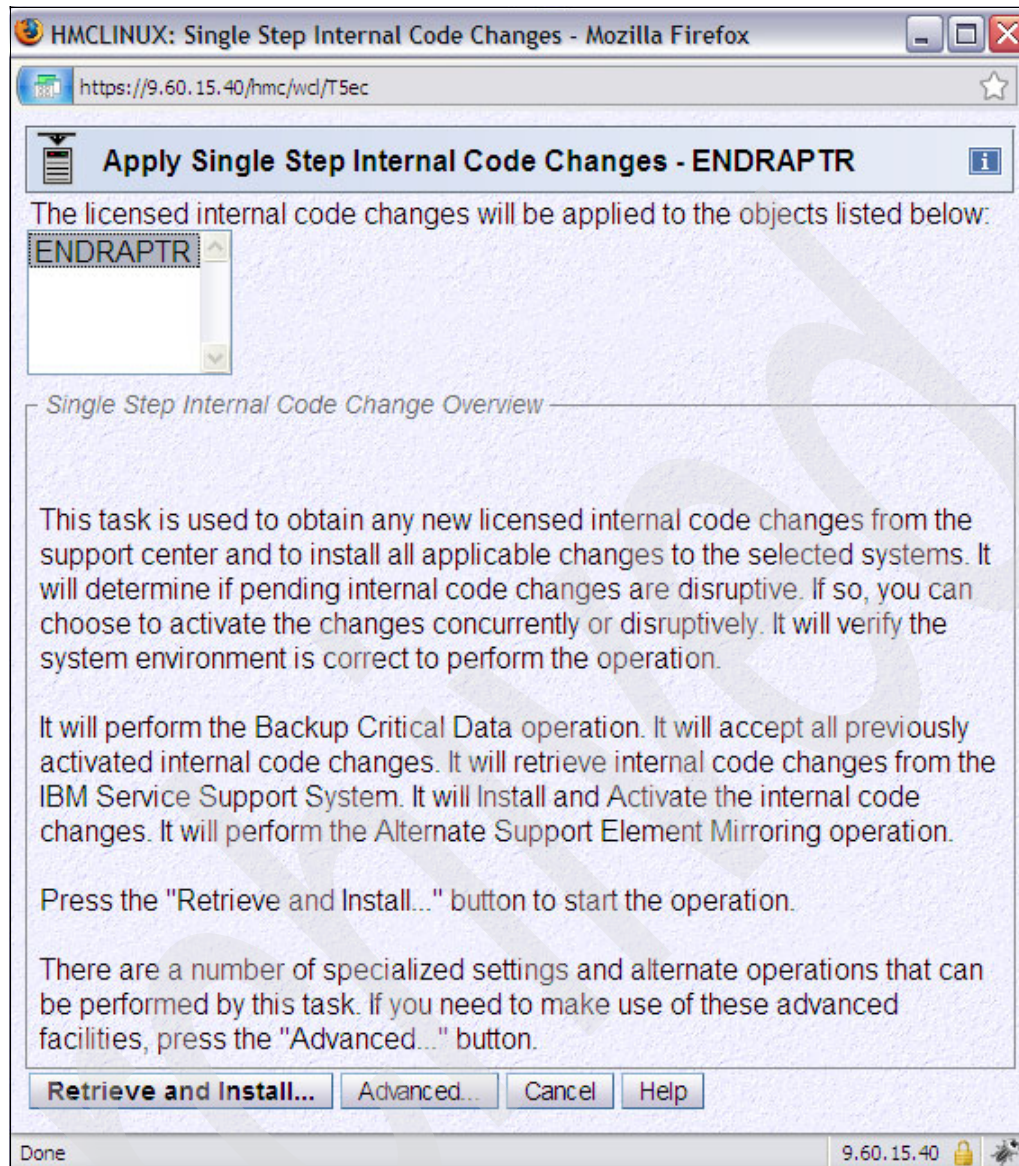


Figure 11-2 Single step internal code changes

The following options are available during single step internal code changes:

- ▶ The user can determine if the pending internal code changes are disruptive and then choose to activate the changes either concurrently or disruptively.
- ▶ The user can retrieve and apply (or clone) internal code changes to match a saved cloneable level.
- ▶ The user can receive all code changes for the system regardless of the requirements for installing the next engineering change level.

Figure 11-3 on page 231 shows a portion of the advanced options.

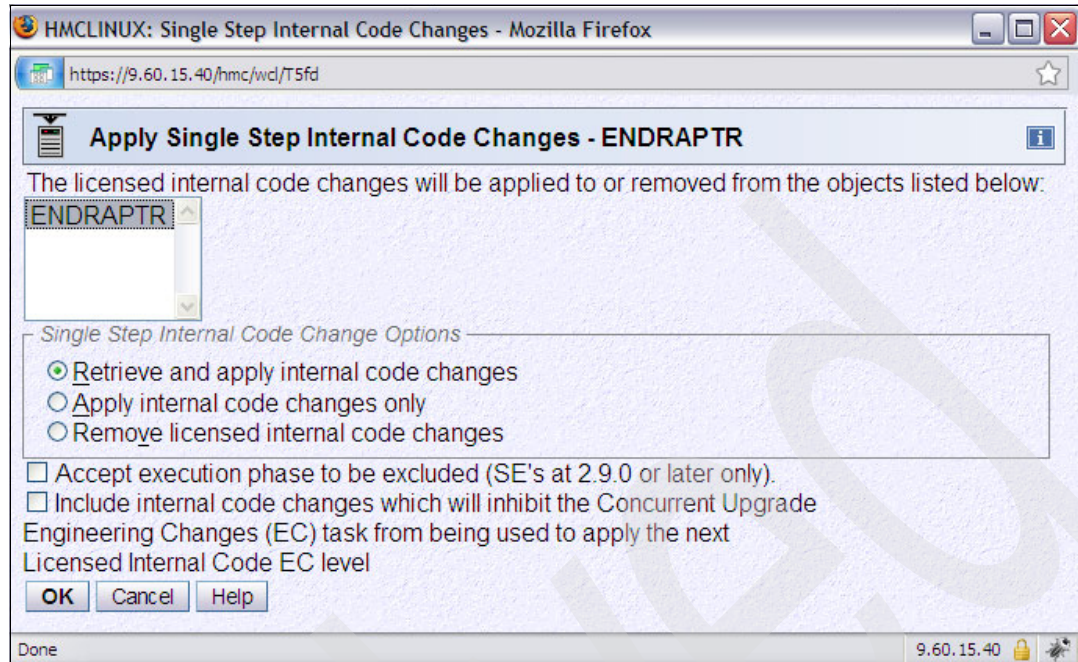


Figure 11-3 Apply single step internal code changes

11.4 Strategy for updating driver

Updating drivers can be done in two different ways:

- ▶ Disruptively: These updates require that the CPC be deactivated as the updates are installed.
- ▶ Concurrently: Updating concurrently allows the CPC to continue to run as fixes are being applied.

11.4.1 Updating drivers disruptively

Fixes can be updated and installed to a CPC by running the engineering changes task from the HMC. Code is copied from a CD or DVD-RAM to the primary SE of the CPC. If both the operating system and the SE code is upgraded, the fix is disruptive, the CPC is deactivated, and the operating systems are stopped. If just the operating system is going to be upgraded, the upgrade can occur concurrently.

11.4.2 Updating drivers concurrently

The major benefit of upgrading firmware by using concurrent driver update (CDU) is that no planned downtime is required. The system can continue processing its normal workload without any performance impact.

A typical driver upgrade through CDU, including the SE preload, takes approximately three hours. A disruptive driver upgrade requires the transfer of the workload to a different system, the shutdown of all applications and operating systems, and a system shutdown. Activating the system again with the new firmware driver, including software IPL, might then take up to eight hours. By using CDU, the customer can continue using the system while it is being upgraded and also save significant time compared with a traditional driver upgrade.

Concurrent driver upgrade was first provided with the z9 mainframe. Customers could upgrade their z9 mainframe to add new functions by using the CDU process. Enhancements were made to CDU with the z10 to make CDU more efficient and more bullet-proof.

Planning for and performing a concurrent upgrade

Before a concurrent upgrade can be performed:

- ▶ The HMC must be configured to allow a CDU.
- ▶ The user must enable Service Status on the CPC, as shown in Figure 11-4.
- ▶ Internal Code Changes must be authorized, as shown in Figure 11-5 on page 232.
- ▶ The alternate SE must be checked to see if it is communicating properly with the primary SE. After these settings are configured, the fixes can be preloaded for the concurrent upgrade.
- ▶ The system must not be in a Service Required State.

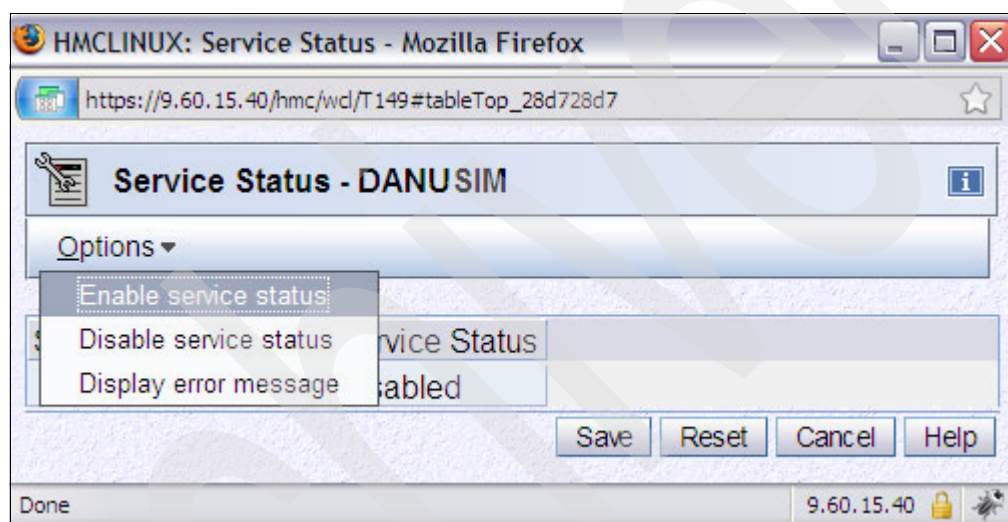


Figure 11-4 Service status

After the requirements are met, the driver from the AROM DVD¹ can be preloaded, as shown in Figure 11-6 on page 233. The CDU preload loads the driver on the alternate SE. After the driver from the AROM DVD is preloaded, fixes must be preloaded from a SUL DVD. The new driver can now be activated. After the new driver is activated, all functionality from the previous code level must be checked to make sure it is working.

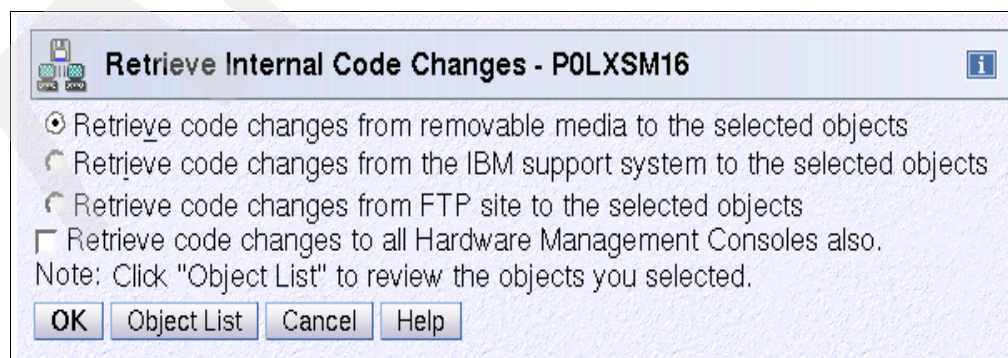


Figure 11-5 Retrieve internal code changes

¹ Complete drivers are normally delivered on a DVD instead of by a dynamic download.

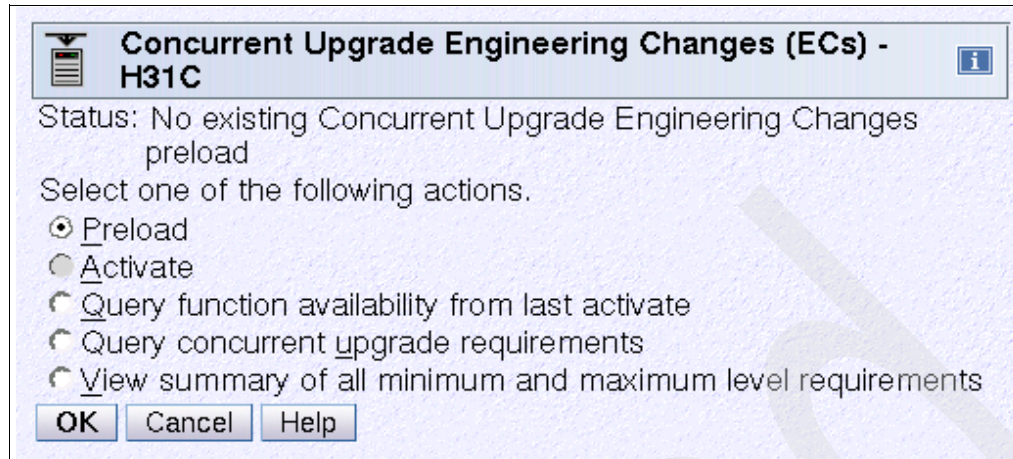


Figure 11-6 Concurrent upgrade engineering changes

11.5 Underlying technology

In this section, we review the technology that is behind the features of the code update functions.

11.5.1 Alternate Support Element

The alternate SE is built when a backup critical data is done from the primary SE and then a hard disk restore is done to the alternate SE. After the alternate SE is built it notifies the primary SE through TCP/IP and the primary SE writes an internal record (to a SIM field) indicating that an alternate SE is installed and that it is to begin mirroring to the alternate SE. These functions require considerable innovation during development, and we take a look at several areas that are involved.

Primary SE selection program (role determination)

The Support Element provides multiple paths leading to its own initialization: the power-on sequence of the ThinkPad, a reboot of the ThinkPad (which can also be initiated remotely from a HMC), a local key stroke combination, or an internal recovery function. During this initialization, the SE must determine whether it is to be a primary or an alternate SE. It is very important to the customer to ensure that whichever SE was last primary remains as the primary SE because there is a possibility that the two SEs contain slightly different data (if they were not yet mirrored after data was customized on the primary SE or if an engineering change task or Licensed Internal Code changes task was performed that placed one SE at a different microcode level than another SE). An exception to this is if the primary SE is dead (hardware problem).

During the time when a SE is determining its role as a primary or alternate SE, the SE is in a pre-SE state. A soft switch file on the SE hard disk identifies which SE was last primary. While in the pre-SE state, the SE first tries to communicate across the service network using TCP/IP to determine whether the other SE is already established as the primary or alternate SE or is in the preSE state and if it has the soft switch file. If the other SE already has a defined role as primary or alternate, the pre-SE takes on the opposite role. If the other SE is also in a pre-SE state, the SE uses an algorithm based on the existence of the soft switch file on each of the pre-SEs to determine which SE must be primary and which becomes alternate.

11.5.2 Making mirroring efficient

An SE mirror operation occurs when a primary SE's hard drive data is mirrored to an alternate SE's hard drive. The source data is not the entire primary SE's hard drive but includes enough data to ensure that the alternate SE can function exactly like the primary SE.

The mirror operation from one SE to another involves obtaining relevant file and directory time stamp information from the alternate SE. These file and directory statistics are gathered by traversing each appropriate directory, obtaining the statistic, and writing a file with this data. When complete, the file is sent to the primary SE. The primary SE then traverses its own corresponding directories and obtains its own file and directory time stamp information. This time stamp information is compared with the alternate SE's data. If the primary data indicates a time stamp that is different than the alternate's time stamp, the corresponding file is included in a zip file that is to be created. After complete with the analysis of time stamp comparisons, the primary SE creates a single zip file that contains a copy of all of the files that require updating on the alternate SE. This zip file is sent to the alternate and unzipped in a *staging area*. At an appropriate time, the alternate SE moves the data from the staging area and replaces the appropriate files.

A major portion of the time that is needed for a mirroring operation is the comparison of the primary's statistics (time stamps) to the alternate's statistics. In an earlier mirroring design, as the primary SE traversed its directories, the entire file of the alternate's statistics was searched for a matching directory and file name. In the current design, only selected portions of the alternate's statistics file is searched. The new design significantly increased the overall mirror performance.

11.5.3 Primary Support Element outage tracker

The primary SE outage tracker is a fairly new feature that tracks any intervals when there is no primary SE available or when the role of primary SE is shared between two SEs. To the extent possible, it also records the reasons for each outage. In this section, we provide a brief description of the design.

Two files are used: the outage tracking file and the outage reason file. The outage tracking file is stored on the primary SE. This file is written at the end of the outage, when the primary SE is again available. After the primary SE comes up and writes the outage tracking file, it sends a copy to the alternate SE as soon as possible.

The outage reason file keeps track of the beginning of the outage. Its time stamp indicates when the outage started, and the file contains the reason for the outage of the primary SE. The outage file is written every minute by the primary SE and also by the alternate SE when it has a link to the primary SE. If there is a power cycle, power outage, or primary SE crash, we want to know the last point in time when the primary SE was working. The outage file is written at the following times:

- ▶ Whenever the primary SE reboots with a reason code indicating why it is rebooting, for example, a user initiated boot, a boot after installing fixes, and so forth.
- ▶ Whenever the primary SE initiates a switch with a reason code of switch from a functional primary.
- ▶ Periodically (every minute) by the primary SE with a reason code of power cycle.
- ▶ Periodically (every minute) by the alternate SE with a reason code of switch from non-functional primary.

The outage reason file and the outage tracking file are sent from the primary SE to the alternate SE during a switch. The outage reason file contains a reason code of switch from a

functional primary. When the primary SE reboots it rewrites the switch reason file with a reason code of failed switch. If the switch fails and the same primary comes back, then when the outage tracking file is written it has the reason code of the failed switch. If the switch succeeds and the old alternate SE comes up as primary, the file will have a reason code of switch from a functional primary.

A base version outage tracking file is included in the driver. The outage reason file is not included in the base driver. When the SE comes up and there is no outage reason file, such as after an EC upgrade, the SE checks if the outage tracking file is the base version. If it is the base version, the date and time stamp of the initial load are stored in the file. If it is not the base version it searches for a hard disk restore marker file. If a marker file does not exist then it assumes there was an initial load with no outage reason file, and an outage file with unknown reason is created, which also triggers a call home operation.

The outage tracking file is saved and restored during a CDU. The outage tracking file is mirrored and backed up. The outage reason file is not mirrored or backed up. When there is a hard disk restore of the primary SE, the code creates a marker file. When the primary SE comes up and there is no outage reason file, the outage tracking file is not initialized, and the new marker file exists, then a reason of hard disk restore is written to the tracking file. After the primary SE comes up and writes the outage tracking file it searches for the new marker file and erases it.

11.6 Fixes

In this section, we discuss how new code is updated and applied to a CPC. Fixes are a complicated process involving many steps to provide up-to-date code without reinstalling the entire SE or HMC operating system each time an update is issued.

11.6.1 Dependencies between fixes

Fixes for the SE can have three different types of dependencies:

- ▶ A fix can have a pre-requisite dependency, which means the fix requires a certain dependency to be activated before or at the same time as the current fix.
- ▶ The second type of dependency is a co-requisite dependency, where the current fix and its co-requisite dependency must be activated simultaneously. In reality, one must be activated slightly before the other, and if it is the case where this cannot be done, when the operating systems are activated the fix must then be marked as disruptive.
- ▶ The last type, activation required (actreq), requires that the dependency be activated before the current fix.

Resolving concurrency when there are dependencies

One of the most complex elements of the whole update process is the code that determines if a fix that is marked as concurrent is still concurrent after all dependencies are considered. For a fix to be concurrent:

- ▶ It must be marked as concurrent
- ▶ All of its dependencies must be available and marked as concurrent
- ▶ Because the fixes are mandatory sequential, all fixes between the original fix and the activated fix must be concurrent
- ▶ All fixes between each dependent fix and the activated fix must be marked concurrent.

- If any of the fixes between the dependent fix and the activated fix have dependencies, they must be checked as well. The solution requires recursion

We take advantage of the object-oriented programming to solve the problem. Each fix is represented by a fix object in memory. Each fix object denotes if it is disruptive by dependency, that is if it is nondisruptive by itself, but some dependency causes it to be disruptive. To do this it keeps track of any prerequisite and corequisite fixes that it has and which of those dependencies are currently disruptive. Whenever the state of a fix in a fix stream changes the entire fix stream is looped through again and each fix is checked for dependencies. Each dependent fix contacts any corequisite fixes (and, if deactivated, its prerequisite fixes) to recalculate whether or not it is disruptive by dependency. If the disruptive by dependency attribute changes, the requisite fix is either added or removed from the dependent fix's disruptive by dependency list. The fix stream of the requisite fix is also added to a list of fix streams that need its concurrency recalculated after we resolve whether each of the requisite fixes are disruptive by dependency.

In addition, when a fix is discovered to have a dependency on another fix, the fix stream is notified that the original fix must be contacted when the state of the dependency fix changes, which takes care of situations where the original fix is retrieved, but the dependency fix is not retrieved yet. After the dependency fix is retrieved, the dependency information about the original fix is checked.

11.6.2 Hex patching

Hex patching is a method of applying updates to an existing file without having to generate a completely new file. A *diff file* is generated and applied to the existing file. A diff file is usually much smaller in size than a complete file, allowing for quicker and smaller updates. We now look at the hex patch file format and how it is applied.

The hex patch file format consists of short control sections, followed, in most cases, by data segments that the control section uses. The current control sections specify where changes are made in the file, how many bytes to manipulate, and a control section for version information.

The basic process is to move bytes from the existing file into a new file until a control section takes effect. The control sections are ordered so that they move through the file sequentially. When a control section offset is reached, its action is performed.

A replace action ignores the indicated number of bytes from the old file and, instead, uses that number of bytes from the hex-patch file and puts these in the new file. These new bytes immediately followed the control section in the hex-patch file. An insert action takes the indicated number of bytes from the hex-patch file and places them in the new file at the indicated point. No bytes from the old file are ignored. A delete action ignores the indicated number of bytes from the old file. There are no bytes of data in the hex-patch file that are associated with this control section. After the ignored bytes, bytes are again transferred from the old file to the new file.

The checksum control section contains a checksum for the resulting new file. After processing all of the hex-patch file control sections, the resulting file must be the same as the file that is shipped if the file is completely replaced. The checksum function helps to verify the correctness of the patch. The checksum control section occurs only after, at the end of the hex-patch file.

The version control section occurs only after, at the beginning of the hex-patch file. If it is absent, then the version is defaulted to being an older format that did not contain a version control section. It contains version information to identify which hex-patch file format is being

used. It also might contain additional information indicating the hex-patch algorithm that is used.

When an error occurs with hex patching, it is usually a case of the hex-patch file not matching the level of the existing old file, which can happen for a variety of reasons, including:

- ▶ The target file was overlaid by another file (or another level of the original file).
- ▶ Mirroring or a restore function changed the level of the target file, such that the patches no longer fit it.
- ▶ Patches or updates were applied in the wrong order.
- ▶ A defective disk drive is returning invalid records and the patches do not fit these records.

Although a hex-patch error is clearly identifiable (typically, an incorrect checksum), it is not always obvious what the source of the error is. Is the old file at the wrong level? Is the old file really a private fix?

To aid in diagnosing hex-patch problems, the fix control file controls checksum information in the fix control file. It provides the checksums for the old file and the new file. It also provides a checksum for the hex-patch itself. These checksums are CRC-32-bit checksums that are produced by using the built-in java CRC support. These CRCs are industry standard, so they are valid on any platform. Note that they are 64 bits long, not like the 16-bit checksum that is used in the hex-patch.

These checksums are used by the hex-patch processing to help clarify the problem if a hex-patch error occurs that corresponds to a failure in performing the patch or a checksum mismatch after applying the patch. When an error is detected, we compute the checksums for the hex-patch file and the old file and include those checksums and the expected checksum from the fix in the diagnostic information (FFDC information) that is logged. This process should make problem determination a bit easier. A checksum mismatch cannot determine why the content of the file was altered; instead, it clearly indicates that the content of the file is not equal to that of the original file. Example 11-1 is an example of a log that is produced.

Example 11-1 Sample log

```
XMCHXPFF:    HexPatch.collectDiagnosticData() - Hex-patch diagnostic
information:
XMCHXPFF:    HexPatch.collectDiagnosticData()          The old file name is
[/console/pucode/i390code.bin]
XMCHXPFF:    HexPatch.collectDiagnosticData()          Expected CRC32 of the old
file: 00000000677ED6A2
XMCHXPFF:    HexPatch.collectDiagnosticData()          Actual CRC32 of the old
file: 0000000082BB6B9F
XMCHXPFF:    HexPatch.collectDiagnosticData()          The hex-patch file name is
[/console/mcf/RM07601001_/console/pucode/i390code.bin]
XMCHXPFF:    HexPatch.collectDiagnosticData()          Expected CRC32 of the
hex-patch file: 00000000F60D717F
XMCHXPFF:    HexPatch.collectDiagnosticData()          Actual CRC32 of the
hex-patch file: 00000000F60D717F
XMCHXPFF:    HexPatch.collectDiagnosticData()          The new file name is
[/console/mcf/RM07601001_/console/pucode/i390code.bin]
XMCHXPFF:    HexPatch.collectDiagnosticData()          The work file name is
[/console/mcf/RM07601001_/console/pucode/i390code.bin.23608.work]
XMCHXPFF:    HexPatch.collectDiagnosticData()          Expected CRC32 of the work
file: 000000002AE9C65D
XMCHXPFF:    HexPatch.collectDiagnosticData()          Actual CRC32 of the work
file: 00000000E5B6FFEF
```

XMCHXPFF: HexPatch.collectDiagnosticData()	Expected 16-bit checksum
of the work file: 00000000000014FF	
XMCHXPFF: HexPatch.collectDiagnosticData()	Actual 16-bit checksum of
the work file: 00000000000082DF	
XMCHXPFF: HexPatch.collectDiagnosticData()	Hex-patch file format
version: 0	
XMCHXPFF: HexPatch.collectDiagnosticData()	Last control section type:
FFFFFFFF	
XMCHXPFF: HexPatch.collectDiagnosticData()	Last control section
offset: 0000000000BAC0F8	
XMCHXPFF: HexPatch.collectDiagnosticData()	Last control section
length: 00000004	
XMCHXPFF: <- HexPatch.collectDiagnosticData()	

A call home function collects and sends the files to IBM for additional analysis.

11.6.3 Cloning

Cloning is a feature that was developed to easily upgrade a machine to match the firmware level of another machine. Businesses that have a test machine can run a certain code level for a while and, if everything goes well, can then decide to bring their other machines up to that same code level. Originally, this was not a simple process, so a *cloning* function was developed. Cloning allows any machine to retrieve a cloneable level and then install the fixes that the cloneable level defines.

After the machine with the desired code level is determined, a task is run that creates a cloneable level of that machine. Figure 11-7 on page 238 illustrates this function.

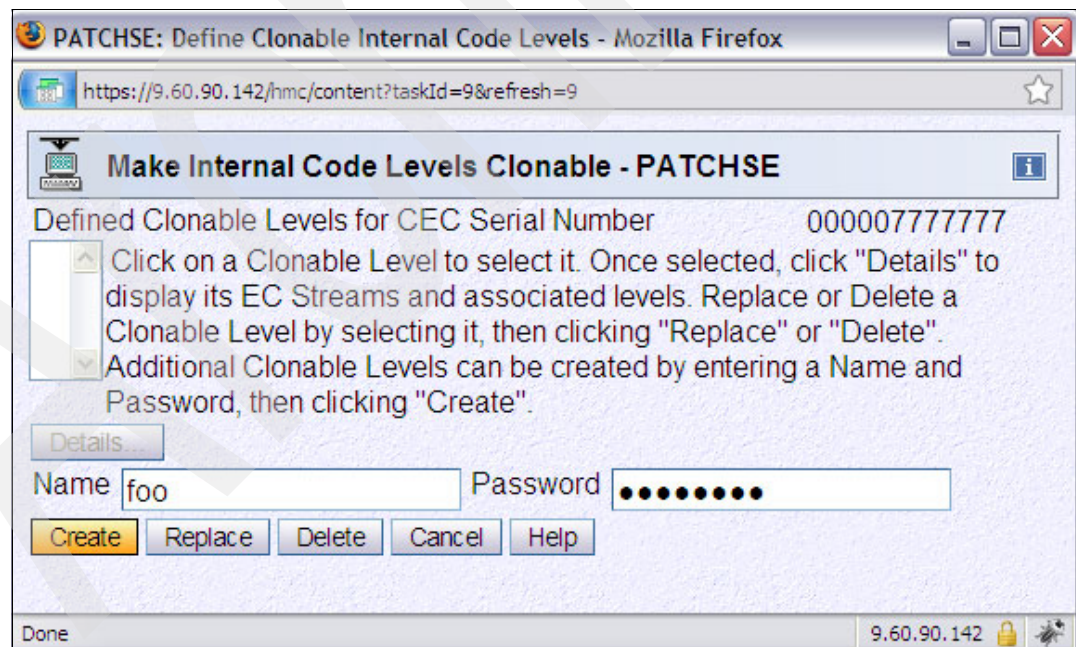


Figure 11-7 Cloning

The cloneable level is then sent to a central hardware and software service database that contains all cloneable levels that were created. This database can be accessed from any machine, anywhere in the world. Any machines that are to be updated to this cloneable level

retrieve the database entry by name and password and run a Single Step CDU that installs these fixes to the machine.

A bookmark is not actually software; instead, it is a cloneable level data file (CLD) that contains a list of fixes that are installed on the machine. When another machine retrieves that bookmark, the machine gets a list of fixes so that it can retrieve and install.

A machine can attempt to install a cloneable level and discover that it has different hardware than the clone base, which implies that they need different fix streams. In this case, only the components that both machines share are cloned.

11.6.4 Alerts

Alerts are a means to send a message to the Customer Engineer before an install and activate (for fixes), remove and activate, or a single step internal code change is started. Alerts can be used for fixes that are applied to an SE or fixes that are applied to an HMC. Alerts are always in English.

After the user selects whether the install and activate are concurrent or disruptive, a query is done to see if there are any alerts. If there is an alert for any of the targeted CPCs, a message is displayed that includes the name of the CPC that has the alert and the alert text. A separate message exists for each alert and for each system that is associated with the alert.

For single step internal code changes, the alert message is displayed prior to the confirmation panel, if any alerts are detected. In addition, if fixes are retrieved that would trigger a new alert, the single step internal code change process is stopped after the retrieve step, and a completion message is displayed indicating that new unreviewed alerts were detected. To review the alerts, repeat the single step internal code change operation.

If there are alerts when a scheduled operation for install and activate, remove and activate, or single step internal code change is started, the code triggers a hardware message. A manufacturing preload or the Activated Read Only Memory (AROM) process are not blocked when alerts are contained in the fixes to be installed and activated and alerts are not displayed.

Any fix that requires an alert in a driver (other than an initial release driver for a product) must be carefully considered if it can go on during a CDU. If it cannot, then the IBM Product Engineering (PE) group must be involved to help resolve the situation.

The CDU detects whether there are any alerts in the set of fixes that it retrieved. If there are, it does not activate the fixes; instead, it produces a message that indicates that unreviewed alerts exist and suggests that the user issue a single step internal code change to review the alerts and activate them.

There is a possibility that by retrieving from a DVD, which excludes the fixes from the alert fix stream or by doing an install and activate specific without using the alert fix stream, a user can activate a fix level (MCL) that requires the alert without seeing the alert. To protect against this, IBM includes a prereq for each MCL that requires an alert, pointing to the first fix in the alert fix stream that contains the alert.

The alert XML file is an XML file that has a self-contained DTD to ensure that the file is built with proper syntax. There is a separate fix number for the alert fix stream for HMC, SE, and TKE. It includes tags, such as:

- ▶ <alerts> is the outer tag of the document that includes all of the alerts.
- ▶ <alert> contains all of the data for a single alert.

- ▶ `<ec_range>` is a tag that contains all of the data for a range of fixes within a fix stream. There can be multiple `ec_range` tags within an alert. When there are multiple `ec_range` tags the alert is displayed when any of the fixes in any of the ranges are being processed.
- ▶ `<ec>` is the fix stream that causes the alert to be displayed.
- ▶ `<max_mcl>` is the maximum fix number within the fix stream that causes the alert to be displayed.
- ▶ `<min_mcl>` is the minimum fix number within the fix stream that causes the alert to be displayed.
- ▶ `<when_to_alert>` indicates when the alert is to be displayed:
 - If the operation is install, the alert is displayed on install and activate.
 - If it is remove, it is displayed on remove and activate.
 - If it is both, it is displayed on both.
- ▶ `<description>` provides actual text of the alert. The description can include `
` HTML tags, which are interpreted as line breaks when displaying the alert panel.

Example 11-2 is an alert file that contains two alerts. One alert is displayed prior to install and activate of J99677 fix level (MCL) 407. One alert is displayed prior to install and activate or remove and activate of any MCL in the J99660 stream.

Example 11-2 An alert file that contains two alerts

```

<alerts>
  <alert>
    <ec_range>
      <ec>J99677
      <max_mcl>407
      <min_mcl>407
    </ec_range>
    <when_to_alert>install</when_to_alert>
    <description>If your HMC has system EC G40102, then you must ensure
    MCL 055 is activated.<br><br> When you have addressed this issue, click
    OK to continue processing. Click cancel to cancel the
    operation.</description>
  </alert>
  <alert>
    <ec_range>
      <ec>J99660
      <min_mcl>001
      <max_mcl>999
    </ec_range>
    <when_to_alert>both</when_to_alert>
    <description>Ensure you have configured off any chpids configured as OSN.
    .<br><br> When you have addressed this issue, click OK to continue
    processing. Click cancel to cancel the operation.</description>
  </alert>
</alerts>

```

The code to handle processing of alerts is implemented in Java. The `MclAlert` class contains a single alert. The class contains methods to get the description text of the alert, the system name that the alert pertains to, and an `UpdateLevel` that represents the MCL that triggered the alert.

The McAlertReader class exists to process the alerts. This class has several methods to designate its functionality. Methods exist to:

- ▶ Store the current retrieved and activated MCL levels so that we know what alerts were reviewed
- ▶ Get the list of alert objects that must be reviewed
- ▶ Record informational events in the log contriving the text of the alerts agreed to
- ▶ A convenience method that gets the alerts, presents them one-by-one to the user, and if agreed to writes the informational events to the log

11.7 Concurrent driver upgrade

The first step in the CDU process is to initially preload the alternate SE with the release n+1 code while the primary SE and the system continue to operate using the release n code. There are two initial preload options: initial preload including MCLs from the IBM Support System and initial preload only. Both options put the base release n+1 driver on the alternate SE using the CDU DVD-RAM, and the only difference between the two options is that the first option downloads any additional fixes that were released after that CDU AROM was released.

The other two preload options allow the download of additional fixes that were not part of the initial preload step. Using these preload options can enhance CDU performance by avoiding the download of additional fixes over a modem connection, which, for security reasons, is a likely setup in a customer environment.

When the user selects one of the CDU initial preload options, the HMC validates that the CDU AROM appropriately matches the from-release-level system EC. The primary SE is requested to validate that the concurrent patch is enabled from previous activities and to retrieve and apply the latest CDU min/max file, ensuring that the latest CDU requirements are known.

Next, a check is made to ensure that the CDU minimum and maximum release n requirements are met. If CDU minimum MCL requirements are not satisfied, but no CDU maximum requirement is exceeded, the user is given the option to concurrently apply additional release n MCLs to meet the CDU requirements. If at least one CDU maximum MCL requirement is exceeded, the user is told that the CDU is not possible. IBM does not generally recommend removing fixes. The customer must wait for the next CDU switch point.

After this validation process completes, the HMC triggers the primary SE configuration and customization upgrade data to be put on the alternate SE upgrade partition, which ensures that the data is preserved during this transition from release n to release n+1 code. The alternate SE then downloads from the HMC an image of the operating system partitions and an image of the firmware partitions. After the download is complete, these two images are unpacked and overwrite the data in all partitions on the alternate SE hard disk except for the upgrade partition. The upgrade data is then restored by taking it from the upgrade partition and transforming it into the release n+1 code structures.

Finally, if the CDU initial preload option included the request to retrieve fix levels (MCLs), the alternate SE retrieves, from the IBM Support System, any additional MCLs that were not part of the CDU AROM. These additional MCLs are applied on the alternate SE hard disk following certain defined restrictions because the CDU activation process must concurrently manage the firmware updates in memory where that firmware executes. These defined restrictions are derived from the CDU min/max file. The CDU preload options, which do not include initial preload, can be executed only after the initial preload is performed.

Figure 11-8 shows a concurrent upgrade.

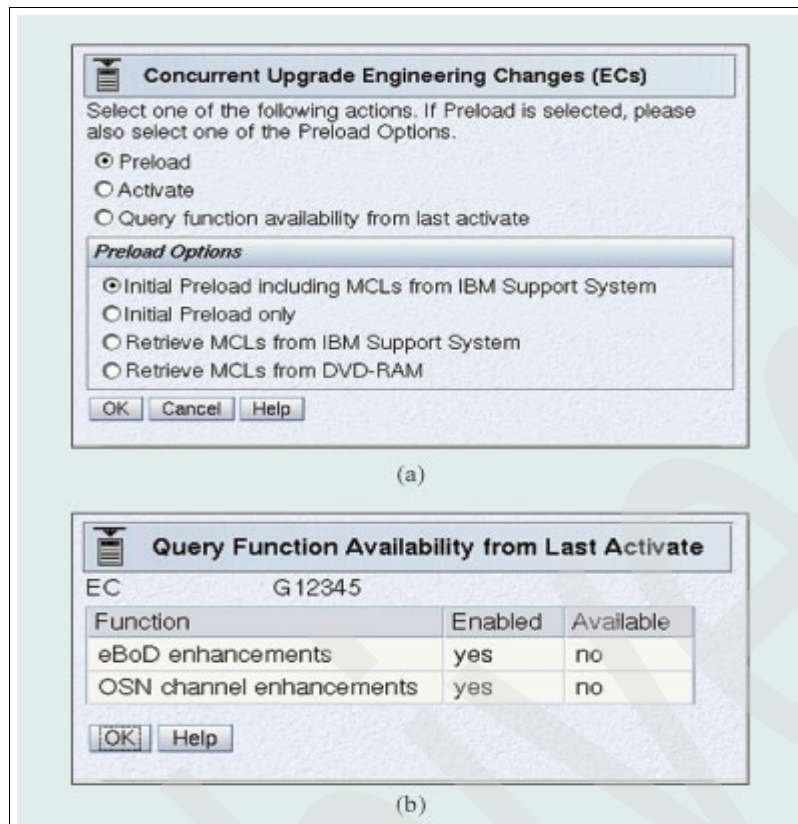


Figure 11-8 Concurrent upgrade

Preparation phase

By selecting the **CDU activate** option from the main CDU option panel, shown in Figure 11-8 on page 242, the following requirements are verified before the code switch starts:

- ▶ The concurrent patch feature is enabled.
- ▶ The CDU min/max requirements are met (MCLs might have been applied to, or removed from, the release n code primary SE since the CDU preload).
- ▶ No pending conditions from previous CDU or concurrent patch sessions exist.
- ▶ There is enough free storage for the additional release n+1 HSA. The driver information contains the CDU release n+1 storage requirements. If there is not enough memory, the customer is told how much memory to free.

After the verification, the HMC needs to know if the mainframe is ready for the very first or next CDU. Functions that were added by using a previous CDU might require special activities to activate the new code. Such new functions would be available but not yet enabled.

The very first CDU is always allowed. However, the next CDU might require that some new functions that were added with the last CDU be available and enabled. CDU can be performed only on a well-defined from firmware level, which implies that the very first step in the CDU preparation phase verifies that the CDU is allowed.

The status of all functions that were added with previous CDU switches is stored in the function availability information (FAI) list. This information is kept over the lifetime of a system. Whether the next CDU is allowed depends on the information in the FAI and is reported to the HMC through a CDU allowed indicator.

After verifying that another CDU can be performed, the HMC checks whether the HSA is sufficient for the new functions. The minimum i390 code firmware level contains the information about the amount of memory that is required for each new function. The total amount of memory for the driver switch is calculated by adding the amount of memory that is required for all new functions. An HMC panel informs the customer about the need for additional free memory if the required HSA exceeds the amount of memory that is available. The customer must provide memory by deactivating a partition or varying off storage in a partition if the complete memory is used.

After verifying the availability of free memory, the HMC issues a command to the i390 code to start the HSA extension. This step marks the point of no return: The firmware and the HSA layout is changed.

The CDU i390 code calls every new function to allocate the required HSA. After each call, the i390 CDU code verifies that there is still enough free memory left to back the HSA requests of all remaining functions that were not yet called. If there is not enough memory left (most likely caused by a new function requiring more HSA than expected), the HMC stops the CDU activation process and informs the customer about the additional memory that is required. After the customer provides the additional memory, the CDU must be started again. The HSA allocation process starts again at the point at which it stopped.

Transition phase

When the CDU activate preparation phase completes, the transition phase begins, which is the heart of a CDU activate because it is in this step that the release n+1 code is applied in each firmware subsystem memory. The SE is the first subsystem to have its release n+1 code loaded, which occurs by executing a CDU alternate SE switch. This causes the release n+1 code to become the new primary SE while the release n code becomes the new alternate SE. The release n+1 new primary SE must restart and perform a warm-start resynchronization with the other firmware subsystems.

This release n+1 SE code must communicate with the release n code to obtain information about the state of the system. In addition, if service or functional requests occur, the release n+1 primary SE code must be able to interact with the other release n subsystem code. After the primary SE completes its warm-start resynchronization, it serially triggers each subsystem to load its release n+1 firmware in the following order: FSP, power, i390 code, millicode, all of the channel subsystems, enhanced initial program loader, LPAR, and CFCC. While additional firmware subsystems transition to the release n+1 firmware, all subsystems must continue to interact with the mixture of release n+1 and release n code levels. The next paragraphs provide an example of the details of what a firmware subsystem must go through to perform the transition from release n to release n+1 CDU code.

The concurrent patch feature of the i390 code and millicode uses a primary and an alternate code load area to switch between two firmware levels. The executable and linking format (ELF) loader receives a new millicode and a new i390 code load at the start of an i390 concurrent patch sequence. Both code loads are copied into the alternate code load area. The ELF loader and millicode perform some relocation activities to map the new code loads to existing data. Finally, the ELF loader calls millicode to perform the actual code switch.

The existing i390 concurrent firmware patch support of the ELF loader did not allow the addition of new static external variables. However, new functions implemented in i390 depend on the ability to store information permanently and might require that new static variables be created and initialized during the concurrent patch application of i390 code. To reduce the complexity of CDU and avoid any delay during the concurrent patch sequence, the dynamic HSA feature is not used. Instead, the ELF loader allocates spare HSA space at POR time to allow for the growth of i390 code and i390 data area sizes during CDU. The preplanned size

of the reserved HSA memory is derived from previous code and data-area changes among drivers.

The ELF loader distinguishes between CDU and the normal i390 concurrent patch case. New function pointers and new remote procedure calls (RPCs) are allowed for CDU, but not for concurrent patches because normal i390 concurrent patches can be deactivated; however, firmware changes that are applied with CDU cannot be deactivated.

The ELF loader continues with the regular concurrent patch process after the new static variables are allocated and the symbol and relocation tables are updated accordingly. Function pointers are recalculated using standard relocation methods of the concurrent patch feature. Finally, the ELF loader sets a new event indicator, CDU i390 switch complete, on all PUs before the switch to the new i390 code load is initiated.

The new i390 code is activated by restarting the i390 environment. The CDU i390 switch complete event is given highest priority, and an ELF loader routine is dispatched that initializes all new static variables. This design ensures that any new i390 code can access the new static data variables only after they are initialized.

Completion phase

Even with a CDU activation success message, additional actions might be required to make all CDU functions available or to completely transition all firmware subsystems to release n+1 code levels. The next section presents the details of determining whether all release n+1 functions are enabled and available. Additionally, on System z9 and z10 there are currently a few channel subsystems, cryptographic coprocessor subsystems, and CFCC functions that require an additional action to migrate their code to a new release. As part of the CDU activation process, a hardware message is potentially displayed directing the user to invoke two tasks to complete the new release transition for those exception firmware subsystems:

- ▶ The Query Channel/Crypto Config Off Pending task allows the user to see which channels and crypto features must be configured offline and put back online to get the new release loaded. If the system does not have any of the exception channel or crypto hardware installed, no action by the user is required.
- ▶ The second potential task that the user can be asked to invoke is the Query Coupling Facility Reactivations task. This task informs the user whether any coupling facility (CF) partitions must be reactivated to move the CFCC code for that partition to the new release.

After the CDU activate is complete, the user can invoke the Query function availability from the last activate option from the CDU option panel, Figure 11-8 on page 242, to obtain an overview of which functions are not yet available or are not yet enabled after the CDU completion step. This panel addresses the exception cases in which one or more new release functions cannot be made available during the CDU activation process.

There are two possible reasons for the occurrence of exception cases. The first is that not all subsystems have transitioned to the release n+1 code level. The user should use the Query Channel/Crypto Config Off Pending and Query Coupling Facility Reactivations tasks to ensure that the new code-level transitions completed. After this new code-level validation is completed, the user should invoke the CDU function availability option to see what, if any, functions are still not yet available. The second reason for exceptions is that one or more functions might require some additional action to enable them and make them available, which could range anywhere from having to configure certain types of channels or processors off and on to having to reactivate the system, including a potential update to the input/output configuration data set (IOCDs).

11.8 Problems and discussions

1. Describe the different ways that an HMC or SE applies updated firmware.
2. What is the reason for an alternate SE? How is it loaded? How is kept in synch?
3. What is cloning? What sub-tasks are performed by cloning?
4. What is Concurrent Driver Upgrade? Explain the prerequisites for concurrent driver upgrade? What sub-tasks are performed by concurrent driver upgrade?

Archived

Input / Output

On most PCs today, if you want to attach an I/O device to it, such as a hard disk or CDROM, you plug one end of a USB cable into the PC's USB port and the other end into the USB device. The operating system automatically configures the, and you have access to the device at speeds up to 60 Mbps, and you could have 127 devices on that same port. At a very high level this same process applies to a System z, but many details are different. Also, System z I/O architecture has changed somewhat through the years.

In this chapter, we review the various types of I/O that can be attached to a System z, how they are plugged into the I/O cages, how they are made known to the operating systems, and how they are configured and monitored from the SE/HMC.

After studying the material in this chapter you should understand:

- ▶ Much of the terminology that is used when discussing System z I/O.
- ▶ The differences in the channel types that are available and the technological progression (history) involved.
- ▶ A general idea of the logical structure of System z I/O and the constraints that are involved.

12.1 Parallel channels

Parallel channels were one of the earliest zSeries channel types. In fact it was used on the early system/360 machines that were announced in 1964.

The parallel I/O interface uses two cables called bus and tag, as shown in Figure 12-1 on page 248. A bus cable carries information (one byte in each direction), and a tag cable indicates the meaning of the information about the bus cable. Each cable itself consists of a bundle of shielded copper coaxial cables with a grounded shield. Bus and tag cables have 48 contacts per connector (signal and ground), which means that each control unit (CU) connection on the channel interface has 192 contacts. Control units can be chained on parallel connections, also known as daisy chaining. The architectural limit to the number of control units in a chain is 256, but the electrical signal power requirements restrict this to a

practical maximum of eight control units on a channel. The connection ends with terminator blocks in the last CU.

The maximum data rate of the parallel channel is up to 4.5 MBps when in streaming mode, and the maximum distance that can be achieved with a parallel channel interface is up to 122 meters (400 feet). These specifications can be limited by the connected control units and devices. A parallel channel can have up to 256 devices on it.

The 400 foot limit is acceptable for a raised floor environment but not as convenient for a distributed environment. Currently there is no Parallel Channel interface card that you can plug into the later System z to connect to parallel devices. To connect parallel channel devices, you must use an ESCON channel with an ESCON converter. However the ESCON converter marketed by IBM is no longer available, so Parallel Channel support on current z System machines is very limited.



Figure 12-1 Bus and tag cable with connectors

12.2 ESCON channels

In the 1990s, the ESCON protocol was introduced for I/O connectivity using fiber optic cables and including a “switched” topology for data traffic.

ESCON provides bi-directional serial bit transmission in which the communication protocol is implemented through sequences of special control characters and through formatted frames of characters. ESCON utilizes fiber optic cables for data transmission. The ESCON link data rate is 200 megabits per second (Mbps), which results in a maximum aggregate data rate of up to 17 megabytes per second (Mbps). The maximum unrepeatable distance of an ESCON link using multimode fiber optic cables is 3 km (1.86 miles) when using 62.5 micron fiber. Longer distances are possible using repeaters, switches, channel extenders, and Wavelength Division Multiplexing (WDM). The maximum number of devices you can have an ESCON channel is 1024.

12.3 FICON channels

Based on the Fibre Channel Protocol (FCP), FICON was introduced in 1998. Because it is much faster than ESCON, it is becoming the most popular connection type.

FICON provides all the strengths of ESCON while increasing the link rate from 20 MBps (200 Mbps) all the way up to 1 Gbps on the FICON Express, 2 Gbps on FICON Express2 features, and 4 Gbps for FICON Express4 features, as well as increasing the extended (non-droop) distance from 9 km to 100 km. Also up to 16K devices can be attached to one FICON channel. The FICON implementation enables full duplex data transfer, so data travels in both directions simultaneously, rather than the half-duplex data transfer of the ESCON implementation. Also, FICON enables multiple concurrent I/O operations rather than the single-sequential I/O operation of ESCON. The FICON channel matches data storage/access requirements and the latest technology in servers, control units, and storage devices. FICON channels allow faster and more efficient data transfer, while at the same time allowing you to use their currently installed single mode and multimode fiber optic cabling plant.

FICON utilizes long wavelength (LX) and short wavelength (SX) transceivers with multimode and single mode fiber optic media for data transmission. FICON and ESCON cables are shown in Figure 12-2.



Figure 12-2 ESCON (right) and FICON (left) cables

12.4 Channel-to-channel

Sometimes instead of sending your data to an I/O device, it would be nice to send it to another System z. Examples of users include the z/VM Pass-Through Facility, TCP/IP connections, and basic sysplex operation for z/OS. Channel-to-channel (CTC) channels provide this capability.

The channel-to-channel function simulates an input/output (I/O) device that can be used by one system control program to communicate with another system control program. It provides the data path and synchronization for data transfer between two channels. When the CTC option is used to connect two channels that are associated with different systems, a *loosely coupled* multiprocessing system is established. The CTC connection, as viewed by either of the channels it connects, has the appearance of an unshared input/output device. The CTC is

selected and responds in the same manner as any I/O device. It differs from other I/O devices in that it uses commands to open a path between the two channels it connects and then synchronizes the operations that are performed between the two channels.

Early CTC implementations were done by two plugging parallel channels into a separate external box called a control unit. The control unit looks like an I/O device to the parallel channels. So while the parallel channels think they are reading and writing to a device, they were really reading and writing to the other system.

When ESCON channels came along, the need for the external control unit was eliminated. One of the two connected ESCON channels emulates the control unit protocol. The other ESCON channel would still do normal device I/O commands and think it was talking to an I/O device. FICON supports CTC in much the same way except with FICON the channels negotiate which end of the connection will perform the control unit function, with ESCON the user defines this.

CTC support exists for:

- ▶ ESCON channels
- ▶ FICON channels using ESCON Director with the FICON Bridge feature
- ▶ FICON Native CTC (FCTC)

CTC control units and associated devices provide the infrastructure for intersystem communication.

12.5 OSA channels

With the popularity of networks, it was important that the System z be able to connect to them and provide the System z qualities of service and I/O bandwidth that the large mainframe customers expect. This ability is provided by the Open System Adapter (OSA) cards. The Open Systems Adapter-Express3 (OSA-Express3), the OSA-Express2, and the OSA-Express comprise a number of integrated hardware features that can be installed in a System z I/O cage, becoming integral components of the server's I/O subsystems, and support various network protocols, such as Ethernet, token-ring, and the Fiber Distributed Data Interface (FDDI). They provide high function, connectivity, bandwidth, data throughput, network availability, reliability, and recovery. As mentioned before, a common use of OSA cards is to connect the System z to Ethernet networks. Current connection rates up to 10 Gigabit per second are available with an unrepeatable maximum distance of 10 km (6.2 miles), and are achieved using a 9 micron single mode fiber optic cable.

Another common use of OSA cards is as an Integrated Console Controller (OSA-ICC) that allows OSA-Express3, OSA-Express2, and OSA-Express 1000BASE-T Ethernet cards to support Ethernet-attached TN3270E consoles. The OSA-ICC can provide a system console function at IPL time and usage by multiple logical partitions. Console support can be used by z/OS, z/VM, z/VSE, z/TPF, and TPF. The OSA-ICC function also supports local non-SNA DFT 3270 and 328x printer emulation for TSO/E, CICS®, IMS™, or any other 3270 application that communicates through VTAM®.¹ OSA-ICC is configured on a port-by-port basis and each port can support up to 120 console session connections. Each port has an RJ-45 receptacle for cabling to an Ethernet switch that is appropriate for the LAN speed (up to 1000 Megabits per second). The RJ-45 receptacle is required to be attached using EIA/TIA category 5 unshielded twisted pair (UTP) cable with a maximum length of 100 meters (328 feet).

¹ This statement is based on typical z/OS usage. For other environments, OSA-ICC generally provides whatever functions were originally implemented for local 3270 devices.

12.5.1 Internal OSA channels

The OSA interfaces that we described are used to send data between two machines, but how would you send data between two logical partitions on the same machine? One way to do this is to place an OSA channel in each partition and connect the two channels. This implementation has a few drawbacks, for instance, it requires a piece of hardware (an OSA card) that is just being used as a passthru device. Furthermore, there are delays in the hardware, which slows down the communication rate.

A better way to do this is to use a channel function known as a *Hipersocket*.

HiperSockets™ provide the fastest TCP/IP communication between consolidated Linux, z/VM, z/VSE, and z/OS virtual servers in a System z server. HiperSockets provide internal *virtual* local area networks that act like TCP/IP networks within the System z server. This integrated Licensed Internal Code (LIC) function, coupled with supporting operating system device drivers, establishes a higher level of network availability, security, simplicity, performance, and cost effectiveness than is available when connecting single servers or logical partitions together using an external TCP/IP network.

HiperSockets eliminate the need to utilize I/O subsystem operations and the need to traverse an external network connection to communicate between logical partitions in the same System z server. You do not even need an OSA card. HiperSockets eliminate the need for any physical cabling or external networking connections between these virtual servers and can free up system and network resources.

The communication between virtual servers (LPARs) is through I/O queues that are set up in the system memory of the System z server. Traffic between the virtual servers is passed at memory speeds. HiperSockets are customizable to accommodate varying traffic sizes.

12.6 Plugging cards in the machine

We covered the basic types of I/O cards, and now we discuss how they are physically attached to the system. Each System z has one or more I/O cages with slots to hold I/O cards. Each slot in the I/O cage is assigned a Physical Channel Path Identifier (PCHID) number. Table 12-1 shows the slots and corresponding PCHID mapping for a System z I/O cage. The PCHID number becomes important when we generate the I/O Configuration data set (IOCDs).

Note: The current largest System z machine can have up to 3 I/O Cages in which to plug in the I/O cards and this is why there are three cages in the table.

Table 12-1 Plugging configuration

I/O Cage Slot	PCHID Range		
	Cage 1 Bottom “A”	Cage 2 Bottom “Z”	Cage 3 Top “Z”
1	100 - 10F	300 - 30F	500 - 50F
2	110 - 11F	310 - 31F	510 - 51F
3	120 - 12F	320 - 32F	520 - 52F
4	130 - 13F	330 - 33F	530 - 53F

6	140 - 14F	340 - 34F	540 - 54F
7	150 - 15F	350 - 35F	550 - 55F
8	160 - 16F	360 - 36F	560 - 56F
9	170 - 17F	370 - 37F	570 - 57F
10	180 - 18F	380 - 38F	580 - 58F
11	190 - 19F	390 - 39F	590 - 59F
12	1A0 - 1AF	3A0 - 3AF	5A0 - 5AF
13	1B0 - 1BF	3B0 - 3BF	5B0 - 5BF
15	1C0 - 1CF	3C0 - 3CF	5C0 - 5CF
16	1D0 - 1DF	3D0 - 3DF	5D0 - 5DF
17	1E0 - 1EF	3E0 - 3EF	5E0 - 5EF
18	1F0 - 1FF	3F0 - 3FF	5F0 - 5FF
19	200 - 20F	400 - 40F	600 - 60F
20	210 - 21F	410 - 41F	610 - 61F
21	220 - 22F	420 - 42F	620 - 62F
22	230 - 23F	430 - 43F	630 - 63F
24	240 - 24F	440 - 44F	640 - 64F
25	250 - 25F	450 - 45F	650 - 65F
26	260 - 26F	460 - 46F	660 - 66F
27	270 - 27F	470 - 47F	670 - 67F
29	280 - 28F	480 - 48F	680 - 68F
30	290 - 29F	490 - 49F	690 - 69F
31	2A0 - 2AF	4A0 - 4AF	6A0 - 6AF
32	2B0 - 2BF	4B0 - 4BF	6B0 - 6BF

It is tempting to simply plug all of the I/O cards in the first set of slots, but that is not a wise thing to do because an I/O cage is connected to the processor(s) by eight cables, as shown by the A, B, C, letters in Figure 12-3 on page 253.

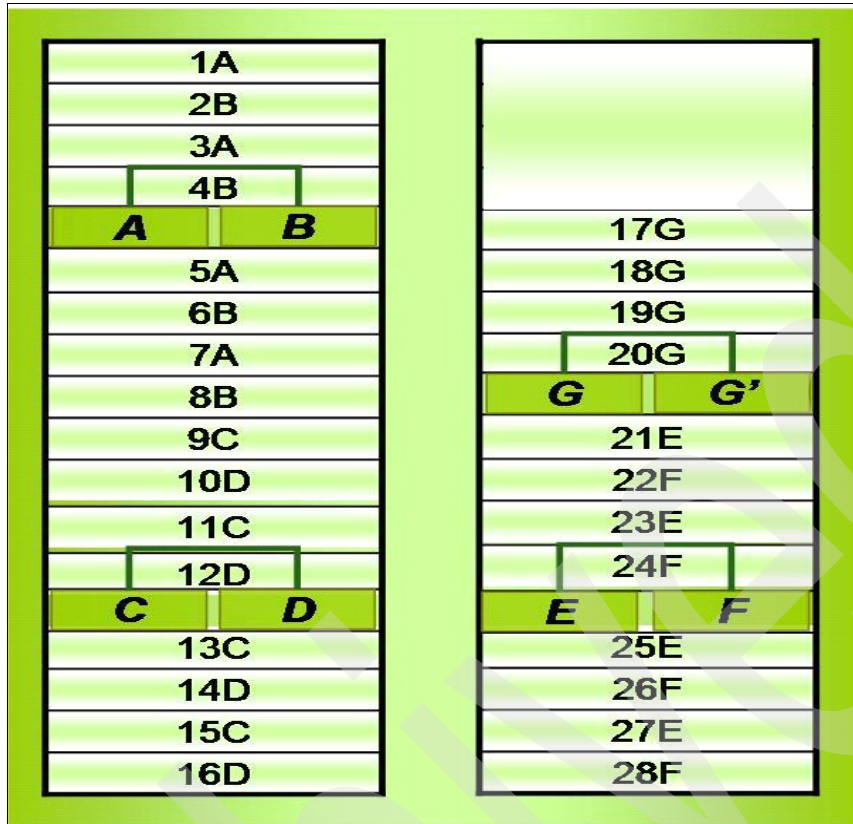


Figure 12-3 I/O cage cables and slot numbers

In Figure 12-3, the data cables attach to the I/O cage at the points with the letters on them and feeds the cards in the slots with their letter on it, for example, data cable A feeds card slots 1A, 3A, 5A, and 7A. This data link has a maximum data rate of 6 GBps. So if all the cards were plugged into just these four slots, they could exceed the data rate of the link, which causes it to become a bottleneck, thus slowing down the data going to and from all of the I/O cards being driven by it.

This is where load balancing comes in. You plug some of your cards so that they are fed by data cable A, some fed by data cable B, and so on. This way fewer cards are competing for the 6 GBps of data being sent down these data links.

Another benefit of spreading the cards in the cage is reliability. If, for some reason, data link A were to fail, cards that were attached by data link B would still be operational and could continue to access I/O that was unavailable using data link B.

One more thing that you will hear about a channel is its CHPID number, which is what the software uses to talk to the channel. There is a mapping of the CHPID to PCHID number that we talk about when we get to an IOCDs generation later in this chapter.

12.6.1 I/O system control units and devices

Thus far, we have I/O cables connected to I/O cards that are plugged into I/O cages. Now we turn to what is attached to the ends of I/O cables to complete the I/O system.

The central processor contains the processors, memory, control circuits, and interfaces for channels. A channel provides a path between I/O devices and memory. Early systems had up to 16 channels. In contrast, modern mainframes can have many channels.

Channels connect to *control units*.² A control unit (CU) contains the logic to work with a particular type of I/O device. A control unit for a printer, for example, has much different internal circuitry and logic than a control unit for a tape drive. Some control units can have multiple channel connections, providing multiple paths to the control unit and its devices.

Control units connect to devices, such as disk drives, tape drives, communication interfaces, and so forth. The division of circuitry and logic between a control unit and its devices is not defined, but it is usually more economical to place most of the circuitry in the control unit.

Figure 12-4 presents a conceptual diagram of a mainframe system. Note that current systems are *not* connected, as shown in Figure 12-4. However, this diagram helps to explain the background terminology that permeates mainframe discussions. Modern mainframes are descended from the architecture that is presented in Figure 12-4 and retain some of the design shown. Because modern mainframes retain backward compatibility with their predecessors, this early design warrants further examination.

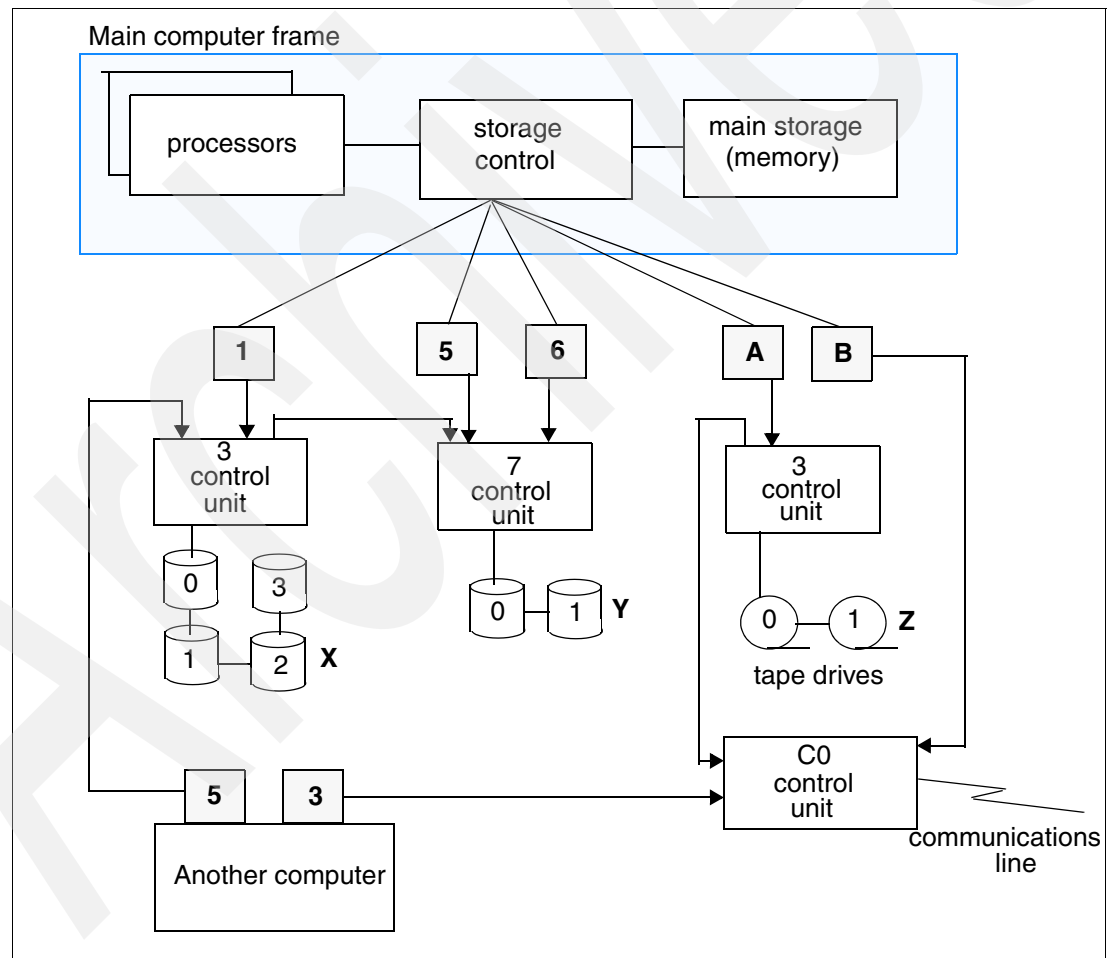


Figure 12-4 Conceptual early mainframe

² This is not always visible. An OSA channel, for example, has the control unit and device function integrated into one physical unit.

The storage control shown in Figure 12-4 on page 254 permits the channels to directly access system memory. Control units can connect to multiple devices. The maximum number of devices depends on the particular control unit.

Each channel, control unit, and device has an address that is expressed as a hexadecimal number. The disk drive marked with an X in Figure 12-4 on page 254 has address 132, which is derived, as shown in Figure 12-5.

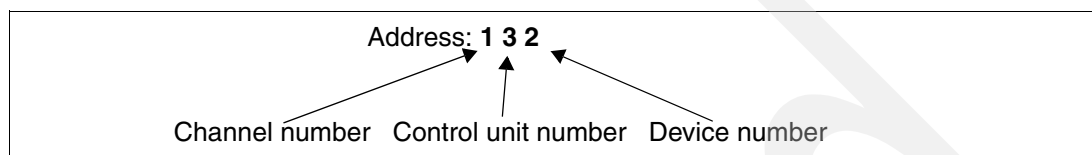


Figure 12-5 Device address in early mainframes

The disk drive marked with a Y in Figure 12-4 on page 254 can be addressed as 171, 571, or 671 because it is connected through three channels. By convention, the device is known by its lowest address (171), but the operating system can use all three addresses to access the disk drive.

Multiple paths to a device are useful for performance and for availability. In the conceptual system in Figure 12-4 on page 254, when an application wants to access disk 171, the system first tries channel 1. If it is busy (or not available), it tries channel 5 and so forth.

The figure also contains another S/360 system with two channels connected to control units that the first system uses. This sharing of I/O devices is common in all mainframe installations. Tape drive Z is address A31 for the first system, but is address 331 for the second system.

Sharing devices, especially disk drives, is not a simple topic, and there are hardware and software techniques that the operating system uses to control exposures, such as updating the same disk data at the same time from two independent systems.

Note that the address of a device in this earlier architecture, specifies a hardware path. This relationship is not present in modern systems. With System z, each device has a device number that is assigned through software (in the IOCDS). There are also hardware-related addresses, such as control unit numbers and device numbers within a control unit, that are needed to specify the path to a device. However, mainframe workers frequently say and write address when they really mean device number. This terminology is carried over from the earlier architecture. If someone says, “We need to IPL the system from address A80,” they almost certainly mean device number A80. We might hear, “It is drive 15 in control unit 02 on CHPID C1,” but the discussion is about hardware addresses.

In general, the context of a discussion quickly determines whether address really means device number or a hardware address.

12.6.2 I/O system switches and directors

An ESCON director, FICON director, or switch is a sophisticated device that can sustain high-data rates through many connections. A large director might have 200 connections, for example, and all of them can pass data at the same time.

The difference between a director and a switch is that a director is somewhat more sophisticated and contains extra functionality, such as built-in redundancy for maximum fault tolerance. The director or switch must keep track of which CHPID (and partition) initiated

which I/O operation, so that data and status information is returned to the right place. Multiple I/O requests, from multiple CHPIDs attached to multiple partitions on multiple systems, can be in progress through a single control unit.

Figure 12-6 shows a more recent system configuration.

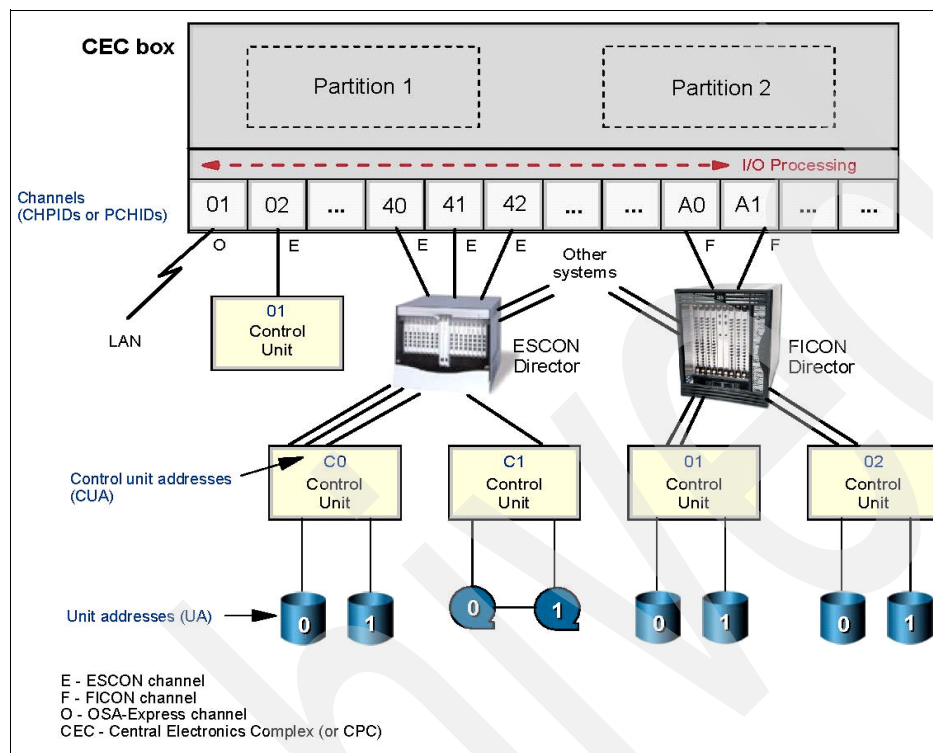


Figure 12-6 More recent system configuration

12.7 Logical devices and logical partitions

In the early System 360/370 days (1960-1970) the channels, control units and devices were all fixed by hardware assignments. Addressing a device took three nibbles. The first nibble was the channel number to the device, the second nibble was the control unit address to that device, and the third was the unit address of the device. These nibbles were fine as long as you were not changing your configuration frequently; however, if you did, then you had to update your device addresses in your programs to reflect this new change. It also limited you to 16 channels, with 16 control units on each channel, and 16 devices per control unit.

Then in 1982, extended architecture (XA) was introduced, which turned the physical device address into a logical device. XA also expanded the addressing to two bytes (four nibbles), allowing a maximum of 64K devices rather than 4K. A program known as Input Output Configuration Program (IOCP) allowed the customer to set up a mapping between the physical device address and its logical device number. Now, if you changed your configuration, you did not have to update the device number in your programs. You could just update the mapping to these new device numbers.

In 1988, another change took place. Up until then, only one operating system at a time could run on a mainframe. But with the development of faster processors it became apparent that this was a waste of processor resources. So logical partitions (LPARs) were created in which operating systems could be run. The logical partitions virtualized the machine resources so

that operating systems that were running in them would think that they were the only operating system running on the real machine, which allowed mainframes to run at higher processor utilization rates because when a program in one partition was waiting for I/O, execution could switch to another partition that was ready to run.

A machine (at that time) could support four logical partitions, which meant that four operating systems could run at the same time. Again the IOCP program was extended to define the logical partitions and the devices that map to those logical partitions.

Over time the number of supported logical partitions went from four to 15 to 60. But since only one byte was allocated for the channel number, only 256 channels can be defined on a machine.

12.7.1 Multiple channel subsystem: removing the 256 channel limit

As designers once thought that no one would ever need a PC with more than 640 KB of memory, early computer designers thought that no computer would ever need more than 16 and then more than 256 channels. A problem that needed to be addressed to expand this 256 channel limit was that software that customers were running for years assumed the channel number would fit in a 1 byte field and customers would not want to rewrite software to expand this field to 2 bytes (assuming they had the source for this software to begin with). The solution to this problem was multiple channel subsystem (MCSS).

In MCSS, there are four groups of 256 channels. Each group is a channel subsystem (CSS). Each CSS can in turn be configured with one to 15 logical partitions with a maximum of 30 logical partitions per server on z9 BC, z990, and z890, and up to 60 logical partitions on z10 EC and z9 EC servers. CSSs are numbered from 0 to three and are sometimes referred to as the CSS Image ID (CSSID 0, 1, 2, or 3 for the z10 EC, z9 EC and z990; and CSSID 0 and 1 for the z10 BC, z9 BC and z890).

Given the four CSSIDs and the 256 channels per CSS, addressability to 1024 channels is now possible. The programs that run in the logical partitions can still keep their 256 channel range, and the System z maps it to one of the real 1024 channels based on the partition's assigned CSSID value. In the next section, we describe how this mapping is done.

12.8 Defining the I/O to the hardware (IOCP)

Our I/O subsystem description is now complete. We now have our I/O cards plugged in the correct slots and know the PCHID numbers that they are assigned. We know what logical partitions need access to which PCHID and which CSS and CHPID. (The combination of a CSS identifier and a CHPID identifier is often written as CSS.CHPID.) All that is left is to provide the mapping between these PCHIDS and CSS.CHPIDS. To do so, we use the Input Output Configuration Program (IOCP) to generate an Input Output Configuration Data Set.

The IOCP program takes the user input and generates data structures that the System z uses to know which CSS and CHPID in which partition maps to which PCHID. This data structure also contains information about the type of this PCHID and its underlying interface. The document *System z Input/Output Configuration Program User's Guide for ICP IOCP*, which is over 300 pages long, describes these functions in great detail. For our present work, we describe a very simple version of an IOCDs to avoid becoming lost in the details.

There are five different types of statements that make up the IOCP source statements that are used to create an IOCDs. They are: ID, RESOURCE, CHPID, CNTLUNIT, and IODEVICE. The ID statement is optional; and you can code it if you want specific identification

information in the heading of IOCP configuration reports and on Support Element displays.³ The RESOURCE statement is required and defines the logical partitions and logical channel subsystems in your configuration. CHPID, CNTLUNIT, and IODEVICE statements are optional and define, respectively, channel paths, control units, and I/O devices in your configuration and the logical partitions that can access these resources.

The following general rules apply to the sequence of IOCP statements in the input file:

- ▶ The ID statement can be coded only once and must precede other IOCP statements.
- ▶ The RESOURCE statement can be coded only once and must precede all CHPID statements.
- ▶ A CHPID statement that defines a channel path must precede a CNTLUNIT statement that refers to the channel path.
- ▶ A CNTLUNIT statement must follow the CHPID statement that defines the channel path to which the control unit is attached.
- ▶ An IODEVICE statement must follow the CNTLUNIT statement that defines the control unit to which the device is assigned.
- ▶ IOCP comment cards (specified by an asterisk (*) in column 1 or by .* in columns 1 and 2) can be inserted in the input file where desired.

Suppose you want to define a FICON channel installed in the hardware PCHID slot 121 for use in logical partition LP01 that is using CSS0 and is to be accessed through CHPID 23, as shown in Figure 12-7.

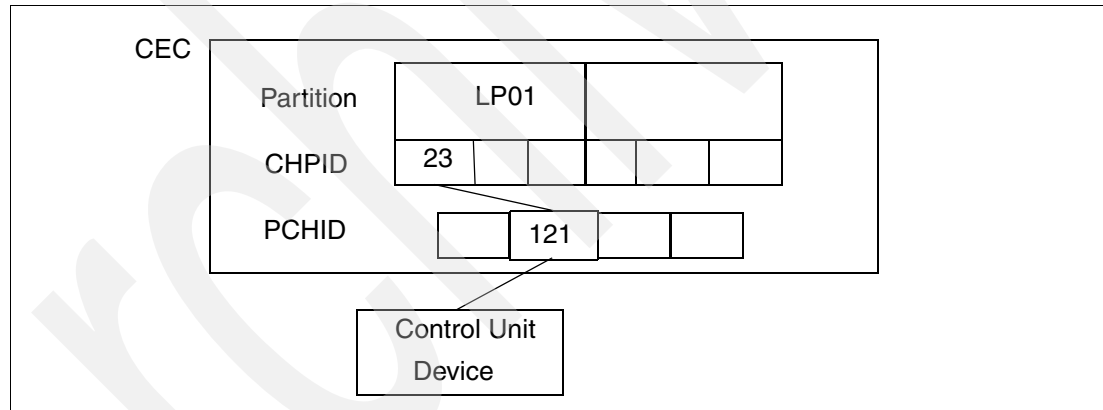


Figure 12-7 Simple configuration

Example 12-1 shows the IOCP source file that you can use for this trivial configuration.

Example 12-1 IOCP source file

ID	MSG1='Starter IOCDS example', MSG2='Created August 2009'	+
*		
RESOURCE	PARTITION=((CSS(0),(LP01,1))	
*		
CHPID	PCHID=121,PATH=(CSS(0),23),TYPE=FC,PART=LP01	
*		
CNTLUNIT	CUNUMBR=0001,PATH=(CSS(0),23),UNITADD=((00,64)),UNIT=2105, CUADD=0	+
*		

³ For other reasons, beyond the scope of this discussion, an ID statement is almost always present.

IODEVICE ADDRESS=(0100,64),CUNUMBR=0001,UNIT=3390B

A few comments might help in reading Example 12-1 on page 258. The IOCP statement format corresponds to an early S/360 assembly language format. A statement is continued by placing a character in column 71 (the + symbols in this example). Example 12-1 on page 258 actually defines 64 disk drives (type 3390B). The device numbers of the drives are 0100, 0101,...,013F since device numbers are expressed in hexadecimal, and many people incorrectly refer to these as the addresses. Elements of the actual hardware address are the UNITADD, CUADD, and PCHID values.

In a classroom environment, you might enter this source code, build an IOCDS, and IML with it. If your channel then indicates Online Operating, you are now an IOCDS expert. Figure 12-8 shows the Support Element's Channel Work Area display.

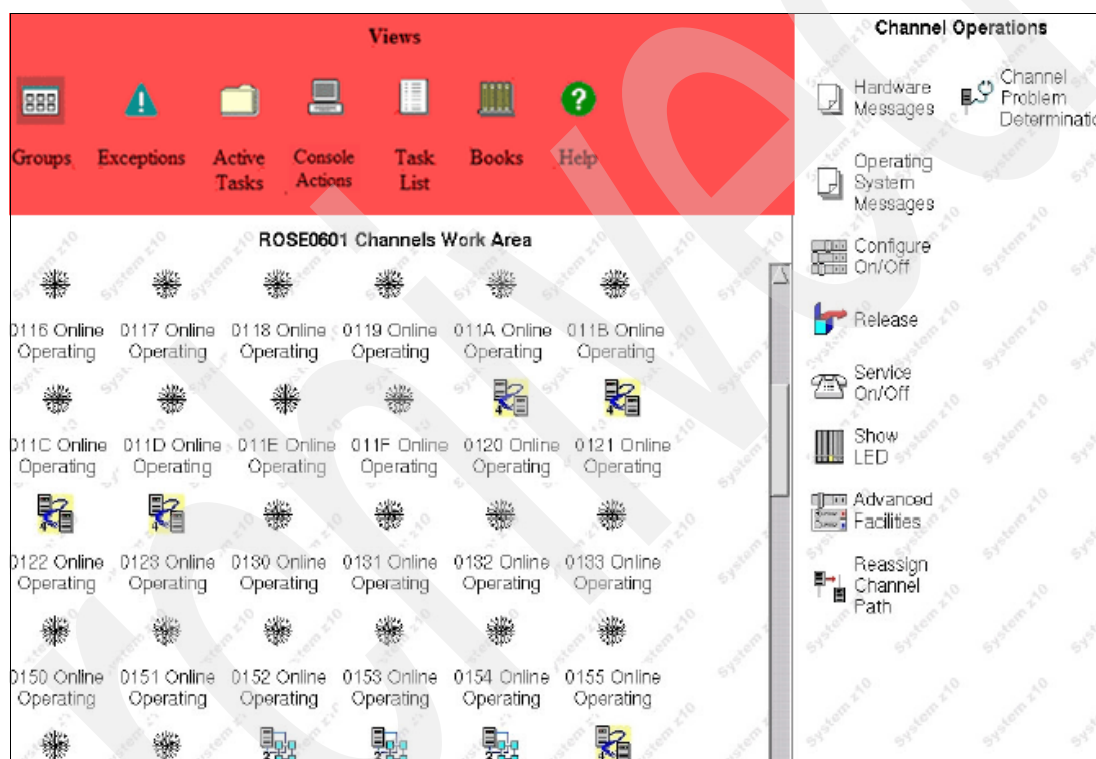


Figure 12-8 Channel work area display

You can see the verify the CSS.CHPID mapping by double-clicking the PCHID icon, which should produce Figure 12-9 on page 260.

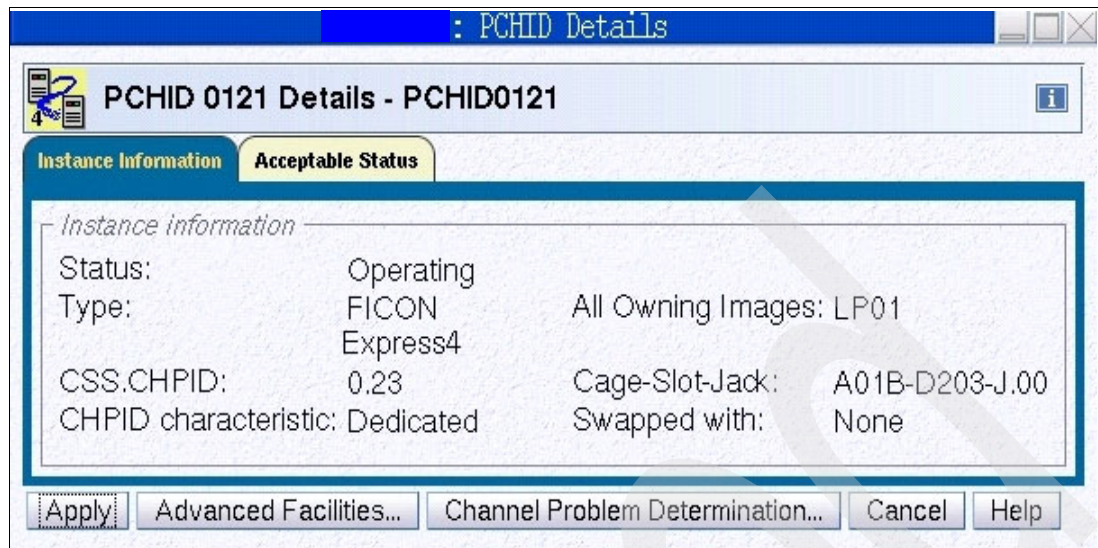


Figure 12-9 PCHID assignment details

12.8.1 Current system I/O specifications

Detailed I/O specifications for a large mainframe can be complex. In the following list, we provide higher-level information for an IBM System z10 (machine type 2097) as of February 2008:

- ▶ The total system has a maximum I/O bandwidth of 288 GBps.
- ▶ It can support 28 I/O cards in each of three I/O cages for a total of 84 I/O cards.
- ▶ Depending on the nature of the card, an I/O card can provide from one to 16 channels:
 - Up to 69 ESCON cards that provide up to 1024 channels.
 - Up to 84 FICON cards that provide up to 335 channels.
 - Up to 24 OSA cards that provide up to 48 channels with two network ports per channel.
 - Up to 12 ISC cards that provide up to 48 coupling channels.
 - Up to eight cryptographic cards that provide up to 16 cryptographic engines. (These cards have no external I/O connectivity).
- ▶ Each I/O cage can be attached to the processor by seven 6 GBps data links for a total of 42 GBps per cage.
- ▶ It can support up to 1024 channels
- ▶ It can support up to 523,264 I/O devices, but it depends on device addressing arithmetic, rather than a practical limitation.

It has logic so that if one of the data links to an I/O cage fails, it automatically switches to an alternate path, and the software is unaware of the failure or recovery action. Such failures create a call home notification, and repairs are usually accomplished without the customer being aware that any problem existed.

Figure 12-10 on page 261 provides one view of the System z I/O structure. The IFB-MP elements perform the automatic switching of data paths that we just described.

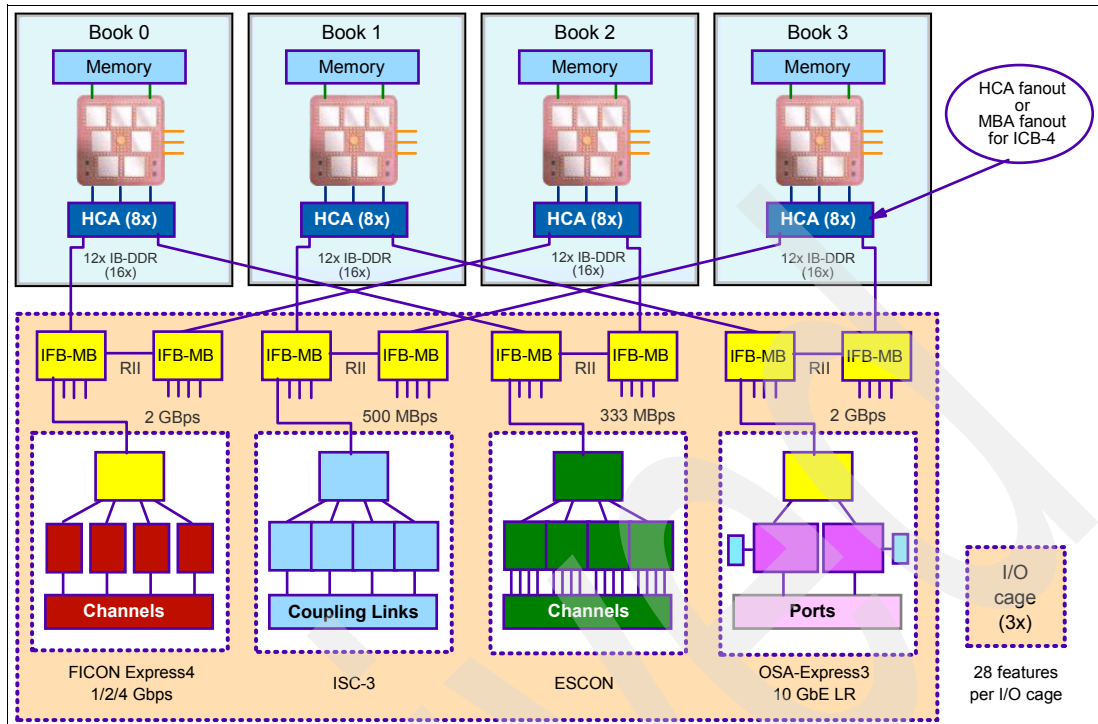


Figure 12-10 IBM System z10 EC I/O infrastructure

12.9 Working with the channels

Usually the channels do their job and get data to and from devices and processors without problems. In this case, you never need to do any of the actions that are described in the following sections. However, sometimes human intervention is required to get things back on track. We describe here some of the common actions that you can use to diagnose problems.

If a channel is definitely causing problems, you can simply configure it off line, which prevents the software from using it.

Note: In a well-configured system, there are alternate channel paths to almost all devices. Taking a channel offline might have a minor performance impact, but seldom prevents applications from running.

After the problem is fixed, the channel can be placed online again, and the operating system software will start using it again.

To change the state of a channel, the appropriate channel is first selected from the general SE channel work area display. Figure 12-11 on page 262 shows the panel to use to alter the state of a channel.

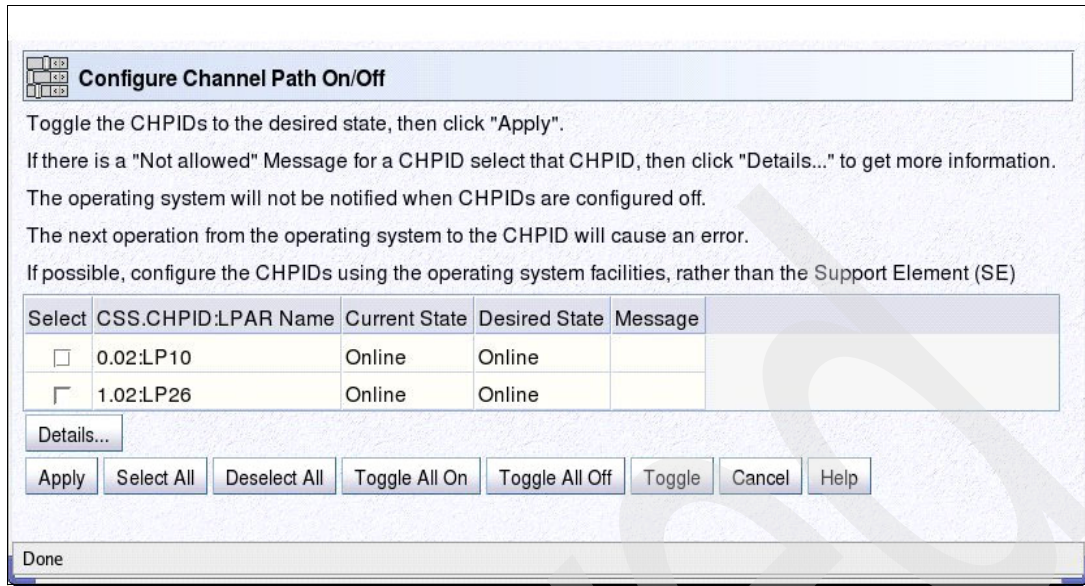


Figure 12-11 Configure Channel Path On/Off

A channel is defined by the CSS, CHPID, and LPAR name that is associated with it rather than by the absolute PCHID name. To change the state, select the channel (or several channels), select **Toggle**, and then **Apply**. As noted, taking a channel offline in this manner makes it immediately unavailable to the operating system. If the operating system tries to continue using it, it will have errors that might require operator action (by the operator of the operating system). As a general rule, it is better to use operating system commands to vary a channel inactive (from the viewpoint of the operating system) before taking it offline, in a hardware sense.

12.9.1 Advanced facilities

You might want to see what a channel is doing rather than simply taking it offline. The Advanced Facilities allows you to set up and monitor channels. You do this by dropping a channel icon from the Channel Work Area (Figure 12-8 on page 259) onto the Advanced Facilities icon under Channel Operations. What advanced facilities are provided depends on the channel type.

In the images in Figure 12-12 on page 263 through Figure 12-14 on page 264, the internal counters of an OSA Gigabit Ethernet channel are displayed, which shows such information as the packet counts that the OSA transmitted and received, which is very useful information when working with a LAN problem. You get to this panel by selecting **Card Specific Advanced Facilities** → **View Port Parms** on the selection menus.

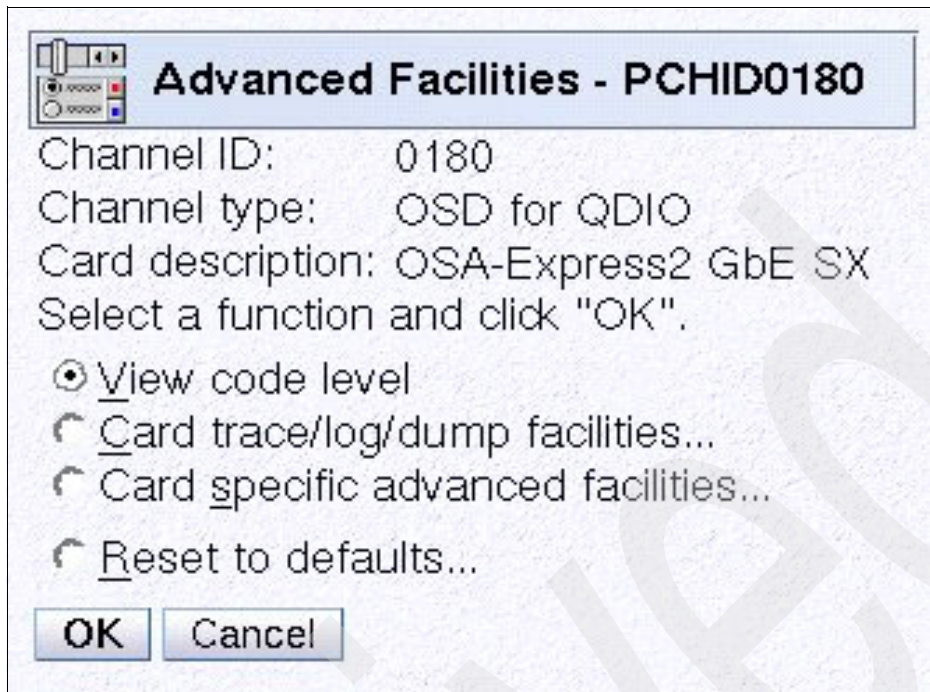


Figure 12-12 Advanced Facilities panel

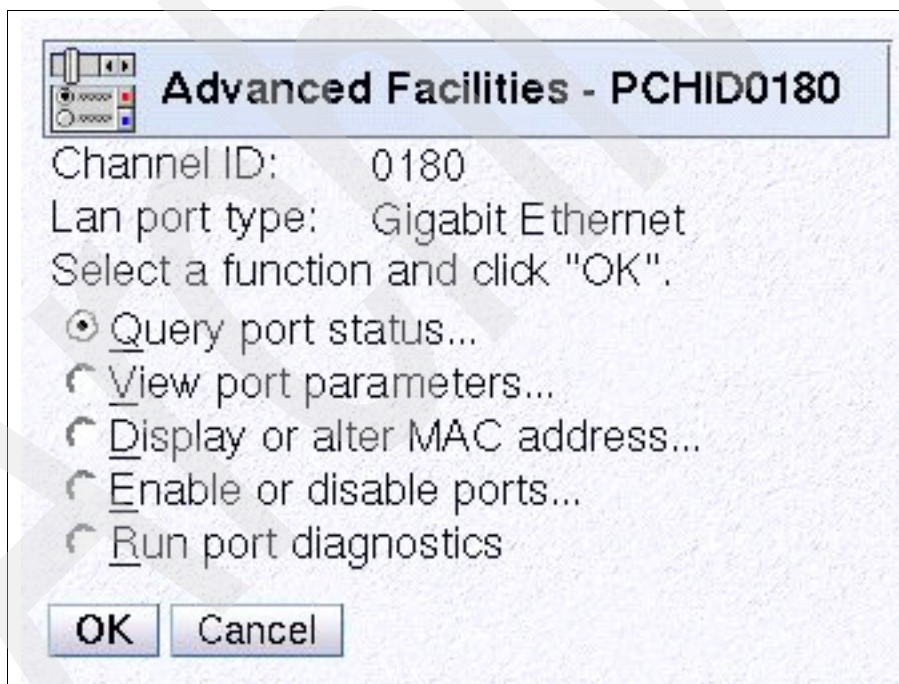


Figure 12-13 Advanced Facilities - 2

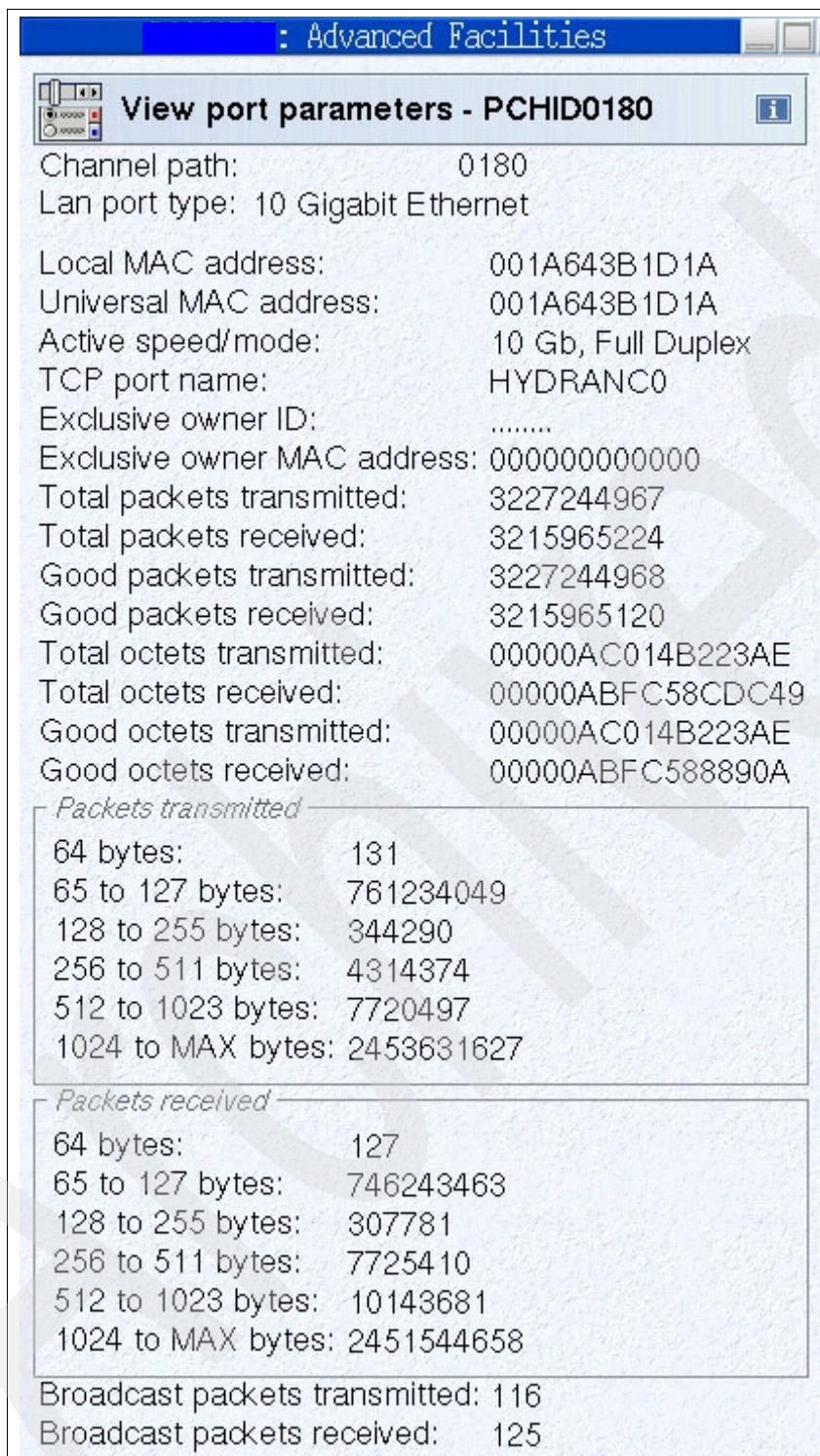


Figure 12-14 Advanced Facilities - 3

Using Advanced Facilities you can also change the mode/speed of the ports on an Ethernet card in the system. The panel in Figure 12-15 on page 265 is displayed to do that.

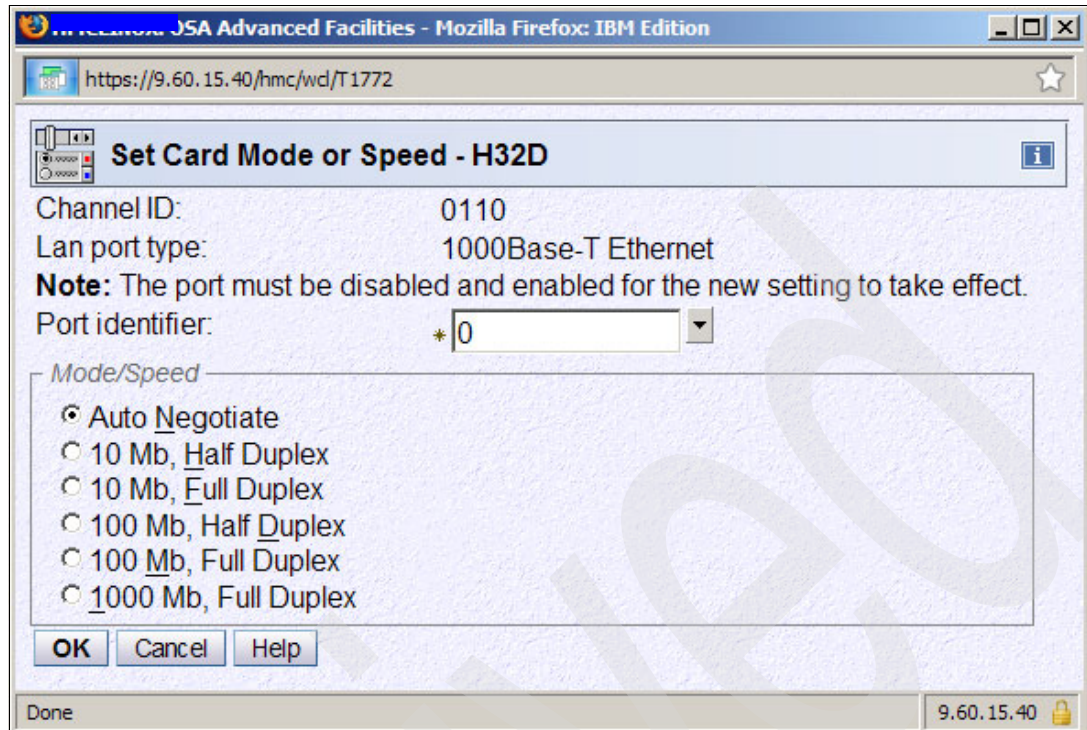


Figure 12-15 Advanced Facilities - Mozilla

Finally if you are having problems with your channel, for example, no data being sent or received, you can run a built-in diagnostic routine to have the card check itself out. Figure 12-16 is an example of that panel.

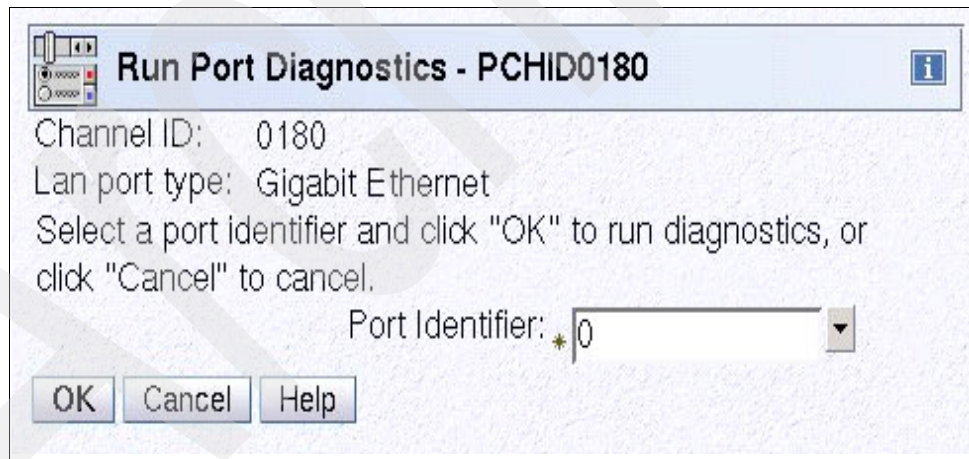


Figure 12-16 Run port diagnostics-1

The results appear as in Figure 12-17 on page 266. The user can ignore the status words if the diagnostic worked. If it failed, IBM service personnel uses the status words to determine what the failure was.

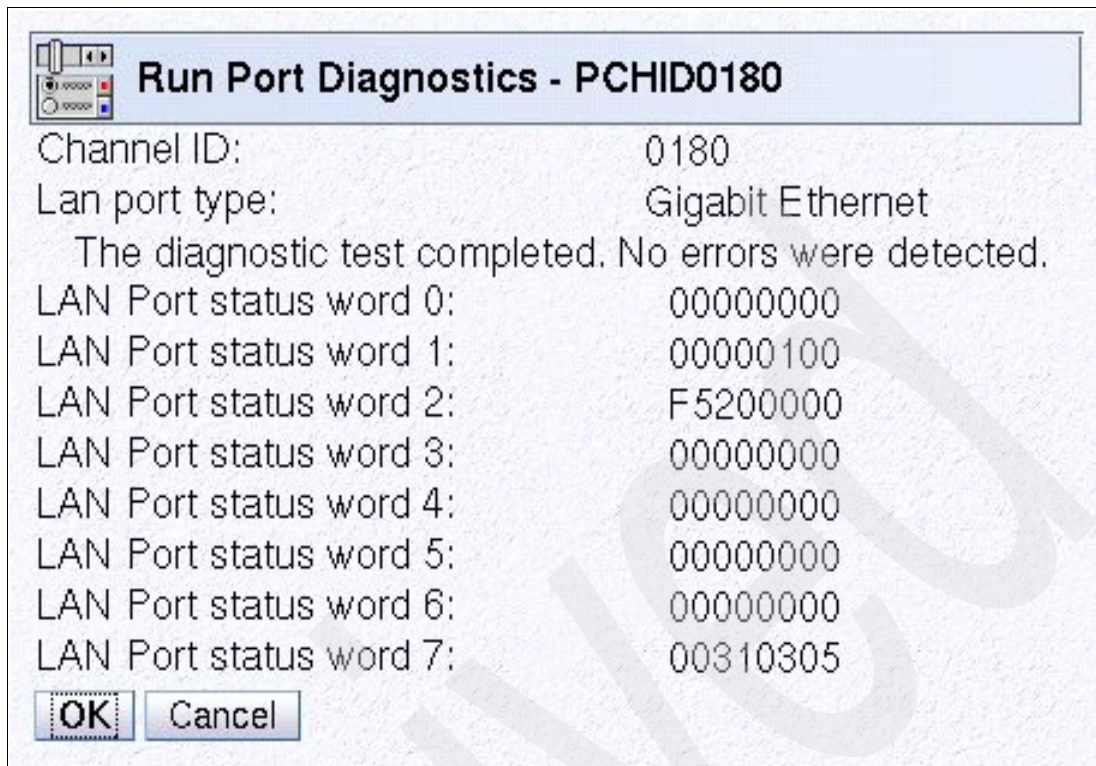


Figure 12-17 Run Port Diagnostics-2

Finally you can alter the MAC address on the card. Figure 12-18 indicates this operation.

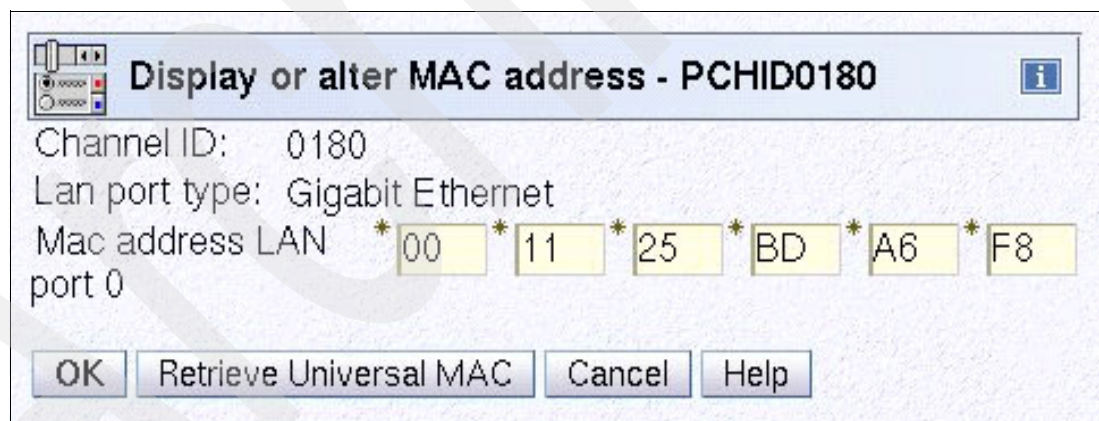


Figure 12-18 Display or alter MAC address

12.10 Conclusion

This document is intended to provide an overview of System z I/O and how it can be set up, controlled, and monitored using the HMC/SE.

There are many other books that go into greater detail on each of these subjects, for example:

- If you want to know more about the IOCP program and defining IOCDS for your machine, the *System z Input/Output Configuration Program User's Guide for ICP IOCP*, IBM order number SB10-7037-06, is an excellent place to start.

- ▶ For more details about the I/O in general, the *IBM System z Connectivity Handbook*, IBM order number SG24-5444-08, is where to look.
- ▶ If you want more details about a particular channel type, such as the OSA's adapter, the *Open Systems Adapter-Express Customer's Guide and Reference*, SA22-7935-10 is the place to look.

As you can see, the System z I/O changed greatly over the years, and it promises to continue to change in the future as new and faster interfaces and devices are developed.

For more information about the development of IBM mainframes since 1964, refer to the following Web site:

http://www-03.ibm.com/history/exhibits/mainframe/mainframe_basinfo.html

12.11 Questions and discussions

1. How many devices can you have on a parallel channel? How many devices can you have on a FICON channel?
2. Where would you find the physical location (cage-slot-jack) of a PCHID?
3. Where would you display the internal counters of an OSA channel?
4. When you configure off a CHPID, what state does it go to?
5. What is the name of the connector cable for the parallel channels?
6. What are the most common channel types on a System z today?
7. Why is a LAN adapter (an OSA) considered a System z channel?

Archived

System Activity Display

Activity monitoring is an important part of System z administration. In environments where high availability and performance is crucial, System Administrators must be able to tell if their systems are effectively using their resources or if they need more resources to accomplish their tasks. Customers can monitor resources from tools that run on the System z, such as the IBM z/OS Resource Management Facility (RMF) or other CPC-based tools. These tools have the advantage because they can access hooks in the operating systems and other software to obtain detailed metrics. They have the disadvantages in that they are a separate product and they consume valuable CPC resources when they run.

The System z hardware provides another option, System Activity Display (SAD), which using it, you can perform highly-customizable activity monitoring for CPCs, including memory, channels, and so forth, from the HMC. It shows performance metrics at the hardware level and, because it runs on the SE and HMC, it does not impact the performance of the CPC. Every 15 seconds the SE polls the CEC and collects all necessary data, which does not affect performance on the CEC because dedicated monitoring hardware is involved. Performance might be affected on the SE or HMC, but this is acceptable.

13.1 History

Very basic activity monitoring was available on earlier IBM mainframes as a busy light on each system's operator console, which showed when the system was in use and when it was idle. However, as systems grew larger with more channels and processors, the number of lights started to become unmanageable. With the introduction of virtualization (that is, when LPARs were introduced), it became impossible to use lights to show information about the virtualized pieces. In addition, viewing a blinking light to determine utilization was not very precise as compared to the System Activity Display functions.

13.2 System z architecture as it relates to SAD

To understand how SAD works, we first must review certain details about System z architecture that are different from other systems that you might have used.

13.2.1 Storage Keys

First, to better separate user processes, CPs have 16 different storage keys under which processes can run. A storage key is a form of memory protection. Processes with a storage key are only allowed to access memory allocated for that key. The keys range from 0 to F, hence 16 different keys. The storage keys can be used to focus troubleshooting to a particular group of processes. In addition, each of these storage keys can be in user mode (which in System z is called problem mode) or supervisor mode, and there is one state for when the processor is idle, so overall, at any time the CP can be in one of 33 states.¹ SAD can filter out or specifically monitor any of these states.

13.2.2 Specialty engines

Servers can have specialty processors in addition to CPs to reduce workload on the CPs. All z systems also have one or more dedicated I/O processors. The dedicated I/O Processor is called the System Assist Processor. SAPs are specialized I/O processors that handle the I/O processing in the system, providing a direct path from channel to memory, which allows CPs to not waste CPU instructions or cycles toward I/O operations. In a PC, there is only one type of processor. The PC processor has to perform both I/O operations and CPU instructions (real application processing). SAPs eliminate this problem and provide a significant increase in performance by allowing the CPs to focus on CPU instructions. The number of SAPs range from one on small systems to at least eight on the largest system. Customers can also purchase additional SAPs if their workload is very I/O intensive and requires more than the standard amount of SAPs included in their specific system. SAD can be used to monitor the activity of these processors and CPs.

13.2.3 LPARs

When using SAD with LPARs, be aware that LPARs can be configured in two different ways:

- ▶ An LPAR has dedicated processor(s) that can only be used by that LPAR
- ▶ An LPAR uses a shared processor pool

Each LPAR is defined to have one or more logical processors. These processors are either virtual or dedicated. Virtual processors appear to the customer application software, within the LPAR, as real processors. For dedicated processors, each logical processor must be backed by a real processor that is dedicated to it. All processors not needed for dedicated use are available in a shared processor pool. There is one shared pool for each type of processor (CPs, and each type of specialty engine). The number of shared processors for a given LPAR cannot exceed the number of that type in the pool; however, because the processors are not dedicated other LPARs can also use these shared processors. The processor pool is managed by the hypervisor and allocates the amount of time an LPAR is allowed to run on the CPU, which causes the shared processor pool to perform very efficiently by decreasing CPU idle time. When there is little to no activity on an LPAR, a dedicated processor will idle and not allow another LPAR to use that resource. The shared pool reduces idle time by allowing any LPAR, that requires CPU time, to use the processor.

If, for example, a system has 10 CPs and one LPAR asks for three dedicated CPs, and a second LPAR asks for one CP, then a total of four CPs are dedicated, which leaves six CPs in the shared pool. LPARs using shared processors could each have from one-to-six logical processors.

¹ This is 16 keys, each with two possible states, plus the wait state.

When an LPAR is configured to use a shared processor pool, a weight can be assigned. This weight is used to prioritize it in relation to other LPARs, for example, if there are three partitions with weights of 50, 30, and 20, if the pool is fully loaded, the LPARs get 50%, 30%, and 20% of the pool respectively at full load. If one LPAR is not asking for its share, the other LPARs can use the resources, for example, the 50% LPAR is idle, then the other two LPARs could have 55% and 45% of the system. Customers can also cap an LPAR's usage of the shared pool to its share. So, in the above example, if the 20% LPAR is capped, it never gets more than 20% of the pool, even if all other LPARs are idle.

One last concept to note is that each type of processor has its own pool with its own set of weights. An LPAR could get 80% of the CP pool, but allocated only 10% of a specialty engine's pool.

SAD handles LPAR activity by normalizing the usage to the LPARs share of each pool, which means that with a shared pool an LPAR's activity is indexed to the weight assigned, for example, if an LPAR is assigned a weight of 40% of the shared pool and is currently using 40%, it is using 100% of the LPAR's resource. In the same case, if the LPAR is now using 20% of the shared pool, the LPAR is only using 50% of the resource. The maximum of this percentage is not capped (not allowed to exceed) at 100%. In other words, an LPAR might be assigned a weight of 40% of the shared pool and is using 60%. The system activity display shows that the LPAR is using 150% of the resource. However, the maximum percentage that will be displayed is 999% regardless of if the actual percentage is higher than this value.

13.2.4 Channels

We cover System z I/O in greater detail in later chapters. For now, all you need to know is that each I/O device uses a channel, and each channel is identified by a CSS and a CHPID. SAD can monitor I/O usage and lists I/O devices by CHPID. Data can also be filtered by CHPID when customizing SAD profiles.

There might be times when you want to know the path to a particular device within the CPC. If you do not already know the CHPIDs for an I/O device on the CPC that you are monitoring, you can look them up using the System Input/Output Configuration Analyzer task on the HMC as shown, as shown in Figure 13-1 on page 272.

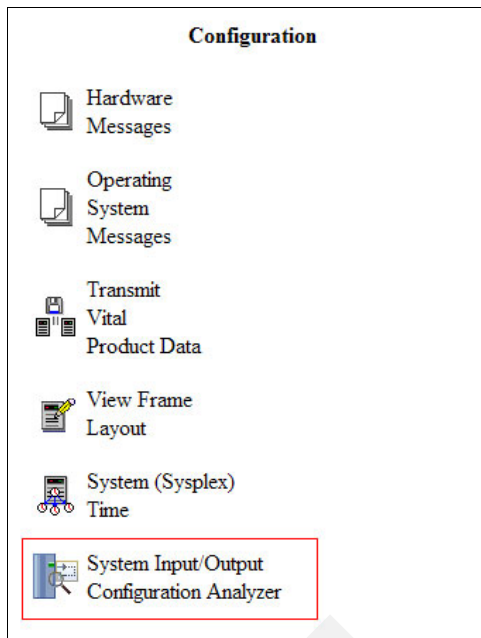


Figure 13-1 Selecting the I/O Configuration Analyzer

The result looks similar to Figure 13-2.

System Input/Output Configuration Analyzer - PCHID Control Unit View. - M05							
File View Filter Sort Help							
PCHID	CSS.CHPID	Type	Switch	Number of Control Units	Control units	2	3
0000	1.23	CBP		0			
0001	1.24	CBP		0			
0100	1.10	CFP		1	00E0		
0101	1.11	CFP		1	01E0		
0120	1.20	OSE		0			
0121	1.21	OSE		0			
0140	0.00	FC	D1	3	4800	4900	1500
0140	1.00	FC	D1	3	4800	4900	1500
0141	0.01	FC	D1	3	4800	4900	1500
0141	1.01	FC	D1	3	4800	4900	1500
0142	0.02	FC	D1	3	4800	4900	1500
0142	1.02	FC	D1	3	4800	4900	1500
0143	0.05	FC	D6	2	E480	4D00	
0143	1.05	FC	D6	1	E480		
Rows : 80							

Figure 13-2 I/O Configuration Analyzer display

13.2.5 High usage and low usage filtering

While you can monitor any specific resource, SAD also permits you to view the n-most or n-least active resources of a type, where you can specify “n”, provided it is within the range allowed by SAD. So, for example, you could set up a filter to show the 10 most-active LPARs, the five least-active CHPIDs, and so forth. For CHPIDs, these could be the most-active or least-active for the entire system or they could be for an individual LPAR.

13.2.6 Usage cases

There are many situations where using SAD can help determine issues with a server's configuration. Looking at the overall resource usage of a server can help determine whether the machine is being used effectively or if the machine needs more capacity. Looking at individual LPARs can help determine if machine resources are correctly being distributed among them, for example, you might check the activity of an individual LPAR and determine that it is limited by the CPUs that are assigned to it. If so, then you could choose to assign it more dedicated processors or purchase more processors to increase the capacity of the system. You could also decide to add more CPs to the machine either permanently or temporarily through Capacity on Demand.

Another example: In systems with very heavy I/O usage, you might determine that I/O is being limited. If this is the case, you might run a SAD profile that monitors the SAP and find that it is maxed out. So, in this case, adding more SAPs alleviates the load and improves I/O performance.

13.3 Profiles

From the user's standpoint, profiles are where SAD gains its power. Profiles allow you to customize what data SAD monitors, which gives you fine-grained control over what you see. A properly customized profile lets you focus on statistics that are important and filter out those that are unimportant.

A CPC can have many different profiles. To use a specific profile, it must first be active. A CPC can have 1-16 profiles active for use, and if more than one is active, the SAD display rotates between them. The system ships with a number of sample profiles, and the customer is free to alter all sample profiles, create new profiles, or delete all profiles but the DEFAULT profile. In Figure 13-3 on page 274, activity profiles for a CPC, we can see the list of SAD profiles for a CPC. Two profiles, DEFAULT and CHANHIGH, are active. The first 10 profiles are default ones that are included with the CPC, whereas the last three are customized.

One other thing to note is that all SAD profiles are stored on the SE. Regardless of where you set up the profiles for use (SE or HMCs), the same set of profiles are used for all places where SAD can be viewed, which means that HMC A will show the same SAD data as HMC B and that will be the same as what you would see using SAD on the SE itself. Changing the active set will of course affect all users.

Customize activity profiles for P0016F5A			
select	Profile Name	Status	Profile Description
<input checked="" type="checkbox"/>	DEFAULT	Active for HWMCA	Processing Activity AND Channel List High Use (28)
<input checked="" type="checkbox"/>	CHANHIGH	Active for HWMCA	Channel List, High Usage (49)
<input type="checkbox"/>	CHANLOW	Not active for HWMCA	Channel List, Low Usage (49)
<input type="checkbox"/>	PROCLIST	Not active for HWMCA	Processing Activity AND Channel List High Use (28)
<input type="checkbox"/>	LPARSUMA	Not active for HWMCA	LPAR Summary - 15 Partitions
<input type="checkbox"/>	LPARSUMB	Not active for HWMCA	LPAR Summary - 10 Partitions AND Channel List High
<input type="checkbox"/>	VMPROCLIST	Not active for HWMCA	For VM- Processing Activity AND Channel List High
<input type="checkbox"/>	VMPROCESSOR	Not active for HWMCA	For VM - Processing Activity(except Supervisor K3)
<input type="checkbox"/>	PROCUSAGEBYKEY	Not active for HWMCA	Central Processing Activity (by Key)
<input type="checkbox"/>	PROCESSOR	Not active for HWMCA	Processing Activity
<input type="checkbox"/>	TEST1	Not active for HWMCA	Processing Activity AND Channel List High Use (28)
<input type="checkbox"/>	LBCHANHIGH	Not active for HWMCA	Channel List, High Usage (49)
<input type="checkbox"/>	LBDEFAULT	Not active for HWMCA	Processing Activity AND Channel List High Use (28)
<input type="button" value="Customize"/> <input type="button" value="Delete"/> <input type="button" value="Change Status"/> <input type="button" value="Reset"/> <input type="button" value="Cancel"/> <input type="button" value="Help"/>			

Figure 13-3 Customize activity profiles

Available types of profile lines

A profile consists of lines, and each line monitors a specific element of the CPC and corresponds to a line on the SAD display. There are generally four kinds of lines:

- ▶ Lines that monitor individual properties.
- ▶ Lines that start a list.
- ▶ Lines that are part of a list and are automatically generated.
- ▶ Blank lines that do not monitor anything. These can be used to separate other lines into groups.
- ▶ Grid lines.

SAD profiles consist of lines and lists. A list consists of multiple rows of data. An example of a SAD profile list is the top ten highest used processors. See Figure 13-4 on page 275, Lines 1-13. Similarly, the Linux command **ps -a** is an example of a list of the processes that are currently running.

A line in a SAD profile consists of only one row of data. An example of a SAD profile line is the total utilization of all processors within the CEC. Similarly, the Linux command **ps <pid>** is an example of a line because it only consists of one row of data to display the process information. Figure 13-4 on page 275 shows part of an activity profile with two lists and a line that monitors the aggregate usage of all CPs.

Customize System Activity Profile - P0LXSM23

Object: P0LXSM23

Profile name:

Description:

Modify line options

- ☒ Change line
- ☐ Insert line
- ☐ Delete line
- ☐ Copy line to
- ☐ Move line to

Select	Line	Component	Description
<input type="radio"/>	1.	List High Use	high usage processor list (all processor types except system assist proc
<input type="radio"/>	2.	PU list	processor list line
<input type="radio"/>	3.	PU list	processor list line
<input type="radio"/>	4.	PU list	processor list line
<input type="radio"/>	5.	PU list	processor list line
<input type="radio"/>	6.	PU list	processor list line
<input type="radio"/>	7.	PU list	processor list line
<input type="radio"/>	8.	PU list	processor list line
<input type="radio"/>	9.	PU list	processor list line
<input type="radio"/>	10.	PU list	processor list line
<input type="radio"/>	11.	PU list	processor list line
<input type="radio"/>	12.	PU list	processor list line
<input type="radio"/>	13.	PU list	processor list line
<input type="radio"/>	14.	CPALL	processor ALL (all processor types except system assist processors) pr
<input type="radio"/>	15.	List High Use	high usage SAP processor list
<input type="radio"/>	16.	Sap list	SAP processor list line
<input type="radio"/>	17.	Sap list	SAP processor list line

OK Save Reset Cancel Help

Figure 13-4 Detailed view of a profile (partial)

The kinds of lists that can be displayed are:

- ▶ **Processor lists:** Display the activity of the n most active or least active CPs (excluding SAPs). Figure 13-5 on page 276 displays this consent along with LPAR lists, and non-list lines that monitor processors can either show all processors, general purpose processors, or specific specialty engines (explained earlier).
- ▶ **LPAR lists:** Displays the activity of the n most active or least active LPARs (by CP usage). See Figure 13-6 on page 276.
- ▶ **SAP lists:** Similarly, these display the n most or least active SAPs, as shown in Figure 13-7 on page 277.
- ▶ **Channel lists:** Displays the most active channels in the system, as shown in Figure 13-8 on page 277.
- ▶ **Channel lists for logical partitions:** Displays the most active channels in the selected LPAR, as shown in Figure 13-9 on page 278.

All list items can either show the n highest use items or the n lowest use items, which is chosen by the High/Low radio buttons that are available on each list's configuration panel.

Physical Processor List Options - M05

Begin list at line 8
 End list at line * 13

Processor Usage

☒ High
☐ Low

Processor Type

☒ All
☐ General purpose
☐ Integrated coupling facility
☐ Integrated facility for Linux
☐ System z application assist processor
☐ IBM System z Integrated information processor

Processor State

☒ Both problem and supervisor
☐ Problem
☐ Supervisor

Program Status Word (PSW) Key

☒ All
☐ Specific
☐ Key to exclude Key number *

Exclude Key for Processor State

☒ Both problem and supervisor
☐ Problem
☐ Supervisor

Threshold

☒ None
☐ Above
☐ Below Percentage 0

OK Reset Cancel Help

Figure 13-5 Physical Processor List Options

Logical Partition Summary List Options - M05

Begin list at line 8
 End list at line * 13

Partition Usage

☒ High
☐ Low

Processor Type

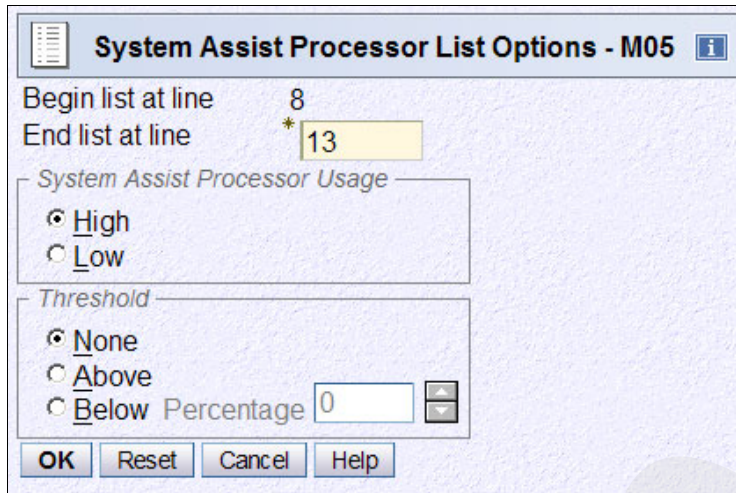
☒ All
☐ General purpose
☐ Integrated coupling facility
☐ Integrated facility for Linux
☐ System z application assist processor
☐ IBM System z Integrated information processor

Threshold

☒ None
☐ Above
☐ Below Percentage 0

OK Reset Cancel Help

Figure 13-6 Logical Partition Summary List options



System Assist Processor List Options - M05 ⓘ

Begin list at line 8

End list at line * 13

System Assist Processor Usage

☒ High

☐ Low

Threshold

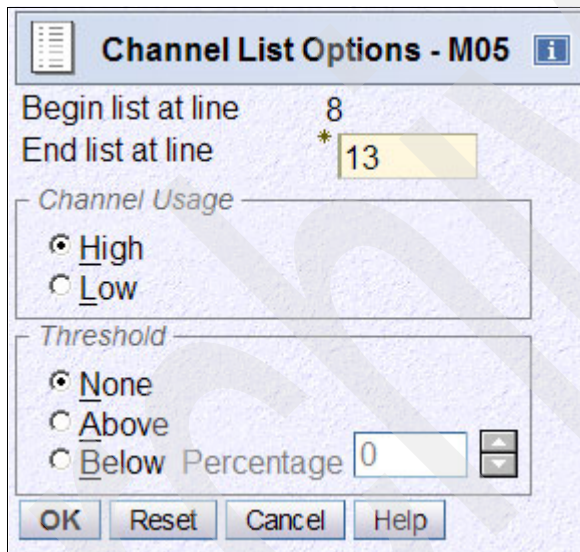
☒ None

☐ Above

☐ Below Percentage 0

OK Reset Cancel Help

Figure 13-7 System Assist Processor List options



Channel List Options - M05 ⓘ

Begin list at line 8

End list at line * 13

Channel Usage

☒ High

☐ Low

Threshold

☒ None

☐ Above

☐ Below Percentage 0

OK Reset Cancel Help

Figure 13-8 Channel List options

Logical Partition Channel List Options - M05

Partition name * FOO

Begin list at line 8

End list at line * 13

Channel Usage

☒ High

☐ Low

Threshold

☒ None

☐ Above

☐ Below Percentage 0

OK Reset Cancel Help

Figure 13-9 Logical Partition Channel List options

These panels display information about:

- Processor activity

The Processor Activity function can either monitor all processors of all types (CPs and specialty engines) or all processors of one type (just all CPs) in aggregate or a specified one. It can also only show usage for a certain processor state, either Problem state or Supervisor state. Finally, it can specifically monitor a storage key (or Program Status Word) from 0-F or exclude one, as shown in Figure 13-10 on page 279.

- Processor dedicated to an LPAR

The Processor Dedicated to an LPAR function is similar to the Processor activity, except it monitors a logical processor that is using a dedicated processor for an LPAR. Unlike a shared logical processor, which can be dispatched to any processor in the shared pool at any time, a dedicated logical processor is bound to a physical processor, which is what allows SAD to be able to show activity for dedicated logical processors. It can determine which physical processor is being used for the dedicated logical processor and thus show the same information as for a physical processor, as shown in Figure 13-11 on page 280.

- LPAR summary

Monitors all processor activity of a specific partition. Problem state, Supervisor state, or both can be chosen. See Figure 13-12 on page 280.

- SAP activity

Monitors usage of a specific SAP or all in aggregate, as shown in Figure 13-13 on page 281.

- Channel activity

Monitors the activity of a specific channel, given its CSS and CHPID, as shown in Figure 13-14 on page 281.

Physical Processor Options - M05

Processor

☐ All Shared

☒ All

☐ Specific Unit *

Processor Type

☒ All

☐ General purpose

☐ Integrated coupling facility

☐ Integrated facility for Linux

☐ System z application assist processor

☐ IBM System z Integrated information processor

Processor State

☒ Both problem and supervisor

☐ Problem

☐ Supervisor

Program Status Word (PSW) Key

☒ All

☐ Specific

☐ Key to exclude Key number *

Exclude Key for Processor State

☒ Both problem and supervisor

☐ Problem

☐ Supervisor

Threshold

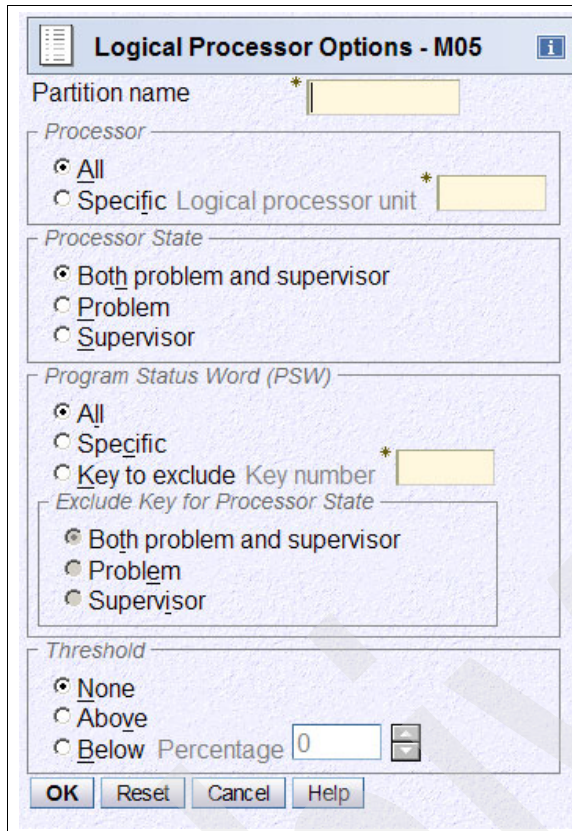
☒ None

☐ Above

☐ Below Percentage 0

OK Reset Cancel Help

Figure 13-10 Physical Processor Options



Logical Processor Options - M05

Partition name *

Processor

- ☒ All
- ☐ Specific Logical processor unit *

Processor State

- ☒ Both problem and supervisor
- ☐ Problem
- ☐ Supervisor

Program Status Word (PSW)

- ☒ All
- ☐ Specific
- ☐ Key to exclude Key number *

Exclude Key for Processor State

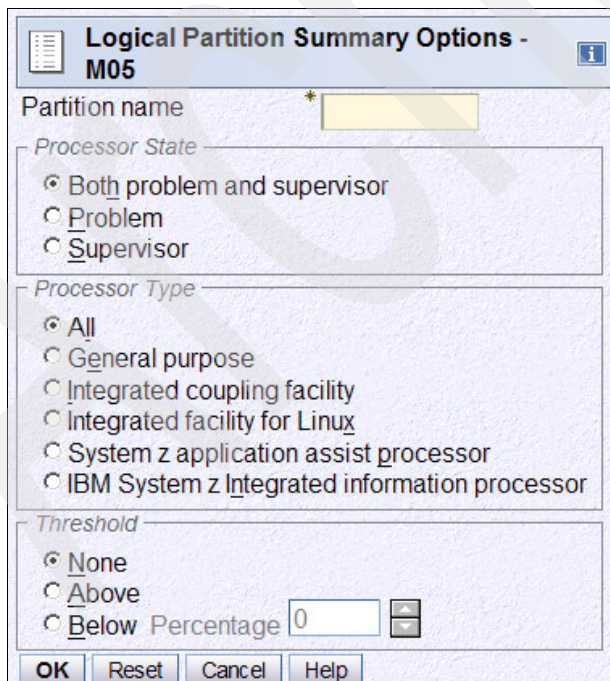
- ☒ Both problem and supervisor
- ☐ Problem
- ☐ Supervisor

Threshold

- ☒ None
- ☐ Above
- ☐ Below Percentage 0

OK Reset Cancel Help

Figure 13-11 Logical Processor Options



Logical Partition Summary Options - M05

Partition name *

Processor State

- ☒ Both problem and supervisor
- ☐ Problem
- ☐ Supervisor

Processor Type

- ☒ All
- ☐ General purpose
- ☐ Integrated coupling facility
- ☐ Integrated facility for Linux
- ☐ System z application assist processor
- ☐ IBM System z Integrated information processor

Threshold

- ☒ None
- ☐ Above
- ☐ Below Percentage 0

OK Reset Cancel Help

Figure 13-12 Logical Partition Summary Options

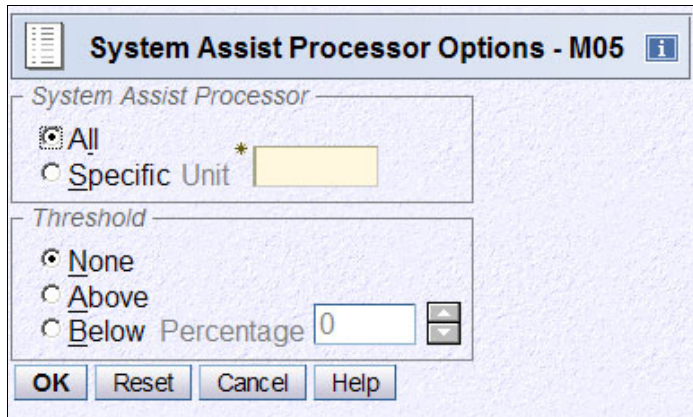


Figure 13-13 System Assist Processor Options

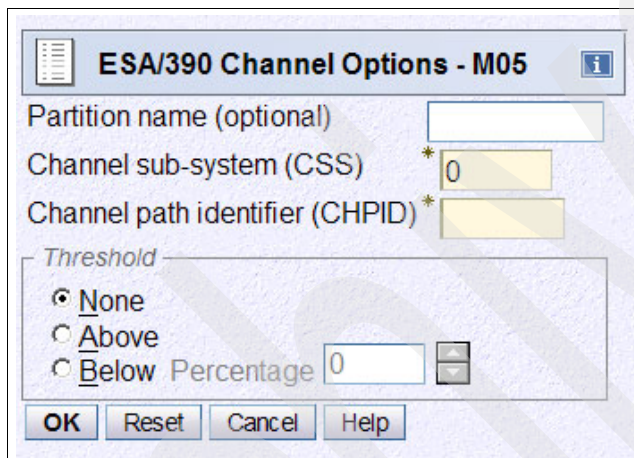


Figure 13-14 Channel Options

Thresholds

Almost all items, including lists, can have thresholds set to highlight the line if it goes above or below the threshold. The lines are always visible, but the user can visually see if the item is within the threshold or outside through a change in line color or threshold marker.

Predefined profiles

There are ten profiles that are defined by default on every CPC:

- ▶ **DEFAULT:** Displays high use CPs, SAPs, channels, and totals for CP and SAP usage.
- ▶ **CHANHIGH:** Displays the 49 highest used channels.
- ▶ **CHANLOW:** Displays the 49 lowest used channels.
- ▶ **PROCLIST:** Same as DEFAULT.
- ▶ **LPARSUMA:** Displays the total CP and SAP use and the highest used LPARs, CPs, and SAPs.
- ▶ **LPARSUMB:** Displays the highest used LPARs and channels.
- ▶ **VMPROCLIST:** Same as Default, except supervisor storage key 3 is ignored because it is used internally in z/VM.
- ▶ **PROCESSOR:** Displays the CP and SAP usage totals and the highest use SAP and CP lists.

- ▶ VMPROCESSOR: Displays the same as PROCESSOR, except supervisor storage key 3 is ignored.
- ▶ PROCUSAGEBYKEY: Displays the profile with a line for every processor storage key (in both problem and supervisor state).

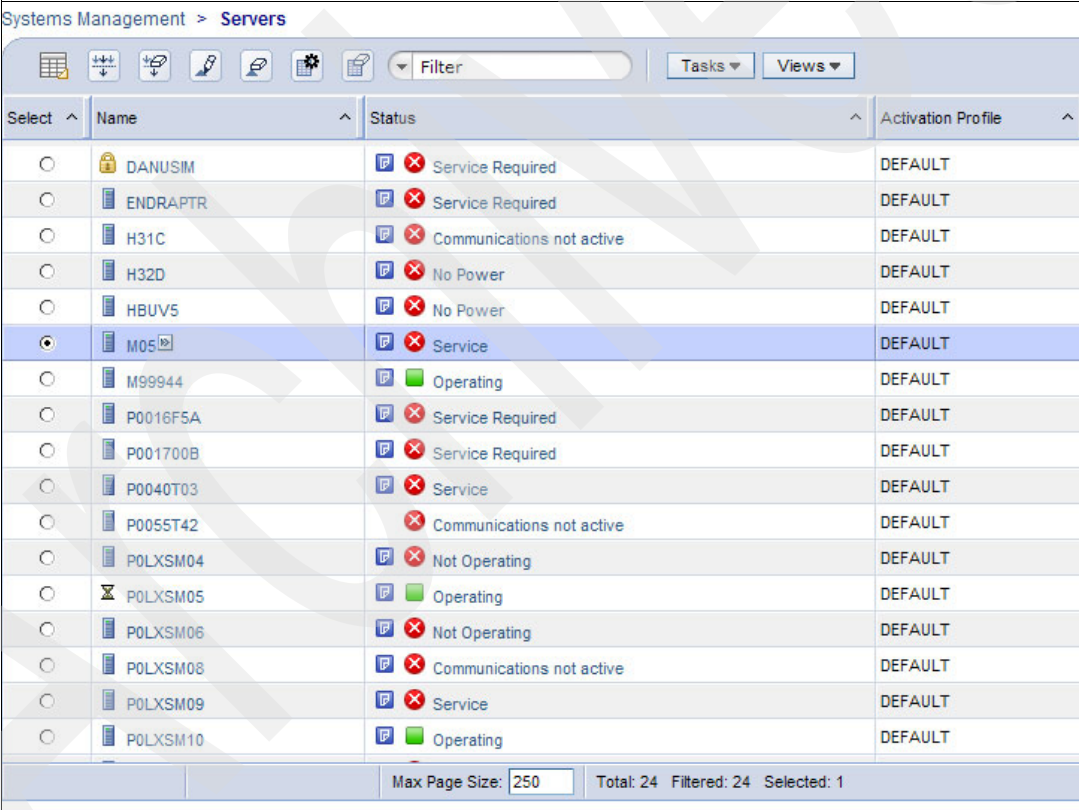
13.4 Using SAD

In this section, we discuss how to use SAD.

13.4.1 Monitoring activity with a predefined profile

To monitor activity with a predefined profile:

1. Before checking activity on a CPC, you must choose the profiles that you want to use. In the Defined CPCs Work Area, select the server you want to monitor, as shown in Figure 13-15 where CPC M05 is selected.



Select	Name	Status	Activation Profile
<input type="radio"/>	DANUSIM	Service Required	DEFAULT
<input type="radio"/>	ENDRAPTR	Service Required	DEFAULT
<input type="radio"/>	H31C	Communications not active	DEFAULT
<input type="radio"/>	H32D	No Power	DEFAULT
<input type="radio"/>	HBUV5	No Power	DEFAULT
<input checked="" type="radio"/>	M05	Service	DEFAULT
<input type="radio"/>	M99944	Operating	DEFAULT
<input type="radio"/>	P0016F5A	Service Required	DEFAULT
<input type="radio"/>	P001700B	Service Required	DEFAULT
<input type="radio"/>	P0040T03	Service	DEFAULT
<input type="radio"/>	P0055T42	Communications not active	DEFAULT
<input type="radio"/>	P0LXSM04	Not Operating	DEFAULT
<input type="radio"/>	P0LXSM05	Operating	DEFAULT
<input type="radio"/>	P0LXSM06	Not Operating	DEFAULT
<input type="radio"/>	P0LXSM08	Communications not active	DEFAULT
<input type="radio"/>	P0LXSM09	Service	DEFAULT
<input type="radio"/>	P0LXSM10	Operating	DEFAULT

Max Page Size: 250 Total: 24 Filtered: 24 Selected: 1

Figure 13-15 Selecting a server

2. Drag-and-drop the CPC (M05 in this example) onto the Customize Activity Profiles task, which takes you to the Profile Task shown in Figure 13-3 on page 274. Select the profiles that you want to use, and click **Change Status**. The profiles that you selected are now activated.

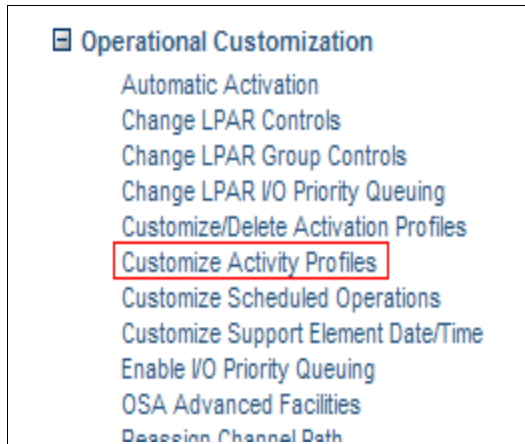


Figure 13-16 Customize Activities Profile task

13.4.2 Monitoring a server

Now we monitor a server. To monitor a server:

1. Select a server.
2. From the Daily group of tasks, choose **Activity**. Two windows are displayed: a summary of system activity that displays the server's Status and Activity Display (STAD) data, shown in Figure 13-17, and a window with full SAD data that is customized with the profile that you are using. STAD is a data stream that contains summary information, namely the average processor and channel usage and power and temperature information. Because the SAD loop only runs every 15 seconds, there is a delay before data is received, shown in Figure 13-18 on page 284.

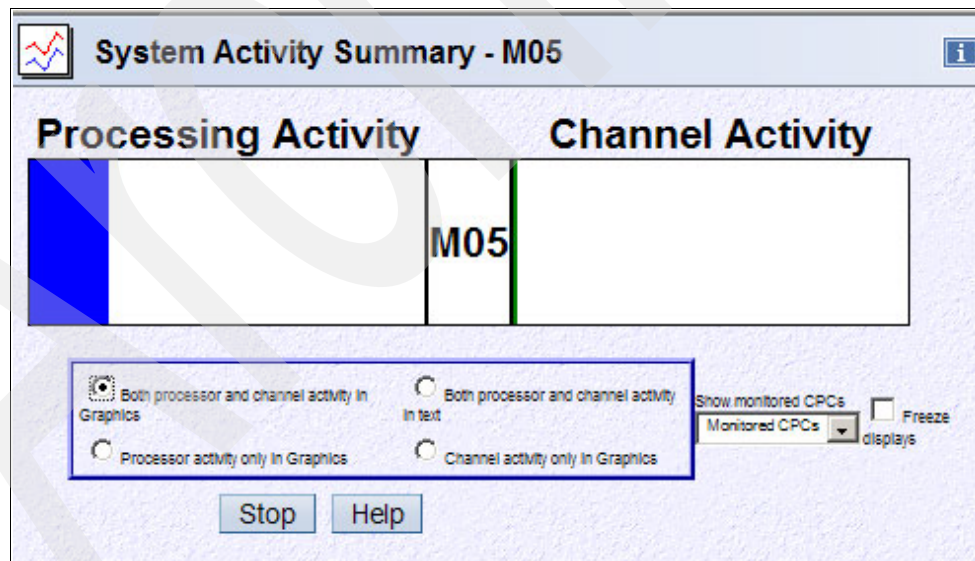


Figure 13-17 System Activity Summary

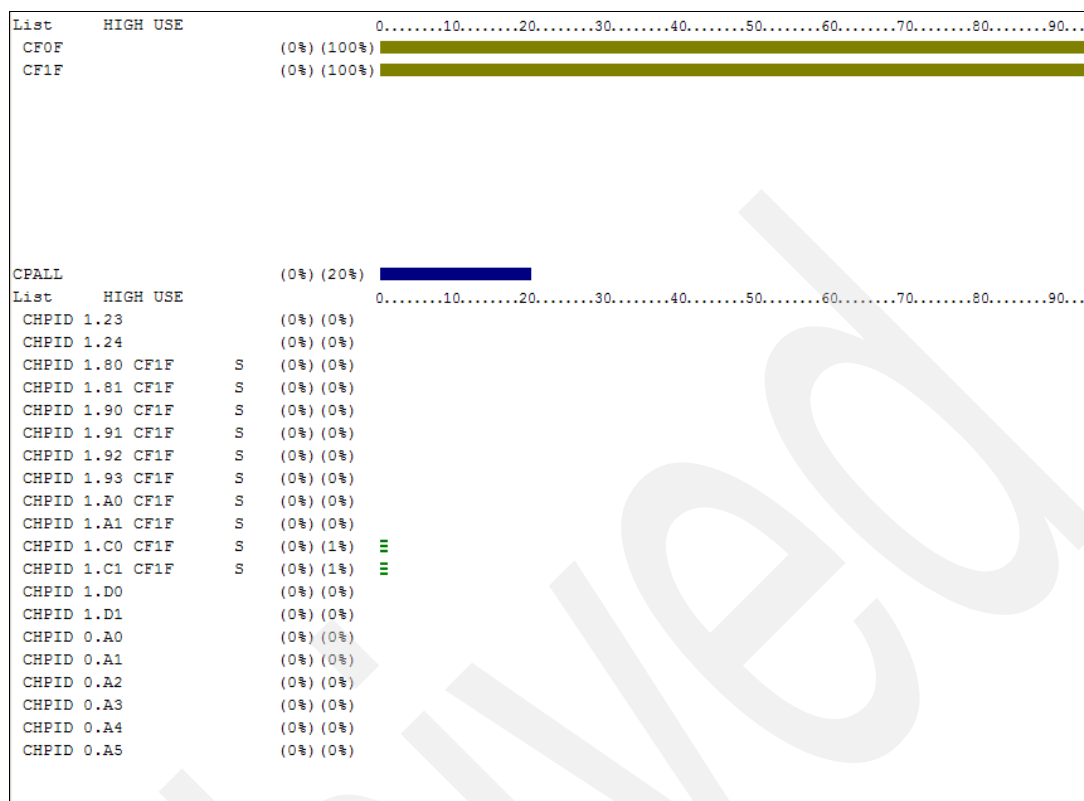


Figure 13-18 SAD window while data is being received

- In the Summary window, Figure 13-17 on page 283, bar graphs of processor and channel activity are shown in Figure 13-18. You can choose to disable either graph using the radio buttons, or disable the graphs and show the statistics as text from the system activity summary panel, shown in Figure 13-17 on page 283, which is useful if you are using a panel reader. You can also freeze the display.

For the most part, the full system activity display is fairly straightforward; however, some of the labels can be confusing. Table 13-2 on page 285 provides a list of all labels that can be used in the display.

Table 13-1 SAD display row definitions

Label	Description
nnnnnnnn (Any name)	Logical partition summary activity label
nnnnnnnn All	All logical processors dedicated to a logical partition activity label
nnnnnnnn d (Any name and number)	Logical processor dedicated to a logical partition activity label
CHPID xx	ESA/390 channel label
CHPID c.xx	ESA/390 channel label (for systems that support channel subsystems)
CHPID xx nnnnnnnn	ESA/390 channel label with logical partition name
CHPID c.xx nnnnnnnn	ESA/390 channel label with logical partition name (for systems that support channel subsystems)

Label	Description
CHPID xx nnnnnnnn	S ESA/390 channel label with logical partition name and shared channel label
CHPID c.xx nnnnnnnn S	ESA/390 channel label with logical partition name and shared channel label (for systems that support channel subsystems)
CPALL	All physical processors label (any type)
GPALL	All general purpose central processors (CP) label
SPALL	All specific processors label
ICFALL	All Internal Coupling Facility (ICF) processors label
AAPALL	All System z Application Assist Processor (zAAP) processors label
IFLALL	All Integrated Facility for Linux (IFL) processors label
IIPALL	All IBM System z9 Integrated Information Processor (zIIP) processors label
CP n	Specific physical processor label
GP n	Specific general purpose central processor (CP) label
SP n	Specific physical processor label (non-general purpose processor)
ICF n	Specific Internal Coupling Facility (ICF) processor label
AAP n	Specific System z Application Assist Processor (zAAP) processor label
IFL n	Specific Integrated Facility for Linux (IFL) processor label
IIP n	Specific IBM System z9 Integrated Information Processor (zIIP) processor label
Grid line	Grid line label
High use nnnnnnnn	High usage channel list label with logical partition name
List High use	High usage list label
List Low use	Low usage list label
Low use nnnnnnnn	Low usage channel list label with logical partition name
SAPALL	All system assist processors label
SAP n	Specific system assist processor label

Table 13-2 SAD display row definitions

13.5 SAD implementation design

Figure 13-19 on page 286 shows the general flow of SAD operation.

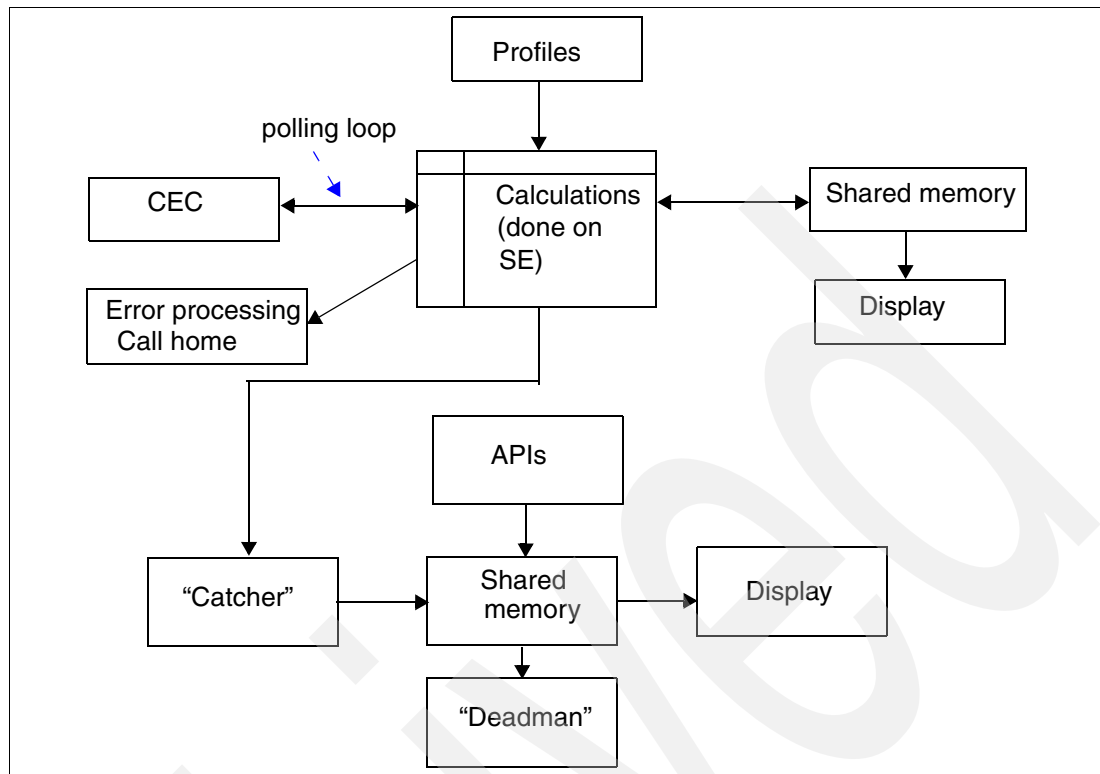


Figure 13-19 SAD data collection and use

13.5.1 Polling loop

In Figure 13-19, the data is first collected by a polling loop from the CEC. Statistics are generated and then sent to the HMC "Catcher" using a Generalized Data Stream (GDS). These statistics can also be *called home* in certain situations (error, repairs, and so on) to aid IBM in servicing the machine. In this section, we describe each stage of the general flow of SAD operations in more detail.

Every 15 seconds, the SE polls the CEC and collects all necessary data, which does not affect performance on the CEC in any noticeable way because dedicated monitoring hardware is polled. Performance might be affected on the SE, but this is acceptable. Performance data is then put in shared memory, and a separate process that is also on a 15-second loop generates statistics from the data. All data is then sent to the HMC in a binary, self-defining data stream format called Generalized Data Stream (GDS).

13.5.2 Calculation of statistics

Statistics are generated from the raw data that the SE collects and then sent to the HMC. Each of the four types of activity that the SE monitors has its own formula for how the raw data is turned into useful data for the SAD display. The raw data is in the form of counters that are updated every few milliseconds by the system. The CPs and SAPs have counters for when they are either busy or idle, and CPs have separate busy counters for each state that they can be in, that is, 16 problem state keys and 16 supervisor state keys. Each channel likewise has a busy counter and a reference counter that is incremented whether the channel is busy or not. The data that is collected for LPARs only counts overall problem and supervisor state usage, so fine-grained filtering by key is not possible.

We refer to these quantities using the following variables:

- ▶ `sap_busy`: Busy counter for SAPs
- ▶ `sap_idle`: Idle counter for SAPs
- ▶ `prob_state (i)`: Counter for all CPs in problem state i ($0 \leq i \leq 16$)
- ▶ `sup_state (i)`: Counter for all CPs in supervisor state i ($0 \leq i \leq 16$)
- ▶ `wait_state`: Wait state counter
- ▶ `lpar_prob_state (i)`: Counter for problem state of LPAR i ($0 \leq i \leq \text{number of LPARs}$)
- ▶ `lpar_supv_state (i)`: Counter for supervisor state of LPAR i ($0 \leq i \leq \text{number of LPARs}$)
- ▶ `chan (i)`: Busy counter for channel i
- ▶ `ref (i)`: Reference counter for channel i
- ▶ `scale (i)`: Custom scaling factor for channel i

Every 15 seconds, when the monitoring loop occurs, SAD computes the differences from the last reading for each of these counters and uses them in the following formulas.

For SAPs:

$$\text{sap_busy} / (\text{sap_busy} + \text{sap_idle}) * 100.$$

This formula is simply the percentage of time that the SAP was busy.

For CPs, the formula is slightly more complicated:

$$(\text{prob_state} + \text{sup_state}) / ((\text{prob_state} + \text{sup_state}) + \text{wait_state}) * 100$$

This is the sum of all active states of the CPs over the sum of all states including the idle state, which is scaled as a percentage. Specific states can be filtered out based on the profile rules.

For LPARs:

$$(\text{lpar_prob_state} + \text{lpar_sup_state}) / (\text{lpar_prob_state} + \text{sup_state}) * \text{proc_activity}.$$

This formula calculates the ratio of each individual LPAR's usage over all LPARs and applies this ratio to overall system usage, for example, if LPAR A's ratio is 50%, LPAR B's is 30%, LPAR C's is 20%, and the overall value is 60%, LPAR A's real utilization is 50% of 60% or 30%. LPAR B's is 18% and LPAR C's is 12%.

Channels can either be dedicated to an LPAR or shared by several LPARs. Regardless of it being shared or not, the formula for overall usage is simply:

$$\text{chani} / \text{ref}(i) * \text{scale}(i)$$

This result is the channel usage over total time elapsed, and `scale(i)` is a custom factor for that type of hardware that normalizes the value from 0-100.

If the channel is shared, another set of data is acquired with a set of counters (one per LPAR using the channel), and ratios are used similarly to the previous LPAR formula to calculate usage.

Finally, SAD also collects power and thermal metrics, but these values can be used directly without modification.

13.5.3 SAD data flow

Two types of data are in fact transmitted from the SE to the HMC: regular SAD data and STAD data. The SAD data contains the detailed system information that is specified in the active

profile, but it is only sent when requested by the HMC for performance reasons. The other type of data, STAD, contains summary information, namely the average processor and channel usage, which is based only on the processors and channels that are shown using the current SAD profile, and power and temperature information. STAD is sent every 15 seconds, and in fact the interruption of STAD data determines that communication with the SE is lost.

A variant of STAD is used to automatically give the HMC the status of monitored CPCs. This data is sent when the HMC first establishes communication with the SE, and contains all state and status information of the CPC and its LPARs. Unlike normal STAD data, this status information is only sent when there is a change in configuration to keep the HMC updated.

Referring back to Figure 13-19 on page 286, the incoming GDS arrives at the catcher. This code parses the GDS into an internal set of structures and stores the information in the shared memory block. The other processes will access the shared memory block as required to obtain information they need. The API would obtain data upon request by API callers. The display code periodically (and asynchronously to the catcher) reads the data and updates the SAD display. The “deadman” is responsible for cleaning out data in the shared memory when it is determined to be too old. It is also responsible for restarting the data flow from the SE if it is lost. This would be to obtain STAD data or the detailed SAD data.

SAD call home support

Very summarized performance metrics are saved on the SE’s hard drive periodically (typically every hour), and then on a longer period (typically one week) all of the hourly samples are sent to IBM. This information is made available to customers through Resource Link and is also used in service situations where historical performance data is necessary. This data is also used to guide the development of future systems by allowing IBM to identify resource hot-spots, which is the first step to knowing what to try to improve.

13.6 Creating a custom profile

Now let us create a custom SAD profile:

1. Return to the Customize Activity Profiles.
2. Choose a profile that most closely resembles the profile that you want to create because the new profile will start as a copy of that one. Make sure only that profile is checked.
3. Click **Customize**. We assume from here on that you are customizing the DEFAULT profile.
4. Save the profile using a new name. Erase the old name, put in a new name, and click **Save**, as shown in Figure 13-20 on page 289.

Customize System Activity Profile - P0LXSM05

Object: P0LXSM05

Profile name:

Description:

Modify line options:

- ☒ Change line
- ☐ Insert line
- ☐ Delete line
- ☐ Copy line to
- ☐ Move line to Line number

Select	Line	Component	Description
<input checked="" type="radio"/>	1.	List High Use	high usage processor list (all processor types except z Blade Extensions and system assist pro
<input type="radio"/>	2.	PU list	processor list line
<input type="radio"/>	3.	PU list	processor list line
<input type="radio"/>	4.	PU list	processor list line
<input type="radio"/>	5.	PU list	processor list line
<input type="radio"/>	6.	PU list	processor list line
<input type="radio"/>	7.	PU list	processor list line
<input type="radio"/>	8.	PU list	processor list line
<input type="radio"/>	9.	PU list	processor list line
<input type="radio"/>	10.	PU list	processor list line
<input type="radio"/>	11.	PU list	processor list line
<input type="radio"/>	12.	PU list	processor list line
<input type="radio"/>	13.	PU list	processor list line
<input type="radio"/>	14.	CPALL	processor ALL (all processor types except z Blade Extensions and system assist processors) p
<input type="radio"/>	15.	List High Use	high usage SAP processor list
<input type="radio"/>	16.	Sap list	SAP processor list line
<input type="radio"/>	17.	Sap list	SAP processor list line

OK Save Reset Cancel Help

Figure 13-20 Saving the SAD profile under a new name (partial)

- The default profile uses the maximum of 50 lines, so to add new lines, you must first delete lines that you do not need, or change list items to have fewer elements. Let us change the high usage processor list to show the five highest use processors, instead of 12. We also change it to show the five highest use processors above an activity threshold of 25%.
- Select **List High Use**, and in the Modify line options section, make sure Change line is elected. Click **OK**. The Change Line dialog is displayed, as shown in Figure 13-21, where you can change the type of line completely by choosing a different line type, but the current type is already selected. Press **OK** to go into the Physical Processor list options dialog, which we show in Figure 13-22 on page 290.

Change Line - P0LXSM05

Object: P0LXSM05

Profile name: TESTNAME

Line	Component	Description
1	List High Use	high usage processor list (all processor types except z Blade Extensions and system assist processors) processor state both problem

Choose Component for this Line

- ☒ Physical processor...
- ☐ Physical processor list...
- ☐ Processor dedicated to a logical partition...
- ☐ Logical partition summary...
- ☐ Logical partition summary list...
- ☐ System assist processor...
- ☐ System assist processor list...
- ☐ Grid line

- ☐ ESA/390 channel...
- ☐ Channel list...
- ☐ Channel list for a logical partition...
- ☐ Blank line

OK Cancel Help

Figure 13-21 Changing a line

Physical Processor List Options - P0LXSM05

Begin list at line: 1
End list at line: 13

Processor Usage

☒ High
☐ Low

Processor Type

☒ All
☐ General purpose
☐ Integrated coupling facility
☐ Integrated facility for Linux
☐ System z application assist processor
☐ IBM System z Integrated information processor

Processor State

☒ Both problem and supervisor
☐ Problem
☐ Supervisor

Program Status Word (PSW) Key

☒ All
☐ Specific
☐ Key to exclude Key number: *

Exclude Key for Processor State

☒ Both problem and supervisor
☐ Problem
☐ Supervisor

Threshold

☒ None
☐ Above
☐ Below Percentage: 0

OK Reset Cancel Help

Figure 13-22 Physical Processor List Options

7. Now you can change the number of lines. Rather than directly specifying the number of lines, you must write the line that the list should end on. The list's heading line is at 1, so if you change the end line to 6, five lines are shown. Change the end line to 6.
8. To monitor CPs with activity above 25%:
 - a. In the Threshold box, select **Above**, which activates the Percentage field.
 - b. In the Percentage field, change the value to 25%, and click **OK**.

You can also change the type of processors monitored from all processors to just general purpose ones or one of the specialty engines listed. You can choose here to only show activity in a certain processor storage key, here called a Program Status Word key. These options are the same in the dialog to add a non-list processor line.
9. After clicking **OK**, you are returned to the Customization page. There are now five PU list lines rather than 13. Now you can add the lines that you want for the profile. Decreasing the number of lines in the list does not delete any lines but rather leaves blank lines, so you must use the Change line tool again.
10. Now we add a line to monitor a specific LPAR's processor use. Select line 7, and in the Change Line dialog, select **Logical Partition Summary**, show in Figure 13-23 on page 291. In this dialog, enter the name of the LPAR. The other options are self-explanatory.

11. Let us fill in the remaining lines with a high use channel list that is specifically for the channels of that partition. Select the next line, line 8, and change the line. For the type of line, select **Channel list for a logical partition**, shown in Figure 13-24. This dialog is also self-explanatory. By default, the list will fill up the remaining lines.

After making these changes, the profile should look something like Figure 13-25 on page 292.

Figure 13-23 Logical Partition Summary Options

Figure 13-24 Logical Partition Channel List Options

Customize System Activity Profile - M05

Object: M05

Profile name: TESTNAME

Description: Processing Activity AND Channel List High Use (28)

Modify line options:

- ☒ Change line
- ☐ Insert line
- ☐ Delete line
- ☐ Copy line to
- ☐ Move line to

Line number: 1

Select	Line	Component	Description
<input type="radio"/>	1.	List High Use	high usage processor list (all processor types except z Blade Extensions and system assist processors) processor state both problem and supervisor for all
<input type="radio"/>	2.	PU list	processor list line
<input type="radio"/>	3.	PU list	processor list line
<input type="radio"/>	4.	PU list	processor list line
<input type="radio"/>	5.	PU list	processor list line
<input type="radio"/>	6.	PU list	processor list line
<input type="radio"/>	7.	FOO	FOO logical partition summary (all processor types except z Blade Extensions and system assist processors) processor state
<input checked="" type="radio"/>	8.	High Use FOO	FOO high usage channel list
<input type="radio"/>	9.	Ch list	channel list line
<input type="radio"/>	10.	Ch list	channel list line
<input type="radio"/>	11.	Ch list	channel list line
<input type="radio"/>	12.	Ch list	channel list line
<input type="radio"/>	13.	Ch list	channel list line
<input type="radio"/>	14.	CPALL	processor ALL (all processor types except z Blade Extensions and system assist processors) processor state both problem and supervisor for all program
<input type="radio"/>	15.	List High Use	high usage SAP processor list
<input type="radio"/>	16.	Sap list	SAP processor list line
<input type="radio"/>	17.	Sap list	SAP processor list line

OK Save Reset Cancel Help

Figure 13-25 Customize System Activity Profile (partial)

Deleting a line is simple. Select the line you want to delete, choose **Delete line** from the line options, and click **OK**. You are not prompted for confirmation, so be careful.

It is also straightforward to copy or move lines. Select the line you want to copy or move, choose either **Copy line to** or **Move line to**, enter the target line number in the Line number box, and click **OK**.

When you finish making changes, click **Save**.

13.7 Questions and discussions

1. Who might use SAD displays? Application programmers, systems programmers, system operators, IBM service people, executive management?
2. Why are SAD profiles important?
3. How busy would you expect a “normal” processor (or group of processors) to be in a typical System z production installation? How does this compare with typical Personal Computer servers? How busy do you expect a given channel (CHPID) to be?
4. SAD functions can collect a very large amount of data. How much does this slow down the System z processors?
5. How accurate is the SAD data? What is the time resolution of the data?

Capacity on demand

Quickly changing the computing capacity in response to changes in demand is a characteristic capability of on-demand computing. IBM provides that capability on mainframe servers through the *Capacity of Demand for System z* functions.

In this chapter, you learn about Capacity on Demand for System z and how the Hardware Management Console and Support Element are used for on-demand computing.

This chapter covers the following topics:

- ▶ Overview
- ▶ Fundamental components:
 - Reconfigurable firmware
 - Unused hardware
 - Process controls
- ▶ Capacity on Demand records:
 - Resource limits
 - Time limits
 - Consumption limits
 - Customer options for setting limits
- ▶ Capacity on Demand offerings:
 - On/Off Capacity on Demand
 - Capacity Backup
 - Capacity for Planned Events
 - Customer Initiated Upgrade
 - Ordering Capacity on Demand records
- ▶ Firmware process controls:
 - Support element LIC
 - Machine LIC

- ▶ Managing CoD records:
 - Support element user interface
 - Other record management options
 - Other considerations and activities
- ▶ CoD record management scenarios:
 - Peak workloads
 - Expense control
 - Disaster recovery
 - More scenarios and instructions
- ▶ Questions and discussion

After studying this material you should understand why capacity on demand is important and the various forms it can take. Less important, but still relevant to our general topic of system management, is how Capacity on Demand is implemented with the System z.

14.1 Overview of Capacity on Demand for System z

The history of Capacity on Demand for System z can be traced to traditional hardware upgrades. From the earliest mainframes to System z today, traditional hardware upgrades are exactly what you would expect. Trained service technicians arrive on site using tools and following instructions to add or replace parts in a mainframe server to change its computing capacity. In short, traditional hardware upgrades are physical hardware upgrades. At one time, physically upgrading hardware was the only way to change the computing capacity of a mainframe server. Today physical hardware upgrades are still necessary at times, but planning, ordering, and installing such upgrades are simply not fast enough nor flexible enough to respond to ever changing demands for computing resources that are so common in today's business environments. So IBM began offering capacity on demand (CoD) upgrades as a faster, more flexible alternative to traditional hardware upgrades.

A CoD upgrade electronically upgrades a System z machine. No service technician is required, and a System Programmer can install and activate the upgrade. The only tools needed are the Hardware Management Console and Support Element. The only part that is needed is specialized type of firmware called a CoD record.

CoD upgrades are possible because all of the necessary physical hardware is already installed. A machine's firmware controls how much of that hardware is available for use. A CoD upgrade changes the machine's firmware to make more or less of its hardware available for use. But CoD upgrades do not physically change a machine's hardware. Changing a machine's hardware still requires a traditional hardware upgrade. CoD upgrades change only the machine's firmware.

This is an important distinction between CoD upgrades and traditional hardware upgrades, and it is the basis of what makes CoD upgrades uniquely qualified to support on demand computing. When a mainframe customer needs an upgrade for only a limited time, perhaps weeks, days, or even hours, to do and then undo a physical upgrade is simply not practical. As electronic upgrades that change only a machine's firmware, CoD upgrades provide a practical way to offer temporary upgrades. Capacity on demand for System z initially supported permanent upgrades only and continues to support them today. Support for temporary upgrades soon followed, and with their lower costs and ease of use, they quickly became the focus of mainframe customer interest in and use of capacity on demand for System z, as well as the focus of IBM development of new types of CoD upgrades.

14.2 Fundamental components

The fundamental components of Capacity on Demand for System z are:

- ▶ Reconfigurable firmware
- ▶ Unused hardware
- ▶ Process controls

14.2.1 Reconfigurable firmware

You learned from the Chapter 2, “Systems management overview” on page 23 that one of the key elements of System z configuration management is that firmware controls the availability of computing resources in the hardware configuration, such as the number and type of processors and the amount of memory. This key element, the use of firmware to control the availability of computing resources makes electronically upgrading System z machines possible.

System z firmware is Licensed Internal Code (LIC). Licensed Internal Code Configuration Control (LICCC) is a type of Licensed Internal Code that defines the hardware configuration of a mainframe server. A capacity on demand (CoD) record is a specialized type of LICCC that allows redefining the hardware configuration. A CoD record is the firmware part that must be installed on a System z machine to enable upgrading it on demand.

Every System z machine has LICCC. IBM Manufacturing configures and installs a unique LICCC record that defines a machine's hardware configuration according to its purchaser's order. Afterwards, if a machine's owner never chooses to upgrade it, its one original LICCC record remains unchanged and forever defines the machine's hardware configuration. Otherwise, if a machine's owner chooses to upgrade it, by any means, the upgrade includes reconfiguring the machine's LICCC to define its new, upgraded hardware configuration. A machine's LICCC is reconfigured every time its hardware configuration is upgraded, regardless of whether it is a physical upgrade or a CoD upgrade.

14.2.2 Unused hardware

A System z mainframe often has more hardware installed than its owner has purchased, which is intentional and is a key element of several System z hardware systems management disciplines and functions, including configuration management and capacity on demand for System z. Each machine's LICCC defines how much of its installed hardware is available for use. Any unused hardware remains installed but cannot be used unless and until the machine's LICCC is redefined to enable using it.

The hardware that is physically installed in System z mainframes varies by machine type and model, and each machine type and model has some hardware that is standard and some that is optional. Mainframe customers can purchase a machine type and model with any configuration of standard and optional hardware that IBM offers. The purchased hardware configurations vary from customer to customer, but customers with the same machine type and model have the same standard hardware installed (optional hardware varies), for example, the standard hardware for machine type 2097 model E64 (an IBM System z10 Enterprise Class) includes 64 processor units (PUs). All customers that purchase a 2097 E64 have 64 PUs installed, regardless of the hardware configuration that they purchase. A 2097 E64 machine purchased with only one central processor (CP) has 63 unused PUs, while a 2097 E64 purchased with 40 CPs has 24 unused PUs. Of course the purchase prices of the two machines vary accordingly. The machine with one CP costs less than the machine with

40 CPs. There is no customer cost that is associated with the PUs that remain installed but are unused in either machine.

Many other 2097 E64 configurations are possible and include other types of processors and hardware (memory, for example). However, every 2097 E64 has the same potential computing capacity from its standard hardware (again, optional hardware varies). Each individual machine's unique LICCC defines how much of its potential capacity can be used. It is the unused portion of a machine's potential capacity that Capacity on Demand for System z can make use of.

14.2.3 Process controls

Capacity on Demand for System z is the process IBM developed to use a mainframe's reconfigurable firmware and unused hardware to meet customer requirements for on demand computing. Since its introduction, Capacity on Demand for System z has evolved with each new generation of mainframes. It is a flexible process that can meet requirements that differ from customer to customer and change as business environments change. Yet it also has limits. Neither IBM nor mainframe customers want an *anything goes* process. The process controls that IBM developed, both to provide flexibility and set limits, are the final fundamental components of Capacity on Demand for System z.

The process controls are implemented in a System z machine's LIC (that is, firmware) and its Support Element, and with a specialized type of LICCC called a capacity on demand record. From a high-level view of the process, the roles of the controls are:

- ▶ A CoD record sets the rules, defining how a customer can use a machine's unused hardware for CoD upgrades.
- ▶ A customer uses the machine's Support Element to choose what they want to do with a CoD record.
- ▶ The machine's LIC carries out the customer requests and records and reports usage back to the customer and IBM.

14.3 Capacity on demand records

A capacity on demand record is the key process control for Capacity on Demand for System z. A CoD record is a type of LICCC. Each record is coded with rules that define the entitlement that the record offers; therefore, CoD records are referred to also as *entitlement records*. Entitlement, in this context, is the extent to which IBM authorizes the use of a machine's unused hardware for CoD upgrades. The terms entitlement and entitlement record typically appear in contracts, license agreements, and similar legal documents that pertain to the use of Capacity on Demand for System z. But they can be found also in announcement letters, product descriptions, and technical support information.

The rules that are coded on CoD records are developed by IBM and are designed to balance two general objectives: meeting customer requirements for on demand computing and meeting IBM requirements for asset protection (recall mainframe owners have not purchased the unused hardware in their System z mainframes). Considering both requirements are subject to change, using a LICCC-based process control is a significant development. CoD records enable IBM to quickly define new rules for meeting new requirements.

IBM develops different sets of rules, which are referred to as *capacity on demand (CoD) offerings* to address different requirements.

For each set of rules, there are three general types of limits:

- ▶ Resource limits
- ▶ Time limits
- ▶ Consumption limits

The flexibility of CoD records is demonstrated by how IBM applies these limits for different CoD offerings, giving each offering its distinct characteristics. Some offerings allow using a CoD record repeatedly for any number of upgrades, for example, while other offerings allow using a CoD record only once. Some offerings allow upgrading different combinations of hardware within a record's resource limits, while other offerings limit records to upgrading to one specific configuration of resources.

14.3.1 Resource limits

Resource limits for a CoD record identify the types of hardware it can be used to upgrade and, for each type, set either a maximum upgrade or target upgrade. Model capacity, memory, and specialty processors, such as the Integrated Coupling Facility (ICF), are some examples of the different types of hardware resources that CoD records can be used to upgrade. Different CoD offerings generally support the same types of hardware, but some offerings support more or less than others.

Setting a target upgrade for each supported type of hardware is characteristic of CoD offerings that allow upgrading to one specific configuration of resources. In contrast, setting a maximum upgrade for each supported type of hardware is characteristic of CoD offerings that allow upgrading both a range of upgrades for each type of hardware and a variety of possible combinations of upgrades of different hardware types. Furthermore, maximum upgrade limits are set as relative values, for example, if a machine has three System Assist Processors (SAPs) and a CoD offering allows doubling that capacity, then a CoD record's maximum limit on SAP upgrades is set as up to 3 more rather than up to 6 total. With maximum upgrade limits set as relative values, a machine's CoD records remain usable even if its permanent hardware configuration changes after the records are configured. Consider again the machine with three SAPs and a CoD record that allows upgrades of up to 3 more. If the machine's hardware configuration is permanently changed to include eight SAPs, the previously configured CoD record for up to 3 more SAPs can still be used for upgrades to nine, ten, or 11 SAPs.

14.3.2 Time limits

Time limits define maximum periods of use, for CoD records, upgrades, or both. Different CoD offerings typically support different time limits. Some CoD offerings limit how long a record can be used before it expires (for example, six months, one year, or five years), while other offerings do not set expiration dates on records. Similarly, when CoD records are used to activate upgrades, some CoD offerings limit how long an upgrade can remain in effect after it is activated (for example, three days, ten days, and 90 days), while upgrades that are activated under other offerings can remain in effect indefinitely.

14.3.3 Consumption limits

CoD offerings might set consumption limits instead of, or in addition to, setting time limits on the use of CoD records and upgrades. Consumption limits control the maximum cumulative usage of a CoD record for activating upgrades. Consumption limits are represented on a CoD record by counters called *tokens*.

One type of consumption limit is simply the number of times a record can be used to activate upgrades. A similar limit is the number of test upgrades a record allows. Mainframe customers need test upgrades for a variety of purposes, such as trying the CoD upgrade process for the first time, training IT staff, and practicing emergency backup or disaster recovery procedures. Test upgrades typically cannot remain activated as long as "real" upgrades because separate time limits are set to control their duration. Tokens that control the consumption of real and test upgrades are set simply as the total number of each type of upgrade. A token is expended each time a CoD record is used to activate an upgrade. When all the tokens are used up, the record can no longer be used to activate upgrades.

Capacity tokens, referred to also as resource tokens, are another type of consumption limit. They limit a CoD record's maximum cumulative activation of CoD upgrades. For CoD records that use capacity tokens, a separate number of tokens is set for each type of hardware that a record can be used to upgrade. One capacity token represents the activation of one unit of capacity for one day. The unit of capacity is determined by the type of hardware, for example, the unit of capacity for a model capacity upgrade is one million service units (MSU), while the unit of capacity for System z Integrated Information Processor (zIIP) upgrades is one zIIP. So one model capacity token allows the activation of one MSU for one day, and one zIIP token supports activating one zIIP for one day.

When a CoD record is used to activate an upgrade, capacity tokens are consumed each day that the upgrade remains activated. Each day's highest activated capacity determines the number of tokens that are consumed that day. When all of the tokens are used up, the record can no longer be used to activate that type of upgrade, for example, consider a CoD record with 20 capacity tokens for zIIP upgrades. The 20 zIIP tokens support several simple upgrade possibilities: 1 additional zIIP for 20 days, two zIIPs for 10 days, four zIIPs for five days, and so on. Alternately, the tokens might be consumed by some combination of zIIP upgrades that are activated separately and perhaps over several months, such as:

- ▶ Two additional zIIPs for three days that consume six tokens
- ▶ One additional zIIP for 10 days that consume 10 more tokens
- ▶ Four additional zIIPs for one day that consume the final four tokens

Using capacity tokens is characteristic of CoD offerings that allow mainframe customers to factor financial or budget considerations into the consumption limits of CoD records. Given the variety of international currencies, monetary values are not practical as consumption limits for CoD records; however, CoD upgrades have monetary value of course. So a monetary value, such as a maximum IT budget amount, can be expressed as a quantity of CoD upgrades, which in turn can be represented by a number of capacity tokens.

14.3.4 Customer options for setting limits

Mainframe customers of a CoD offering typically can set some or all of the resource, time, and consumption limits for their CoD records within the ranges that IBM established for the offering. They set a record's limits when they order it from IBM, for example, consider a CoD offering that allows upgrading model capacity up to twice its current capacity (that is, up to a 100% increase in model capacity). A customer can choose to order a CoD record that allows upgrades up to 100% more model capacity, they might order a record that allows upgrades of up to 50% more, or perhaps just 10% more. Similarly, if the CoD offering allows using a CoD record for up to six months, a customer can choose that time limit or set a lower limit when they order a CoD record, which is typical of most CoD offerings. The CoD offering establishes a range of options up to a maximum limit, and individual customers choose the limit that they want to set for each CoD record.

Some CoD offerings also allow *record replenishment*. A mainframe customer can change or reset some or all of the resource, time, and consumption limits that were set for previously

configured CoD records by ordering replenishment records that are configured to set the new limits. A record can be replenished at any time after it is ordered. Record replenishment is particularly useful when customers have a CoD record installed with an upgrade activated, and the record is approaching a time or consumption limit. If a customer wants the upgrade to remain activated beyond the record's pending limit, they must simply order and install a replenishment record that resets the limit anytime before the limit is reached.

14.4 Capacity on demand offerings

CoD offerings vary among different System z machine types now and can change with subsequent generations of mainframes to take advantage of advances in technology or to meet new customer requirements. Indeed, IBM designed CoD records to accommodate such changes. At this time, as an example of the different characteristics of different offerings, we look at some of the CoD offerings for the IBM latest mainframe, the IBM System z10.

At the time of writing, Capacity on Demand offerings for System z10 include three types of temporary upgrades:

- ▶ On/Off Capacity on Demand (On/Off CoD)
- ▶ Capacity Backup (CBU)
- ▶ Capacity for Planned Events (CPE)

System z10 also supports a Customer Initiated Upgrade (CIU) offering for permanent upgrades.

In general, CoD records that are used for temporary upgrades are referred to as temporary records or temporary entitlement records, while records that are used for permanent upgrades are called permanent records or permanent entitlement records. More specifically, the name of an offering is often used to describe its CoD records and the upgrades activated with those records, for example, On/Off CoD records are used for On/Off CoD upgrades under the On/Off CoD offering, and CBU records are used for CBU upgrades under the CBU offering.

14.4.1 Model capacity

In earlier chapters, we described how central processors (CPs) for some mainframe models can be configured to run at less than full speed and that a model's CP speed is its *capacity setting*. In the context of CoD offerings, a *model capacity upgrade* can be an upgrade of the number of CPs, the capacity setting, or both.

14.4.2 Software license charges

A CoD offering might describe its CoD upgrades as providing either additional hardware capacity or replacement hardware capacity to indicate whether upgrades might increase software license charges. Software license charges vary according to the license agreements for different software products, but in general, CoD upgrades that provide additional capacity typically increase software license charges, while upgrades that provide replacement capacity do not.

14.4.3 On/Off Capacity on Demand

The On/Off Capacity on Demand (On/Off CoD) offering for System z10 provides temporary, additional processor resources. It typically is used in response to short-term increases in demand. This offering enables purchasing a mainframe with a hardware configuration right-sized for an enterprise's normal demand for computing capacity, yet still be prepared for higher demand, whether it is expected (weekly peak workloads, for example) or unexpected (increased transaction processing due to unusual increase in insurance claims or product sales, for example).

The resources that are eligible for On/Off CoD upgrades are model capacity and specialty processors (ICF, IFL, SAP, zAAP, and zIIP). Upgrades generally are limited to twice the machine's purchased capacity, but other limits might apply to some specialty processors. Customers can set lower limits on upgrades when they order an On/Off CoD record, for example, a System z10 with three IFL processors is eligible for upgrades of up to three more, for a total of six IFLs. Yet the machine's owner can choose to configure a record for model capacity upgrades only. They can prevent using the record for IFL upgrades by limiting it to zero additional IFLs.

An On/Off CoD record can be used for up to six months, but customers can set an earlier expiration date when they order a record.

Consumption limits are optional. Customers can set consumption limits when they order an On/Off CoD record by configuring the record with either the exact upgrades that they want to purchase, referred to as prepaid upgrades, or with the total spending limit for each type of upgrade that the record allows. In either case, the customer's consumption limits are set on the record as a number of capacity tokens for each type of upgrade.

An On/Off CoD record can be used repeatedly for any number of upgrades, within its consumption limits and time limit. Customers can replenish On/Off CoD records at any time. They can replenish a record to increase its resource limits, extend its time limit, or add more capacity tokens to its consumption limits (within the ranges established by IBM for the offering).

The daily cost of an On/Off CoD upgrade is a fraction of the cost of an equivalent permanent upgrade. Mainframe customers who consider On/Off CoD upgrades must weigh the total costs of activating one or more temporary upgrades over a period of time against the costs of a permanent upgrade.

14.4.4 Capacity Backup

The Capacity Backup (CBU) offering for System z10 provides temporary, replacement processor resources for up to 90 days. As its name suggests, it is used for backup purposes, typically for unexpected increases in demand that are created by emergencies or disasters. This offering enables purchasing a mainframe to serve as an emergency backup for other mainframes. Mainframes that are used as backups often have configurations with significantly more unused hardware than purchased hardware to better ensure enough resources are available in an emergency.

The resources that are eligible for CBU upgrades are model capacity and specialty processors (ICF, IFL, SAP, zAAP, and zIIP). Upgrades are generally limited only by the machine's maximum installed hardware. Customers can set limits on upgrades when they order a CBU record.

A CBU record can be used for up to five years, which is the maximum CBU contract length under this offering. But shorter contract lengths are available. Customers choose the contract length, from one-to five years, when they order a record.

CBU records support both real and test upgrades. Each record allows one real CBU activation for up to 90 days and up to 15 test activations for up to 10 days each.

Customers can replenish CBU records at any time, which enables them to use a CBU record repeatedly for any number of real and test upgrades during the length of its contract. If the record is used for a real activation, expending its one real activation token, then the record must be replenished to get a new token to enable another real activation. Customers can replenish a CBU record also to increase its resource limits, extend its contract length up to five years total, or add tokens for test activations (up to 15 total at a time).

The cost of a CBU record is based on its resource, time, and consumption limits, and on the fact that its real activation is limited to 90 days and its test activations are limited to 10 days each. A mainframe customer pays this cost regardless of whether the record is ever used for a real activation, much like the premium for an insurance policy. Mainframe customers who are considering buying CBU records must weigh their costs against the costs of owning and operating backup systems with permanent configurations that could provide equivalent resources in an emergency.

14.4.5 Capacity for Planned Events

The Capacity for Planned Events (CPE) offering for System z10 provides temporary, replacement processor resources for up to three days. It typically is used for backup purposes for expected increases in demand that are created by planned events, such as weekend outages for system maintenance.

The resources that are eligible for CPE upgrades are model capacity and specialty processors (ICF, IFL, SAP, zAAP, and zIIP). Upgrades generally are limited only by the machine's maximum installed hardware. Customers can set limits on upgrades when they order a CPE record.

Each CPE record allows one real activation for up to three days. A CPE record cannot be replenished; however, it can be used at any time, and it does not have an expiration date.

The cost of a CPE record is based on its resource limits and the three-day limit on its one real activation. Mainframe customers who consider buying CPE records must weigh their costs against the costs of owning and operating substitute systems with permanent configurations that could provide equivalent resources for short-term needs.

14.4.6 Customer Initiated Upgrade

The Customer Initiated Upgrade (CIU) offering for System z10 supports permanent upgrades of processor resources and memory. It typically is used in response to or in anticipation of sustained growth in demand for computing capacity.

The resources that are eligible for permanent upgrades are model capacity, specialty processors (ICF, IFL, SAP, zAAP, zIIP), and memory. Permanent upgrades also support unassigning model capacity and IFL processors, that is, a permanent upgrade can decrease a machine's model capacity or number of IFLs. Upgrades generally are limited only by the machine's maximum installed hardware, but other limits can apply to some specialty processors.

Each permanent upgrade is a one-time event. So resource, time, and consumption limits on permanent upgrade records are set accordingly. When a mainframe customer orders a permanent upgrade, they choose the exact target configuration that they want to upgrade to. The cost of the upgrade is based on the target configuration. A permanent upgrade record can be activated only once (that is, its only consumption limit is one real activation), and it cannot be replenished. After the upgrade is activated, it remains activated, and the upgrade's target configuration becomes the machine's new permanent configuration.

14.4.7 Ordering records for capacity on demand offerings

Mainframe customers of CoD offerings can order CoD records using the Internet from an IBM support Web site called IBM Resource Link. They can place orders by themselves without any direct IBM assistance or involvement. The Resource Link Web pages for ordering records set the ranges of the resource, time, and consumption limits that IBM established for CoD offerings.

Mainframe customers with access to the centralized IBM support network, called RETAIN, can take delivery of CoD records shortly after ordering them by downloading the records from RETAIN to their machine's Support Element; otherwise, for customers without access to RETAIN, IBM delivers CoD records to them on removable media, such as a DVD.

14.5 Firmware process controls

As the key process control for Capacity on Demand for System z, a CoD record sets the rules and defines how a customer can use a machine's unused hardware for CoD upgrades. The two other process controls are System z firmware. The first is Licensed Internal Code on a System z machine's Support Element. The second is LIC on the machine itself.

14.5.1 Support element LIC

After a mainframe customer orders CoD records, they must use a hardware systems management application to choose what they want to do with the records. The Support Element LIC includes one such application. With a dedicated Support Element that is connected to every mainframe and its configuration management role already established, the Support Element was the IBM obvious choice for implementing and controlling CoD record management processes for individual machines.

More specifically, CoD record management is implemented and controlled through the Support Element configuration manager, which is a fundamental component of a Support Element's LIC that has long supported other mainframe configuration management functions. The Support Element configuration manager provides the user interface for managing CoD records, such as installing records and using them for CoD upgrades. The configuration manager also supports SNMP application programming interfaces (APIs) for a subset of CoD record management functions. System Programmers for mainframe customers and independent software vendors can use SNMP APIs to develop their own hardware systems management applications.

Another component of the Support Element's firmware, its LICCC, is used for communications between the configuration manager and the machine's LIC and for storing the information that they exchange, for example, a mainframe customer takes delivery of a CoD record by using the configuration manager's user interface to either download the record from RETAIN or import it from removable media, such as a DVD. In either case, the Support Element LICCC is used to store the CoD record, which is referred to as staging a CoD record

on the Support Element. Staging a CoD record makes it available for installation on a machine.

14.5.2 Machine LIC

After a mainframe customer chooses what they want to do with a CoD record, the machine's LIC carries out the customer request and records and reports usage back to the Support Element. Ultimately, when a CoD upgrade is activated, it is the machine's LIC that reconfigures its own Licensed Internal Code Configuration Control to make a CoD upgrade take effect.

The role of a machine's LIC as a CoD process control begins when a mainframe customer uses a machine's Support Element to install a CoD record on the machine. The Support Element sends the machine a copy of the CoD record to install. But the LIC does not unconditionally install a record. It has a dual role as a process control that includes protecting IBM assets, namely the unused hardware in the machine that its owner has not purchased, for example, the LIC validates all CoD records to verify that their LICCC was coded by an IBM LICCC generation utility. It will not install or allow using a CoD record that was tampered with. The LIC also controls how many temporary records can be installed. On System z10 machines, for example, there are eight slots in the hardware system area (HSA) for installing up to eight temporary records.

After a CoD record is installed, the machine's LIC also handles all subsequent customer requests to use it. Options for using records vary among CoD offerings and machine types, for example, a CoD record for a permanent upgrade is activated immediately after it is installed and no further action is necessary or allowed, while a temporary CoD record for System z10 supports both activating and deactivating upgrades, and uninstalling the record when it is no longer needed.

When a mainframe customer activates a CoD upgrade, the machine's LIC first verifies that the CoD record's resource, time, and consumption limits permit activating the upgrade. The LIC activates an upgrade only if it is within the record's limits. To activate an upgrade, the machine's LIC reconfigures its own LICCC to make the new configuration of hardware resources available for use. Activating CoD upgrades does not interrupt the machine's current hardware operations. The machine's LIC can activate CoD upgrades concurrently with other hardware operations. There is no need to shutdown a machine before activating an upgrade and no need to restart it afterwards.

After a CoD upgrade is activated, the machine's LIC records and reports its usage back to the Support Element. For permanent upgrades, recording and reporting usage is a one-time event. For temporary upgrades, the LIC records and reports usage back to the Support Element at regular intervals, such as daily, for as long as the temporary upgrade remains activated.

Recording and reporting temporary upgrade usage varies among machine types. For the System z10, for example, the machine's copy of an installed temporary record, called an accounting record, is used for recording and reporting information about temporary upgrades that the record is used to activate. The machine's LIC uses the accounting record as a meter, recording each upgrade's type, capacity level, and duration. Similarly, for records with time or consumption limits, the machine's LIC decrement tokens from the accounting record's applicable token pools while a CoD upgrade remains activated.

The machine's LIC reports upgrade information back to the Support Element to make it available to both the customer and IBM. The Support Element LIC uses the information to maintain a CoD upgrade history as part of the machine's vital product data (VPD). The VPD and its CoD upgrade history are among the operational data the Support Element regularly

sends to IBM via the Hardware Management Console's Remote Support Facility (RSF), which you learned about in Chapter 9. IBM uses a machine's upgrade history to compute the bill for its upgrades.

14.6 Managing CoD records

Working with CoD records does not require physical access to a System z machine. All System z machine types provide a Support Element user interface mainframe that customers can use to manage a machine's CoD records. Some machine types also support other options. But much like CoD offerings, options for managing CoD records vary among different System z machine types now and might change with subsequent generations of mainframes. As an example, we look at some of the CoD record management options for the IBM latest mainframe, the IBM System z10.

14.6.1 Support Element user interface

The Support Element configuration manager provides the user interface for manually managing CoD records for System z10. The user interface is part of the Support Element's Perform Model Conversion task. The task is in the CPC Configuration task list. System Programmers can use the task remotely at a Hardware Management Console, through a console session with the Support Element.

The Perform Model Conversion task's CoD record management functions include staging, installing, and removing CoD records, and activating and deactivating CoD upgrades. Figure 14-1 on page 305 shows the Perform Model Conversion menu for temporary upgrades. There are no special techniques for using the task. Its simple user interface consists of a main menu that you use to select the type of upgrades that you want to work with, and the work you want to do. You will not find offering-specific terms, such as On/Off CoD or CBU, on the Perform Model Conversion window. The main menu is designed to support general types of upgrades rather than particular CoD offerings. So to work with CoD records for a particular offering, you must choose the type of upgrades that the records support, either permanent upgrades or temporary upgrades.

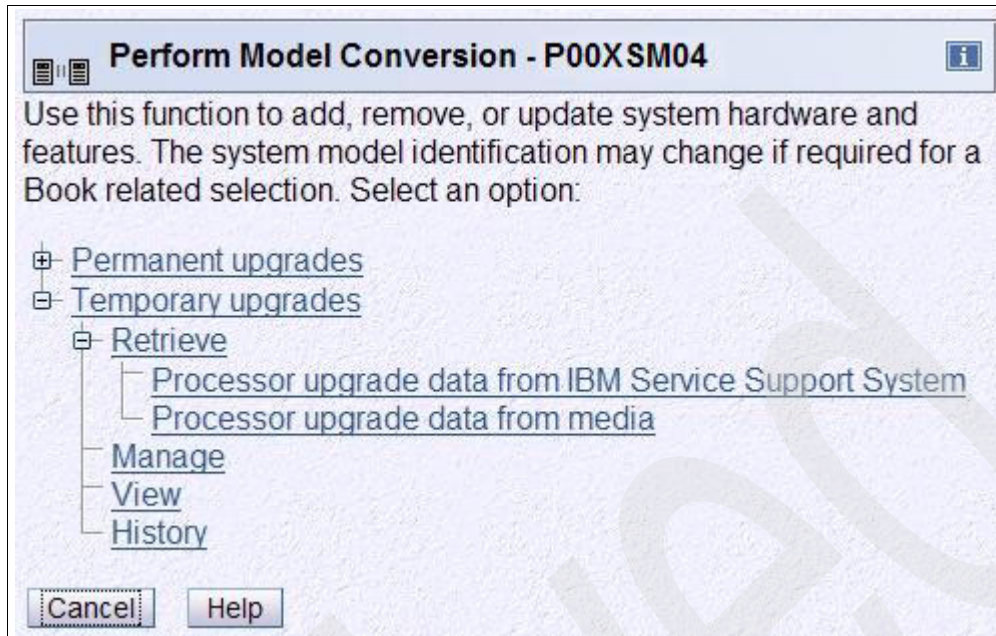


Figure 14-1 Perform Model Conversion menu options for temporary upgrades

The main menu uses several other general terms for CoD records and record management actions that are applicable to CoD offerings for System z10:

- ▶ Processor/memory upgrade data refers to LICCC for permanent upgrades, which includes CoD records for permanent upgrades.
- ▶ Processor upgrade data refers to LICCC for temporary upgrades, which includes CoD records for temporary upgrades, such as On/Off CoD, CBU, and CPE records.
- ▶ Retrieve means staging CoD records on the Support Element, either by downloading them from the IBM Service Support System (RETAIN) or importing them from media.
- ▶ Apply means installing and activating CoD records for permanent upgrades.

Figure 14-2 shows the Perform Model Conversion menu for permanent upgrades.

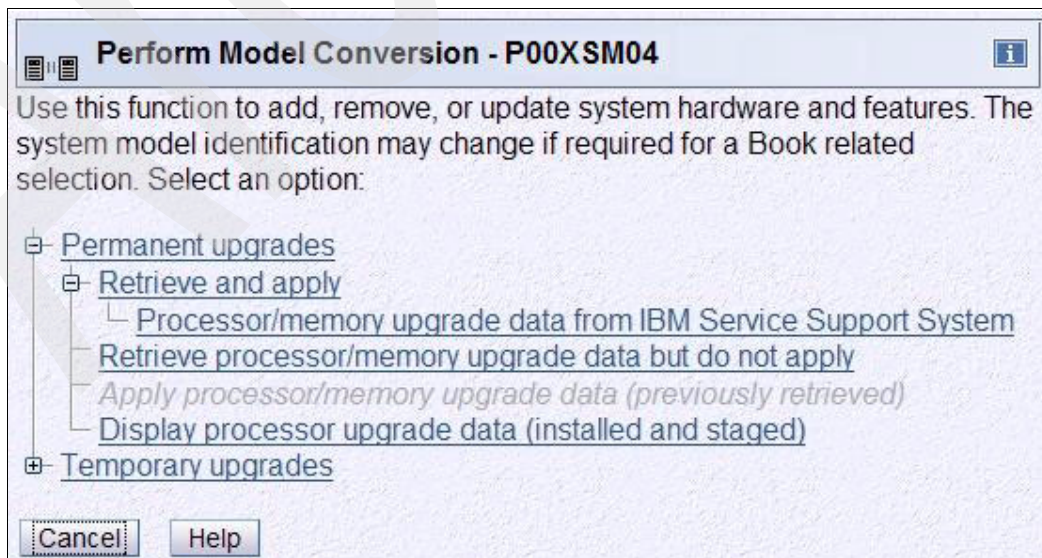


Figure 14-2 Perform Model Conversion menu options for permanent upgrades

Upon selecting an action from the main menu, the user interface displays the windows that you need to complete the work that you want to do, for example, selecting **Manage** under Temporary upgrades opens a window that you can use to work with installed CoD records or staged CoD records (if any) for temporary upgrades, such as On/Off CoD, CBU, and CPE records.

The screenshot shows a window titled "Temporary Upgrades - P00XSM04". It has two tabs: "Installed Records" (selected) and "Staged Records". Below the tabs, there is a text area explaining the table and providing instructions. The table lists installed records with columns for Record ID, Record Type, CPs, SAPs, ICFs, IFLs, zAAPs, zIIPs, and Status. A specific record (28234339, CBU) is highlighted. Below the table, there is a "Description:" section with status details and a note about unlimited values. A "System Summary" section provides capacity and processor information. At the bottom, there are buttons for "Details...", "Add processors...", "Remove processors...", "Delete", and "Help". A "Cancel" button is located at the very bottom left.

Record ID	Record Type	CPs	SAPs	ICFs	IFLs	zAAPs	zIIPs	Status
28234339	CBU	*0/0/0	0/0/0	0/0/0	0/0/0	2/0/0	2/0/0	Installed
Active Temporary		0	0	0	0	0	0	
Permanent		2	3	0	0	0	0	
Total Used		2	3	0	0	0	0	

Description:
Status details: N/A
* - The maximum value is unlimited.

System Summary
Model-Capacity Identifier: 402 MSUs: 36
Model-Temporary-Capacity Identifier: 402 Available PUs: 10
Model-Permanent-Capacity Identifier: 402

Buttons: Details... Add processors... Remove processors... Delete Help

Cancel

Figure 14-3 Installed records for temporary upgrades

Installed records can be used to activate and deactivate upgrades. To activate an On/Off CoD upgrade for example, a System Programmer selects an On/Off CoD record on the Installed Records page of the Temporary Upgrades window, and then click the **Add processors** button, which opens the Change Activation Levels window, Figure 14-3, which provides controls for selecting the model capacity and processor configuration that you want to upgrade to within the selected record's resource limits.

Record ID: 28234459 Record Type: On/Off CoD Status: Installed
Description:
Model-Capacity Identifier: 402 CPs: 0 MSU Value: 36

--- Select Action ---

Select ^	Target Model-Capacity ID ^	CPs ^	Target MSU Value ^	MSU Cost ^
<input checked="" type="radio"/>	402	0	36	0
<input type="radio"/>	403	1	52	16
<input type="radio"/>	404	2	66	30

Processors

Select the counts you would like for each processor type.

SAPs: Current: 0
ICFs: Current: 0
IFLs: Current: 0
zAAPs: Current: 0
zIIPs: Current: 0

When you have finished changing the activation levels, press the "OK" button to save your changes.

OK

Cancel

Restore Current Levels

Help

Figure 14-4 Change Activations Levels window for On/OFF CoD record

Similarly, to deactivate the On/Off CoD upgrade, a System Programmer selects the same On/Off CoD record on the Installed Records page of the Temporary Upgrades window, but this time click the **Remove processors** button, which opens the Change Activation Levels window again that provides controls for restoring the machine's configuration to its base capacity.

14.6.2 Other record management options

The System z10 supports several other options for using temporary CoD records to activate and deactivate temporary upgrades:

- ▶ Scheduled operations
- ▶ SNMP APIs
- ▶ z/OS Capacity Provisioning

All of these other options apply only to temporary CoD records that are already installed through the Support Element user interface provided by its Perform Model Conversion task.

A System Programmer can schedule some console operations by using its Customize Scheduled Operations task to choose the operations and set up their start times. For System z10, the Support Element's Customize Scheduled Operations task supports scheduling the activation and deactivation of On/Off CoD upgrades. Recall the On/Off CoD offering provides temporary, additional processor resources to cover short-term increases in demand. Using the On/Off CoD offering and scheduled operations together enables System z10 customers

to automate the use of temporary upgrades to meet predictable or regularly recurring increases in demand.

We earlier introduced the use of SNMP application programming interfaces for automated system management of System z hardware. System Programmers for mainframe customers and independent software vendors can use SNMP APIs to develop their own hardware systems management applications. The SNMP APIs for System z10 include commands for activating and deactivating temporary capacity, CoD-related objects, and object attributes that applications can query for their current settings. The IBM publication *System z Application Programming Interfaces* (order number SB10-7030) documents the use of SNMP APIs, including APIs for capacity on demand.

z/OS Capacity Provisioning, a component of the IBM z/OS Release 9 operating system, is an example of a hardware systems management application that uses the SNMP APIs for capacity on demand. For a System z10 that runs z/OS R9 or greater, z/OS Capacity Provisioning supports manual and automated activation and deactivation of On/Off CoD upgrades. Its support for automated operations includes using either time-based schedules or workload-based conditions to control the duration of On/Off CoD upgrades. The IBM publication *z/OS V1R9.0 MVS Capacity Provisioning User's Guide* (order number SC33-8299) documents the use of z/OS Capacity Provisioning.

14.6.3 Other considerations and activities

Managing CoD records is only one component of effective use of Capacity on Demand for System z. Planning and preparing for the use of CoD offerings, and decision-making about when to upgrade, by how much, and for how long, are also essential. Though the Support Element and Hardware Management Console are not necessarily used to conduct these other activities, they are described here briefly to provide a more complete picture of the context in which mainframe customers would manage CoD records and activate CoD upgrades:

- Planning for CoD offerings early

Mainframe customers can sign up for CoD offerings at any time, but planning for CoD offerings early, when planning to purchase a machine for example, is often the most effective approach. All CoD offerings rely on the availability of unused hardware. The question is whether a machine has enough unused hardware to meet the purpose of using an offering. Buying a System z machine to serve as a backup system is a common example. Capacity Backup (CBU) is the most appropriate CoD offering in this case. It provides temporary, replacement processor resources for up to 90 days. But to use CBU effectively, the configuration of a backup system must have enough unused hardware to meet its planned purpose, whether that is to backup one system at a time or perhaps multiple systems at once. By deciding whether to use the CBU offering when considering which model to buy, a mainframe customer can better decide how much unused hardware the machine should have.

- Ordering CoD records in advance

After a machine is enabled for CoD offerings, its owner can prepare to use the offerings by ordering CoD records in advance before they are needed for activating upgrades. Mainframe customers can use the IBM Resource Link Web site to order CoD records that are configured with the resource, time, and consumption limits of their choice within the ranges that IBM established for each offering. Before ordering CoD records for an offering, it is often worthwhile to review the ordering options, decide how many records to order, and determine each record's purpose, all in advance, to better plan how to configure each record's limits when ordering.

- Staging and installing CoD records in advance

CoD offerings for some machine types support staging CoD records on a machine's Support Element (that is, downloading or importing records and storing them on the Support Element), at any time, to make them available for installation. Some CoD offerings also support installing CoD records prior to using them to activate upgrades. If a mainframe customer ordered CoD records for such offerings, they can further prepare for activating CoD upgrades by using the Support Element's Perform Model Conversion task to stage and install the CoD records in advance, which simply saves time in situations that require activating upgrades as quickly as possible.

- Preparing operating systems for using CoD upgrade resources

Before using CoD records to activate upgrades, mainframe customers should consider preparing operating systems to use the additional resources made available by the upgrades. CoD upgrades are considered disruptive if an operating system must be restarted before it can use the additional resources. Otherwise, if restarting the operating system is not necessary, then CoD upgrades are nondisruptive. Support for nondisruptive CoD upgrades varies among different operating systems, and those that do support nondisruptive CoD upgrades often require advanced planning and programming to use the additional resources that are made available by the upgrades.

- Monitoring workload, performance, and so on

To guide decision-making about when to upgrade, by how much, and for how long, mainframe customers should consider how they will monitor workload, performance, or any other conditions relevant to their decisions about using CoD upgrades. There are many options for monitoring machine conditions, including Support Element and Hardware Management Console tasks, operating system components such as z/OS Capacity Provisioning, IBM software products such as the IBM z/OS Resource Measurement Facility (RMF), and applications developed by independent software vendors or mainframe customers themselves.

Other considerations and activities are possible, depending on how mainframe customers choose to use CoD offerings. The common characteristic of these and other components of using of Capacity on Demand for System z effectively is that they precede the activation of CoD upgrades, which emphasizes an important point: the ability to activate CoD upgrades on demand does not diminish the importance of advanced planning, preparations, and decision-making. On the contrary, these activities are essential to establishing an effective on demand environment.

14.7 CoD record management scenarios

To demonstrate some basic CoD record management activities, we look at several common hardware systems management scenarios where temporary CoD upgrades for System z10 are most useful:

- Peak workloads
- Expense control
- Disaster recovery

14.7.1 Peak workloads

Peak workloads generally describes any short-term increase in demand for computing capacity that is significantly higher or longer-lasting than ordinary fluctuations in workload. A mainframe customer with a System z10 machine can respond to peak workloads by using the

On/Off CoD offering. They can sign up for the offering at any time. But considering the offering when purchasing a machine is recommended, for example, consider a customer that needs a machine that is configured with model capacity 709 (9 CPs running at the maximum capacity setting) and 1 System Assist Processor (SAP). Any of the five models of the System z10 EC (E12, E26, E40, E56, E64) will support that configuration, but each model has a different number of unused processor units (PUs) that are available for On/Off CoD upgrades. In this example, the customer wants a machine configured to use 10 PUs (9 CPs and 1 SAP). On a model E12, with its standard hardware of 12 PUs, this customer's configuration leaves two PUs unused. On a model E64, with 64 PUs, the same configuration leaves 54 PUs unused. Considering the On/Off CoD offering limits upgrades to twice a machine's purchased capacity, the E12 provides too little unused capacity to take full advantage of the offering, while the E64 provides much more than could be used. The customer must consider purchasing a model E26. With its 26 PUs, it has 16 PUs unused and available for On/Off CoD upgrades. It is the smallest model that both supports the customer's configuration and enables taking full advantage of the On/Off CoD offering.

Let us assume that this customer purchased that machine, a model E26, configured for model capacity 709 with 1 SAP and signed up for the On/Off CoD offering. Their next step is to order at least one On/Off CoD record from the IBM Resource Link Web site. The On/Off CoD offering limits upgrades to twice a machine's purchased capacity, so the customer can order a record that enabled upgrades of up to 100% more model capacity and for one additional SAP. But perhaps this customer knows their peak workload typically requires no more than a 35% increase in model capacity and never requires additional SAP capacity. They could instead order a record that enables upgrades of up to 35% model capacity only and no additional SAPs, thereby eliminating the possibility that a System Programmer can create unwanted additional expenses by activating more capacity than is needed.

The customer can still consider ordering additional On/Off CoD records for other purposes, for example, while they need that 35% model capacity upgrade record for the peak workloads they expect, they might still want a 100% model capacity upgrade record to be prepared for the unexpected, such as unusual increases in demand beyond their peak workloads. It costs nothing to order these types of On/Off CoD records and have them on hand.

Next, the customer installs at least the On/Off CoD record that they ordered for handling their expected peak workloads. Installing the record makes it available for activating On/Off CoD upgrades. A System Programmer uses the Support Element's Perform Model Conversion task to work with this temporary upgrade record. They retrieve the record to stage it on the Support Element, and then using the window for managing temporary upgrades, they select from the list of staged records the On/Off CoD record that they want to install.

To further prepare for their peak workloads, the customer might also want to plan and prepare for activating CoD upgrades nondisruptively, if the machine's operating systems support nondisruptive upgrades.

After installing the On/Off CoD record, the customer's internal procedures determine when they use the record to upgrade the machine's capacity. It can be as simple as upgrading every Friday morning at 6 am or based on monitoring measurements of workload, performance, or other conditions.

When it is time to upgrade, the System Programmer uses the Support Element's Perform Model Conversion task to open the window for managing temporary upgrades. Select the On/Off CoD record from the installed records list, and click **Add processors** to open the window that they can use to select the configuration that they want to upgrade to.

After activating the upgrade, the customer's internal procedures determine how long it should remain in effect. It can be as simple as following a schedule or based on monitoring

conditions. In any case, when it is time to deactivate the upgrade, the System Programmer uses the Support Element's Perform Model Conversion task to open the window for managing temporary upgrades. They select the same On/Off CoD record from the installed records list, and then click **Remove processors** to open the window that they can use to restore the machine's configuration to its base capacity.

14.7.2 Expense control

The On/Off CoD upgrades that we described in the previous example about peak workloads are referred to as postpaid upgrades because IBM computes the bill for each upgrade after it is activated. The bill for an upgrade takes into account how much capacity was added and how long the upgrade remained activated. On/Off CoD records for postpaid upgrades do not have any consumption limits. They can be used repeatedly for activating upgrades, and upgrades can remain activated indefinitely while the record is within its expiration date, which can create unacceptable financial uncertainties and exposures for some customers. Other customers might simply prefer to pay for upgrades in advance. Such customers can still consider using the On/Off CoD offering. There are two alternatives to postpaid upgrades that allow setting consumption limits on On/Off CoD records to control expenses:

- ▶ On/Off CoD records with prepaid upgrades
- ▶ On/Off CoD records with spending limits

Let us return to our previous example and apply these alternatives. The machine is a System z10 EC model E26 configured for model capacity 709 with one SAP, and the customer is using the On/Off CoD offering to handle expected peak workloads and unexpected, unusual increases in demand beyond their peak workloads.

The customer knows their peak workload requires no more than a 35% increase in model capacity. Now let us assume that they also know this peak workload lasts for three days and occurs only once a month. In this case, they can order an On/Off CoD record with prepaid upgrades. They can still configure the record to enable upgrades of up to 35% more model capacity only and no additional SAPs as they did in the previous example. In addition to setting those resource limits on the record, they also set consumption limits by further configuring the record to support activating one or more three-day upgrades to model capacity 713 (which is approximately a 35% increase from model capacity 709), for example, for a six month record, they can order six prepaid upgrades to model capacity 713 for three days each.

As with our previous example, the customer might consider ordering additional On/Off CoD records for other purposes. While they need a 35% model capacity record with prepaid upgrades for the peak workloads they expect, they might still want a 100% model capacity upgrade record to be prepared for the unexpected. Because they cannot anticipate the exact target model capacity or length of the unexpected upgrades in advance, ordering a record with prepaid upgrades is not an option. To better control expenses in this case, they can order an On/Off CoD record with spending limits. A spending limit is simply the maximum total dollar amount (or an amount in their local currency) that the customer is prepared to spend on upgrades, for example, the customer can still configure a record to enable upgrades of up to 100% model capacity, but they can also set a \$200,000 spending limit on model capacity upgrades.

The consumption limits for prepaid upgrades and spending limits are represented on an On/Off CoD record by capacity tokens. For these examples, tokens are consumed each day a model capacity upgrade remains activated. Each day's highest activated capacity determines the number of tokens that are consumed that day. When all of the tokens are used up, the record can no longer be used to activate model capacity upgrades. On/Off CoD records for prepaid upgrades and spending limits can be replenished at any time. So if a customer wants

to continue using a record beyond its previously configured consumption limit, they can order and install a replenishment record that resets the limit anytime before it is reached.

After ordering On/Off CoD records with prepaid upgrades or spending limits, basic record management is the same as the previous example. The records must be staged on the Support Element to make them available for installation, and then installed to make them available for activating upgrades. The customer follows their own internal procedures to determine when to activate and deactivate upgrades.

14.7.3 Disaster recovery

Disaster recovery generally describes procedures for responding to and recovering from unexpected system outages. Using emergency backup systems to restore capacity that is lost elsewhere is a common component of disaster recovery procedures. A mainframe customer can use a System z10 machine with the Capacity Backup (CBU) offering as an emergency backup system. They can sign up for the offering at any time. Considering the offering when purchasing a machine is recommended, for example, consider a customer that needs a z10 EC as a backup system for production systems supported by three other z10 EC machines:

- ▶ A model E12 configured with model capacity 703 (3 CPs running at the maximum capacity setting) and one Internal Coupling Facility (ICF)
- ▶ A model E26 configured with model capacity 712 (12 CPs running at the maximum capacity setting) and one System Assist Processor (SAP)
- ▶ A model E26 configured with model capacity 718 (18 CPs running at the maximum capacity setting), one Integrated Facility for Linux (IFL), and two SAPs

If the customer wants to be prepared to backup one system at a time, perhaps because their different physical locations make it unlikely that all three would suffer an outage at the same time, then they can choose the z10 EC model that matches the model of the production system with the greatest configuration, which is a model E26 in this example. Otherwise, to be prepared for backing up all three systems at once, the customer must choose a z10 EC model that can support their combined configurations. An E40 is the smallest model that can support the combined configurations of 38 processor units (PUs) in this example, leaving two PUs unused. But perhaps the customer wants to use the backup system also as a production system with model capacity 709 and one SAP. They must purchase at least a model E56 to have enough PUs to support both the production configuration and backup the combined configurations of the other three systems.

Let us assume that this customer purchased this machine, a model E56, that is configured for model capacity 709 with one SAP and signed up for the CBU offering. Their next step is to order at least one CBU record from the IBM Resource Link Web site. The CBU offering allows upgrades to the machine's maximum capacities, so the customer can order a record that enables upgrades for backing up the combined configurations of the three production systems: up to model capacity 742 (that is, up to 33 more CPs), up to one more ICF, up to one more IFL, and up to three more SAPs, or they might prefer to order three CBU records instead, one each for backing up the particular configuration of each production system. Let us assume though that they order one record for backing up all three systems, and choose a five year contract. So the record has a five year time limit, and its consumption limits are for five test activations and one real activation.

Next, the customer makes all of the necessary preparations for using their model E56 as a backup system for their production systems, such as preparing its operating systems, software, peripheral hardware, and so on. These preparations include installing the CBU record. Installing the record makes it available for activation. A System Programmer uses the

Support Element's Perform Model Conversion task to work with this temporary upgrade record. First they retrieve the record to stage it on the Support Element. Next, using the window for managing temporary upgrades, they select from the list of staged records the CBU record they want to install.

After installing the CBU record, the customer can use the CBU record's test activations to test their disaster recovery or emergency backup procedures. In the event of an actual disaster or emergency, they can use the record's real activation to restore capacity that is lost from the systems that are affected by the event. In either case, when it is time to upgrade, the System Programmer uses the Support Element's Perform Model Conversion task to open the window for managing temporary upgrades. To open the window that they can use to select the configuration that they want to upgrade to and indicate whether the upgrade is a real or test activation, select the CBU record from the installed records list, and click **Add processors**.

Test activations can last for up to 10 days, and real activations can last for up to 90 days. But the customer can always deactivate earlier. To deactivate the upgrade, the System Programmer uses the Support Element's Perform Model Conversion task to open the window for managing temporary upgrades. They select the same CBU record from the installed records list, and then click **Remove processors** to open the window that they can use to restore the machine's configuration to its base capacity.

14.7.4 More scenarios and instructions

The IBM publication *System z10 Capacity on Demand User's Guide* (order number SC28-6871) provides step-by-step instructions for using the Perform Model Conversion task to manage CoD records for System z10. The IBM Redbooks publication *IBM System z10 Capacity on Demand* (order number SG24-7504) includes more detailed scenarios for using CoD offerings and, in the context of each scenario, provides instructions for using software tools to monitor workload and for using the Perform Model Conversion task to manage CoD records. Both publications are available at no charge at ibm.com.

14.8 Questions and discussion

1. In the earlier section on HMC and SE system management disciplines, capacity on demand is described as a function of multiple disciplines: configuration management, performance management, and serviceability. Why is this appropriate?
2. Should a System Programmer be responsible both for ordering CoD records and for activating CoD upgrades, or should someone else be responsible for ordering CoD records? What are the advantages and disadvantages of separating these responsibilities? What are the advantages and disadvantages of not separating them?
3. What are some ways a mainframe customer can activate On/Off CoD upgrades on multiple System z10 machines at the same time?
4. Research utility computing on the Internet. Compare and contrast on demand computing with utility computing.

Archived

Repair and Verify

In System z, most of the hardware subsystems have redundant backups, which allows concurrent and transparent repair of the System z components (FRUs).

The Repair Action task, also known as Repair and Verify (R&V), is the component that is used to repair problems in the system. R&V automates the human interaction with the machine to avoid inadvertent side effects during the complex part replacement. It is only used by a trained Service Representative (SR), either from IBM or a third-party provider. R&V displays sequences of panels to guide the SR through the repair. The immediate goal is to fix the problem as quickly as possible with the fewest parts possible and the least impact to customer operations.

Additionally, a higher-level goal is to provide all of the information that is necessary for the SR to complete the repair without assistance and with a minimum amount of prior training. The main reason for doing this is that the typical SR has very little actual hands on experience because of the extremely low hardware failure rate of the System z. In place of this experience, R&V guides the SR through the repair, displaying detailed directions on panels using text, graphics, and in some cases videos of specific actions. In addition, there is also Web-based education that is available through Resource Link that allows the SR to review specific procedures before going to the account.

Repair & Verify interacts with many of the other components and subsystems using APIs to perform tasks, such as turning power on and off to parts of the system, putting PCHIDs (a physical channel identifier) into a particular state, running diagnostics, and checking for redundant hardware. Not only does this shorten the time that it takes to perform a task, because the SR does not have to navigate to various panels and make selections, but also it greatly reduces the opportunity for human error. Because most repairs are done concurrently with customer operations, it is critical to avoid an error, such as powering off the wrong part which could cause the entire system to crash.

While it is extremely unlikely that a normal user will ever use R&V, the intent of this section is to provide a high-level understanding of the repair process, the potential impact on customer operations, and to highlight the state-of-the-art RAS characteristics of the System z.

15.1 Repair flow

When a hardware failure occurs, error data is collected by the hardware and firmware in the affected subsystems. This error data includes a list of all the parts that could possibly have caused the error. The error data is sent to Problem Analysis, resulting in an open problem. The list of all of the possible parts that could have caused the failure is included in the data for the problem. The list is ordered, from highest to lowest probability of being the cause of the failure with each part having a percentage probability assigned to it. The sum of the percentages for all the parts in the list should be 100%. Normally the SR is sent to the customer with enough parts, up to a maximum of three, which typically total a 90% or greater probability of fixing the problem. A typical list has one-to-five parts in it, which is a tiny fraction of the several hundred parts in an average system.

The general approach for all repairs is to exchange parts in the list, one at a time, starting with the highest probable part. In this context, exchange means remove an old part and replace it with a new one. Because of the extremely high reliability of the system, it is assumed that only one part is failing at any given time. After exchanging a part, a procedure is run to determine if the problem is fixed. Typically, this means to turn on the part so that it is actively being used and wait for the same failure to occur. Some parts also require running a diagnostic test, which is done using calls to APIs. If no failure is detected, the problem is fixed and the repair is complete. If a failure is detected, the problem is not fixed. The new part is removed, the original part is put back in, and the repair continues with the next part in the list.

The problem is closed when the repair is complete (the problem is fixed). At this point the data about the repair is transmitted via RSF to RETAIN. This data includes:

- ▶ Customer account identification
- ▶ System serial number
- ▶ Problem number
- ▶ System reference code generated by the failure
- ▶ Summary of parts exchanged (only for new parts left in the system), including the type of part, old and new part numbers, old and new serial numbers, and the position in the list of parts

This data is used for tracking purposes. All field part failures are accumulated in a database and continually analyzed for failure trends, for example, if a number of failures are seen in a particular type of part, all within a small range of serial numbers, it is an indication of a potential manufacturing problem with that part.

15.2 Customer impact considerations

One of the critical aspects of a repair is to minimize the impact on the customer's operations. R&V makes use of the inherent RAS capabilities of the system to accomplish this goal when exchanging parts. These capabilities include redundant power parts, alternate I/O paths, spare processors (CPs), and many other hardware availability options.

In this section, we provide an overview of repairs for subsystems that involve redundant (N+1) power, I/O, concurrent book repair (CBR), and non-concurrent repairs.

15.2.1 Redundant (N+1) power

In this case there is no impact to customer operations. The N+1 refers to the power design that includes one more power part than is required. If any 1 fails, then the remaining N can supply all the power that is required to keep the system running, for example, a processor book has three power supplies, but only requires two; if one fails, then the other two automatically increase their output to make up for the lost one. Therefore the system is now running on N power supplies, the +1 power supply can be taken off line and replaced with no impact to the customers workload (IBM calls this concurrent with customer operation).

This approach is also followed in the control network within the system. There are two network controllers (only 1 is required) in each processor book, each I/O cage and the CEC cage, that allow concurrent repair if one network controller fails.

When repairing a problem that involves N+1 parts, R&V checks, using an API, that it is safe to turn off the failing part and that the remaining parts are functioning normally. Usually, the failing part is already turned off. However, depending on the nature of the failure, the part might still be turned on and in use. If it is safe to turn off the part, then R&V continues with the repair by turning off the failing part, exchanging the failing part, activating the part (which restores N+1 operation), and verifying that the problem is fixed.

15.2.2 Input/Out

Many parts in the system have PCHIDs associated with them, which includes: I/O cards and cages, the cards in the processor books that are connected to the I/O cages, and the cables from the cards in the processor books to the I/O cages.

I/O cards can have from two to sixteen PCHIDs, and the cards in the processor books can have from eight to sixty four PCHIDs. When any of these parts are exchanged, the associated PCHIDs must be put in a Reserved Service state so that the operating system does not try to use these PCHIDs.

Ideally, the I/O configuration is set up so that there are multiple paths from each processor book to each I/O device in the network connected to the system—the same I/O devices can be accessed using multiple PCHIDs, which are referred to as alternate paths. In this case, whenever a part with PCHIDs is exchanged, R&V calls an API to automatically reroute I/O traffic to the alternate path. The PCHIDs for the part to be exchanged can then be safely put into Reserved Service state. There can be some loss of performance because of the increased I/O traffic on the PCHIDs on the alternate path, but typically this is minimal.

If an alternate path is not available, the affected PCHIDs must be taken offline and put in Reserved Service state before the repair can continue. Because this, in all probability, affects the customer operations because the SR must get approval from the customer to continue. Assuming that the customer approves, there are several options that are available to put the PCHIDs in Reserved Service state. These options are:

- ▶ R&V can call an API that the I/O subsystem provides that will automatically take the affected PCHIDs offline, and inform the operating system that the PCHIDs are no longer available. This choice comes with some limitations:
 - The system must be running with at least one partition IPL complete.
 - The operating system must support this option (z/VM and z/OS do, Linux does not).
 - The last path to a device is not varied off.

Therefore it is possible that even if this option is used, there might still be PCHIDs that are online and operating. In this case, one of the following options must also be used:

- ▶ The SR can request that the customer put the affected PCHIDs in Reserved Service state from the operating system, typically by using manual commands from the operating system console. The potential problem here is that each PCHID must be taken offline in each partition where it is defined because there can be many PCHIDs, each defined in up to 60 partitions, there might be hundreds of CSS.chpids to process. Therefore it is recommended that the previous option be used to vary off as many CSS.chpids as possible and whatever ones left online are varied off manually.
- ▶ Finally, R&V can call an API to the I/O subsystem to immediately vary off all the PCHIDs. While this is the quickest option, the operating system is NOT informed that the PCHIDs are no longer available and will likely try to use them. Depending on the operating system, and its tolerance of losing PCHIDs this way, this can lead to a lot of error handling and recovery by the operating system. Typically this option is used only during testing, when frequently no operating system is running.

After all of the affected PCHIDs are in Reserved Service state, the repair can continue. When the repair is complete, as part of closing the problem, R&V displays a list of the PCHIDs that were put into Reserved Service state with a reminder that they might need to be put into Online Operating state. If the first option was used, the I/O subsystem knows what PCHIDs it processed, and R&V can call an API to the I/O subsystem to put these PCHIDs back to the Online Operating state. Other than this, however, there is no API that is available to R&V to put these PCHIDs back to the Online Operating state and it must be done manually by either the SR or the customer.

15.2.3 Concurrent book repair (CBR)

In this case, a processor book must be exchanged while the system is running. Processor books contain the actual processor chips and the memory for the system. An analogy is to repair a broken piston in the engine of a vehicle while driving on an interstate highway without any loss of horsepower, which is only possible if there are two or more processor books in the system. This is done by using spare resources on other books, including CPs, I/O, and memory. Then the failing processor book can be fenced (isolated) from the rest of the system, exchanged, and then put back into the system. R&V calls a set of APIs to move the resources in use to these spare resources, for example, a CP that is running is suspended, and the jobs it was running are transferred to a spare CP in another processor book. Data in memory is transferred to spare memory on another book. I/O resources (PCHIDs) are switched to an alternate path.

It is possible that there are not enough spare resources. Perhaps the processor book that is being exchanged had 12 CPs running, and there are only eight spares available. In this case, some performance capability is lost, and typically if this is the case, the repair is scheduled when the system load is the lowest. Also, in the case of I/O, if an alternate path is not available, the affected PCHIDs must be put in the Reserved Service state.

Non-concurrent repairs

Some parts can never be exchanged while the system is running. These include the CEC cage backplane that the processor books plug in to, some I/O cages, and processor book parts in a system with only one processor book. In these cases, it is necessary to perform the repair with power off. If the system is running, R&V directs the SR to gracefully shut down the operating system. After multiple confirmations that the operating system is shut down, R&V calls the APIs to turn off power to the system and the repair can continue.

There is an additional check on any running system before powering it down. R&V checks via an API to see if the system is a timer source for the Server Timer Protocol (STP) function; if so, there is an additional warning panel and confirmation displayed.

15.3 Other options

In addition to repairing open problems, R&V has two other options. One of these options allows the SR to display a list of the parts in the system and exchange any one of them. When a part is selected for exchange, R&V opens a problem itself with the selected part as the list of parts associated with the problem. From this point, R&V handles the repair of the problem as though it were a normal problem that Problem Analysis opened (It is closed and data about the repair is transmitted to RETAIN).

This option is typically used when:

- ▶ An open problem was closed before it was fixed (R&V only works on open problems).
- ▶ A failure is caused by a part that is not in the list of parts that is associated with the problem (very rare occurrence).
- ▶ There is a defect discovered in a particular type of part that makes it likely to fail, and we want to exchange these parts in all machines before the defect causes a failure. The intent is to purge the defective parts from all customers in a controlled manner to minimize the effect on customer operations.

The second R&V option is used to report the repair of a failure that does not have an open problem associated with it, which is a rarely used option. Originally it was provided to handle cases, such as the replacement of the SE itself at a time when there was only one SE in the system. In this case, all previous error log data on the SE hard disk is lost, and there is no record of the original problem after installing the new SE. The option allows the SR to specify what parts were exchanged, and then, similar to the previous option, opens a problem itself with the selected parts as the list of FRUs for the problem. This open problem is then closed and data about the repair is transmitted.

15.4 Sample sequence of panels for a repair

Figure 15-1 on page 320 through Figure 15-16 on page 335 show a typical sequence of panels that the SR sees during a repair. The failing part (known as a Field Replaceable Unit, or FRU) in this case is a fan controller module. The fan controller modules are known as Motor Drive Adapters (MDA).

Figure 15-1 on page 320 shows the panel where open problems are listed. In this case, there is only one open problem.

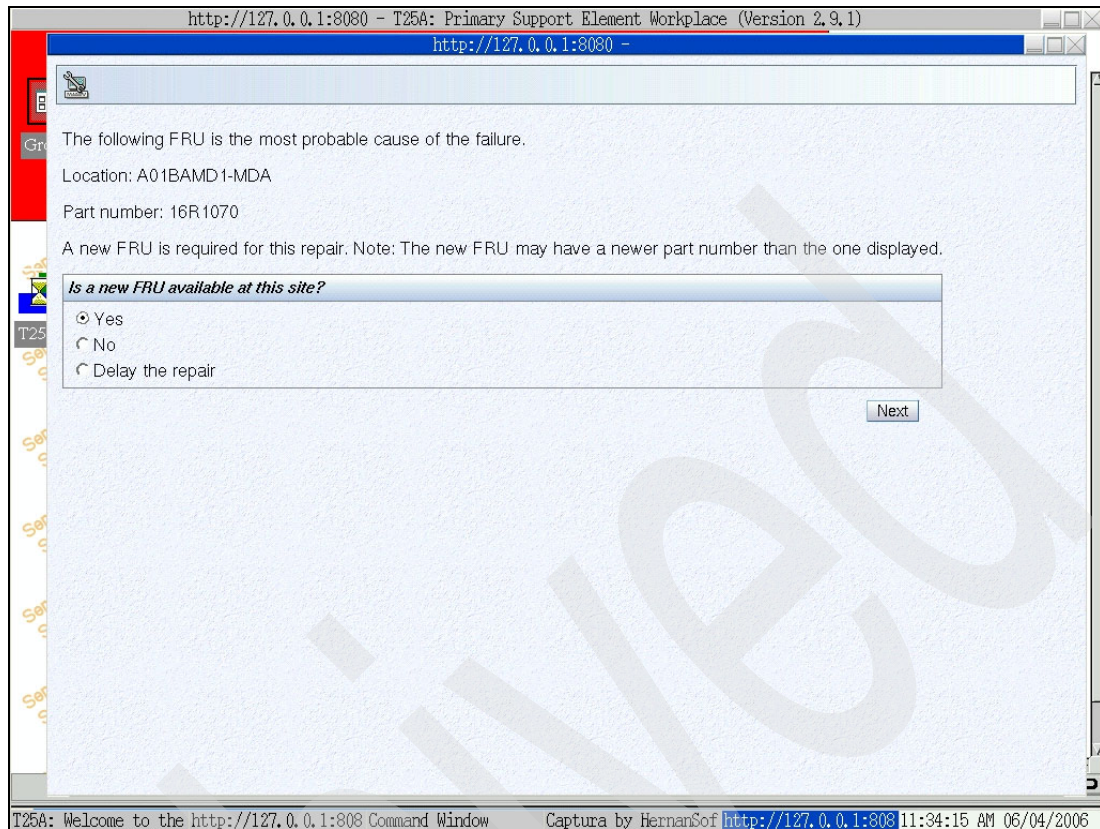


Figure 15-2 Checking availability of new part

After the SR confirms the availability of the new part, R&V displays a confirmation panel, shown in Figure 15-3 on page 322, that includes data for the FRU that will be exchanged with the new part.

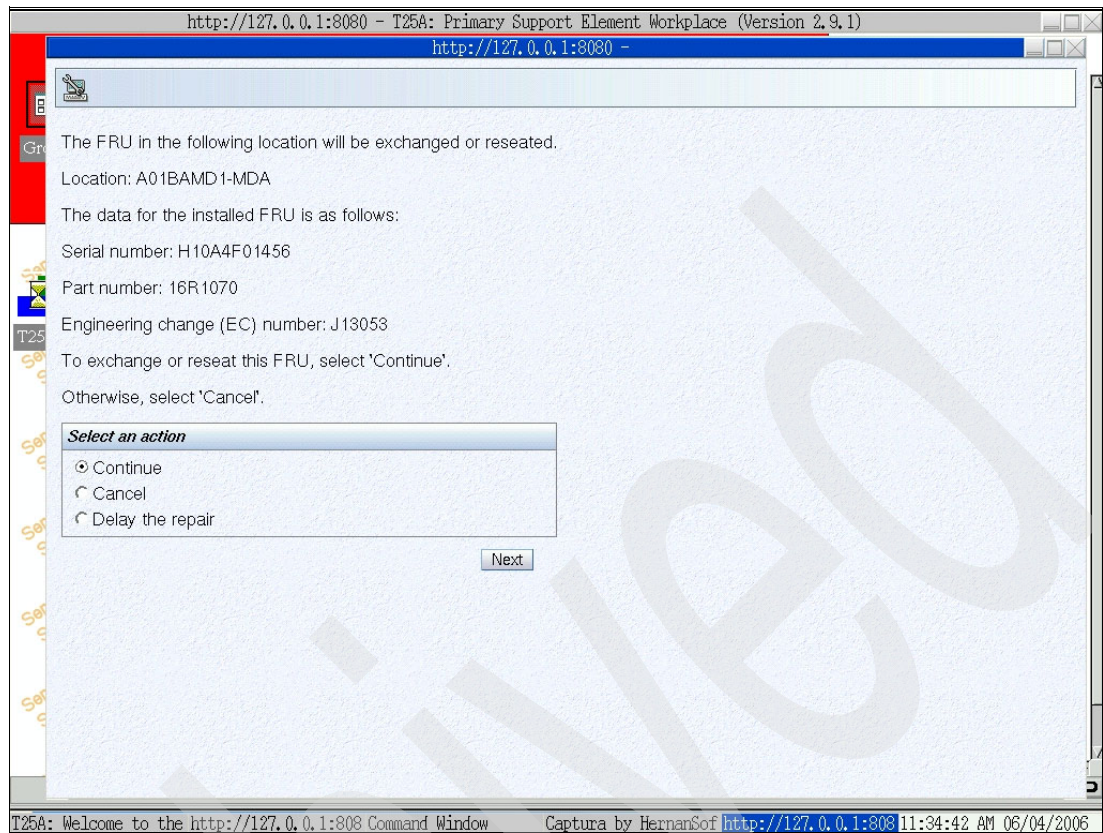


Figure 15-3 Confirm continuation with selected part

Whenever R&V exchanges a part, it first writes some data to the SEEPROM on the FRU that is being replaced. This data includes the machine type and serial number of the system that the FRU is in and error data from the problem, such as the system reference code and problem number. IBM uses this data when the part is returned for analysis. This data is used to determine failure rates for parts or to identify a bad batch of the same type of parts (based on serial numbers of failing parts). While this data is being written, R&V displays a busy message, as shown in Figure 15-4 on page 323.

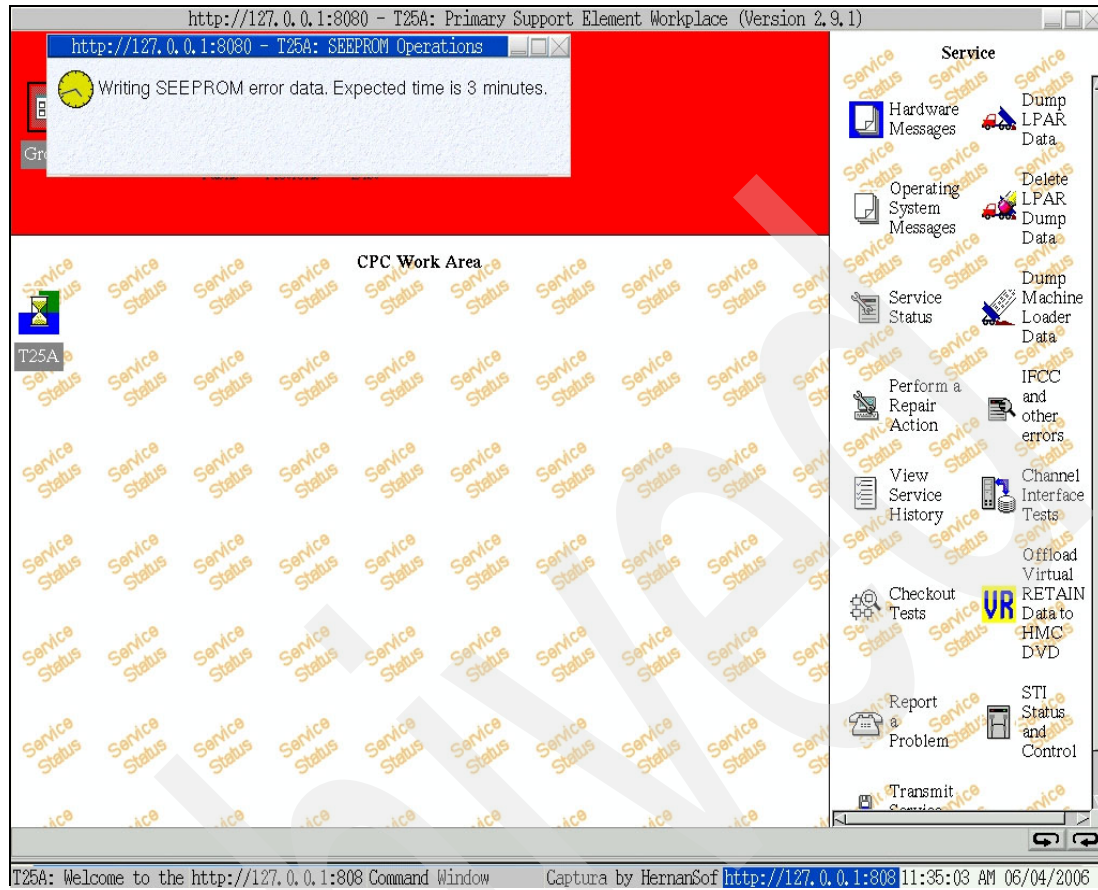


Figure 15-4 Busy message while writing data to FRU

After writing data to the SEEPROM, the part is deactivated or powered off. Figure 15-5 on page 324 shows the busy message that is displayed while the part is deactivating.

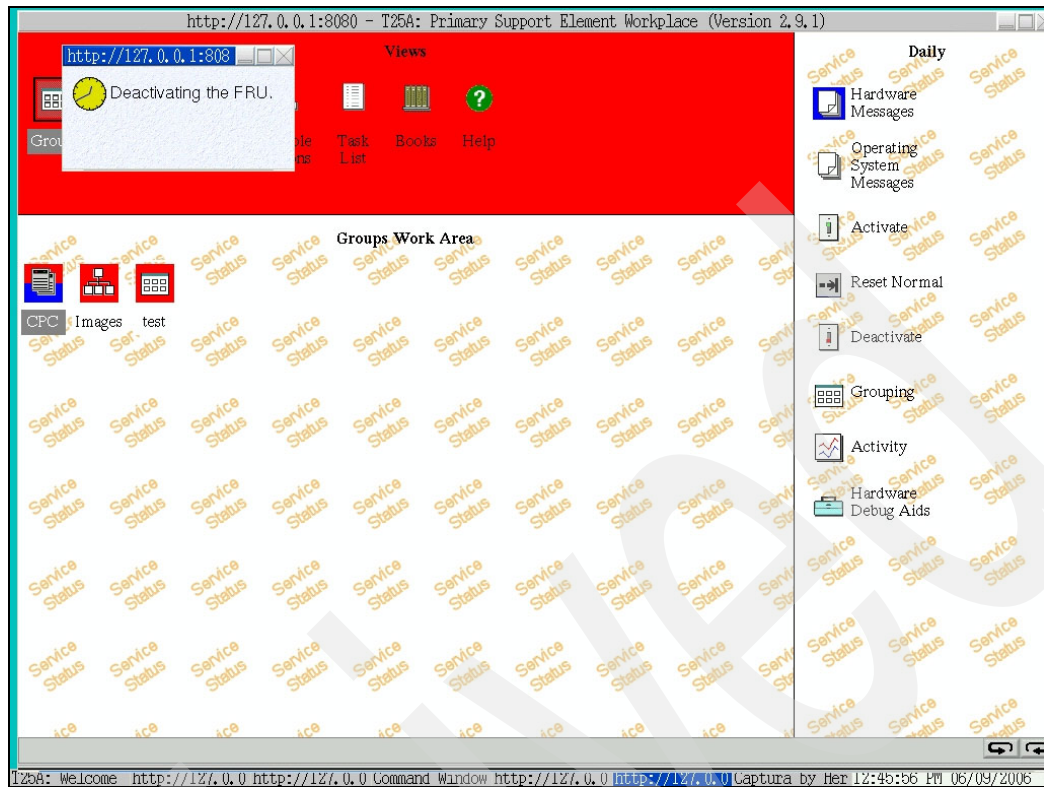


Figure 15-5 Busy message while deactivating FRU

Whenever a power part is involved, for safety reasons, all indicators on the part must be off and the SR must confirm that all indicators are off. Figure 15-6 on page 325 shows the confirmation panel.

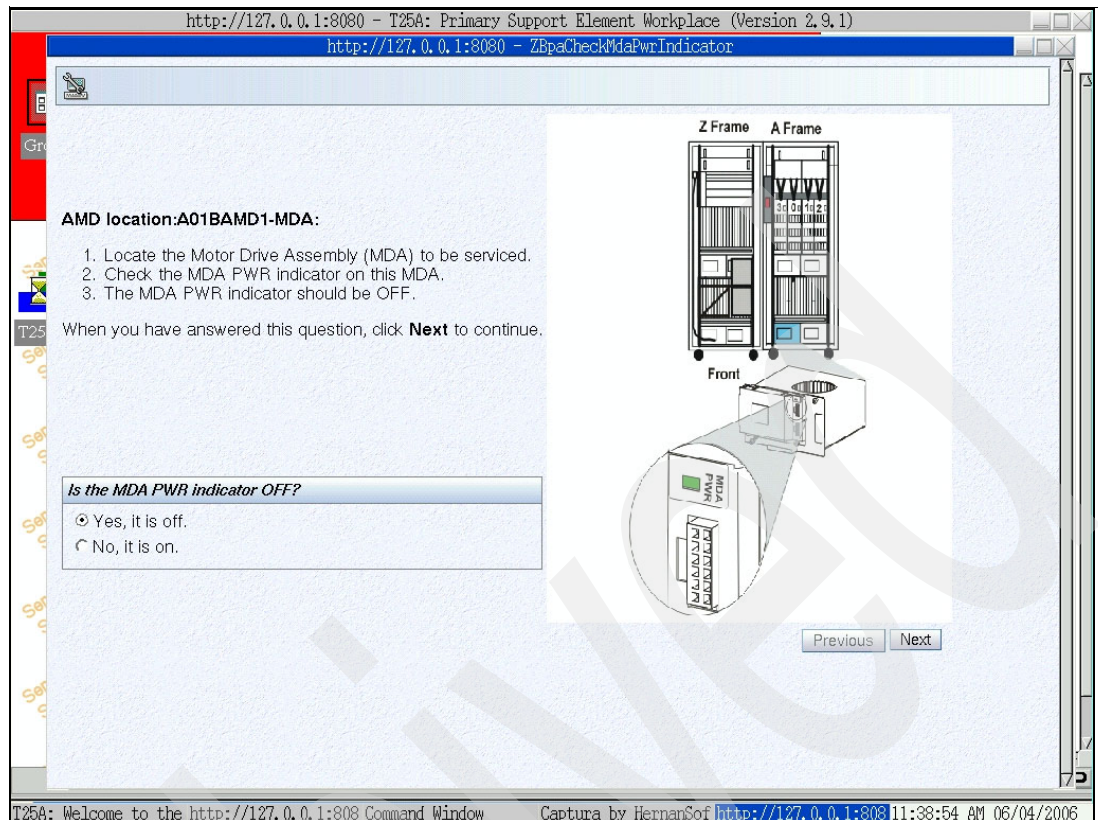


Figure 15-6 Confirming FRU is powered off

At this point the FRU is powered off. Now, R&V displays a panel that guides the SR to find the failing FRU in the system using graphics and text to do so, as shown in Figure 15-7 on page 326.

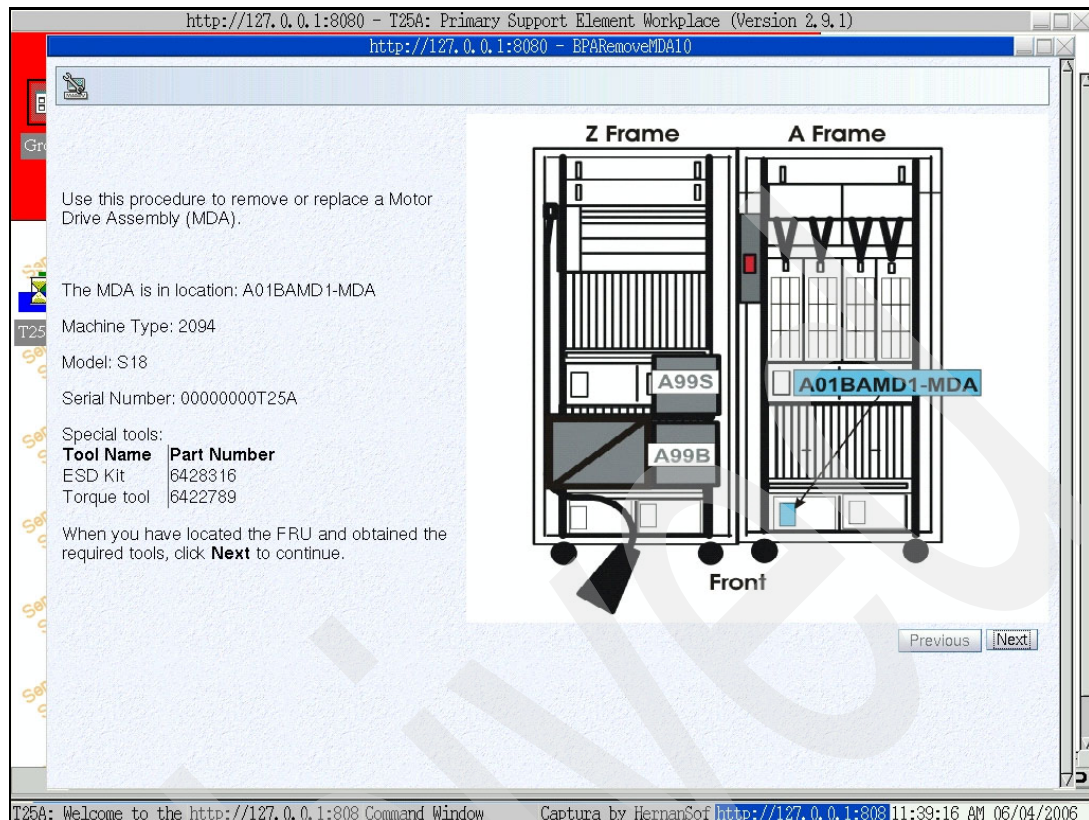


Figure 15-7 Locating the FRU

Before the SR actually touches anything, a warning panel is displayed, shown in Figure 15-8 on page 327, with general instructions about handling electronic parts. These parts can be damaged by static electrical discharge. The SR uses a conductive tether (the ESD wrist strap) that connects to the machine to prevent any buildup of electrical charge.

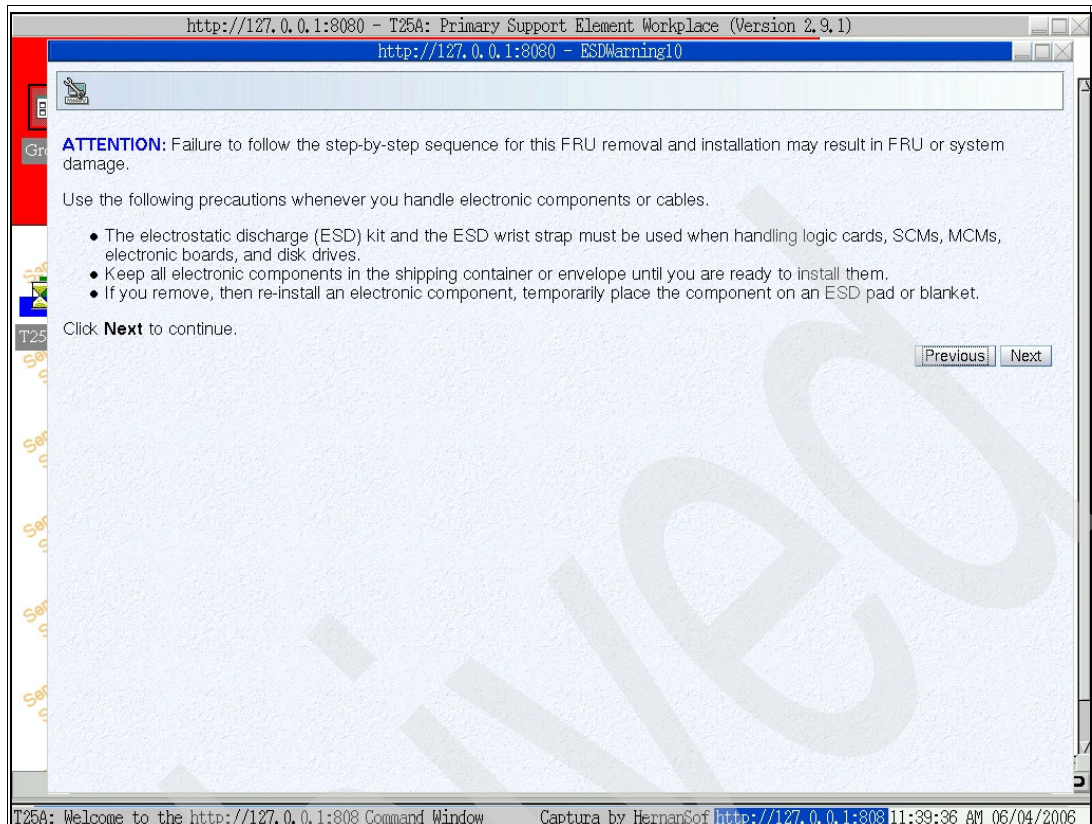


Figure 15-8 Generic part handling

The panel, shown in Figure 15-9 on page 328, is displayed to guide the SR in removing the failing FRU.

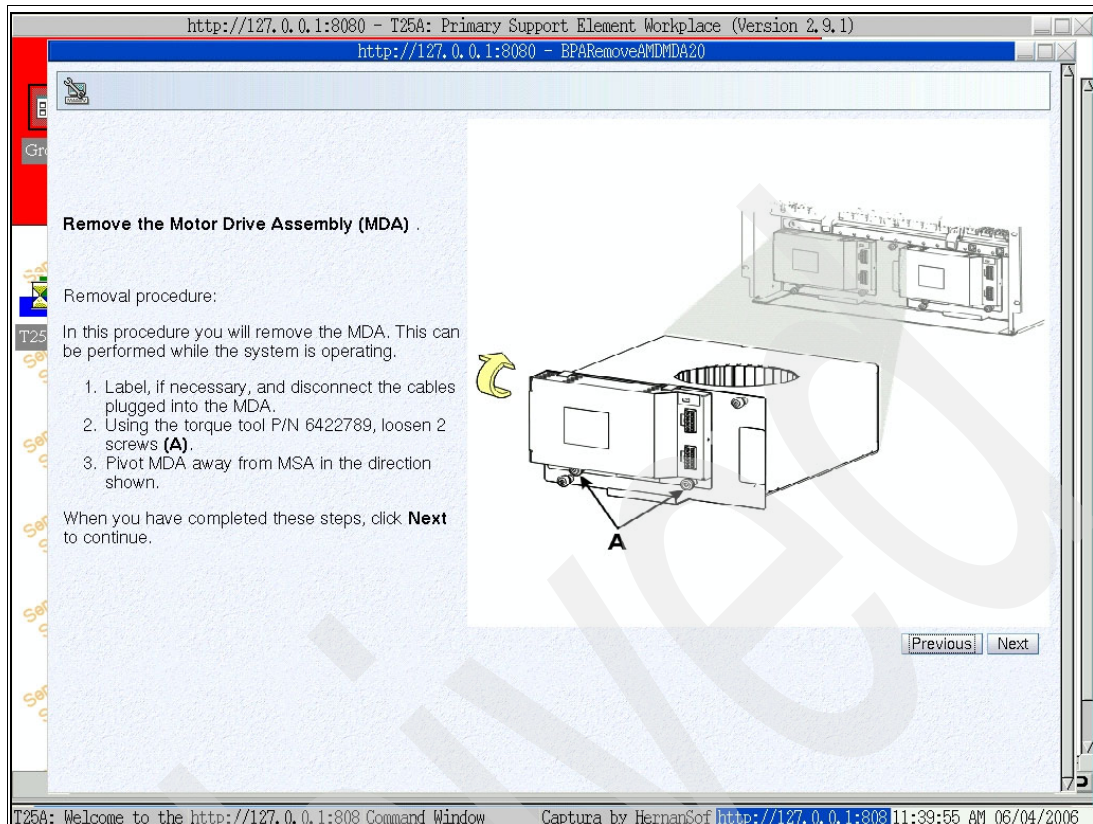


Figure 15-9 Removing the failing FRU

Next, a panel in Figure 15-10 on page 329 is displayed to guide the SR to install the new FRU.

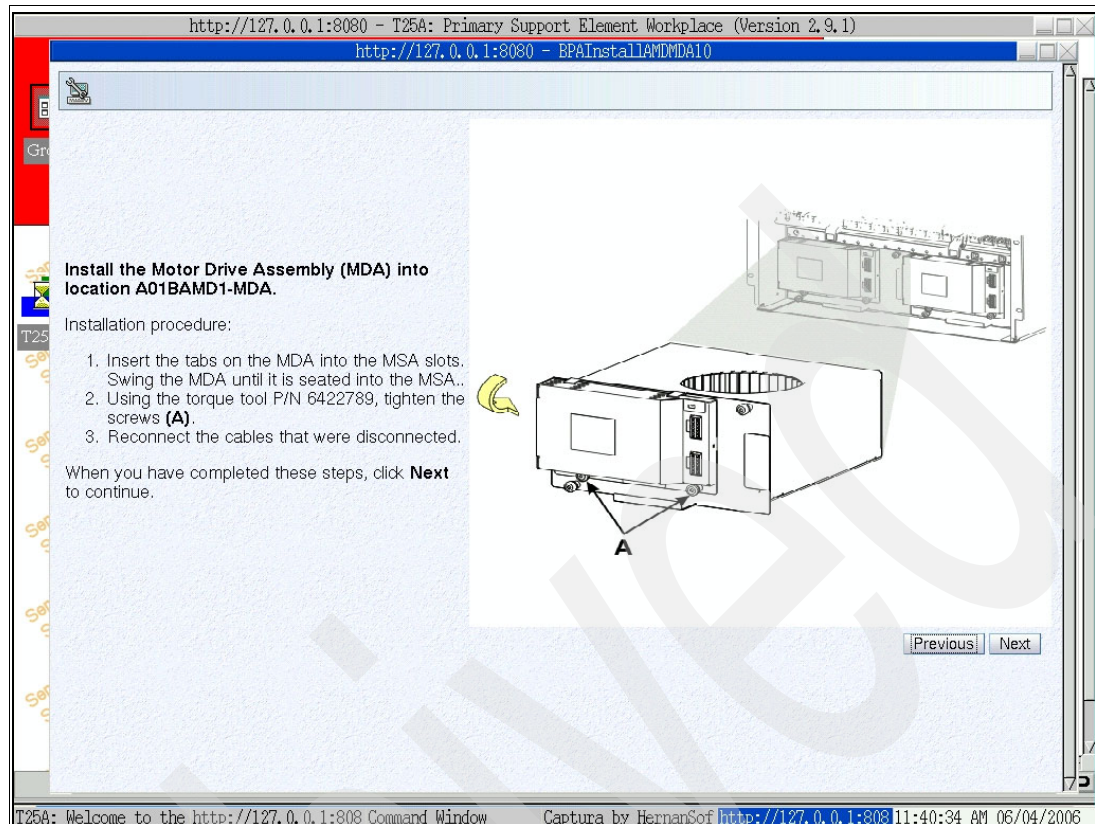


Figure 15-10 Installing the new FRU

After the new FRU is installed, it is activated (powered on) automatically. Figure 15-11 on page 330 shows the busy message that is displayed while the FRU is activating.

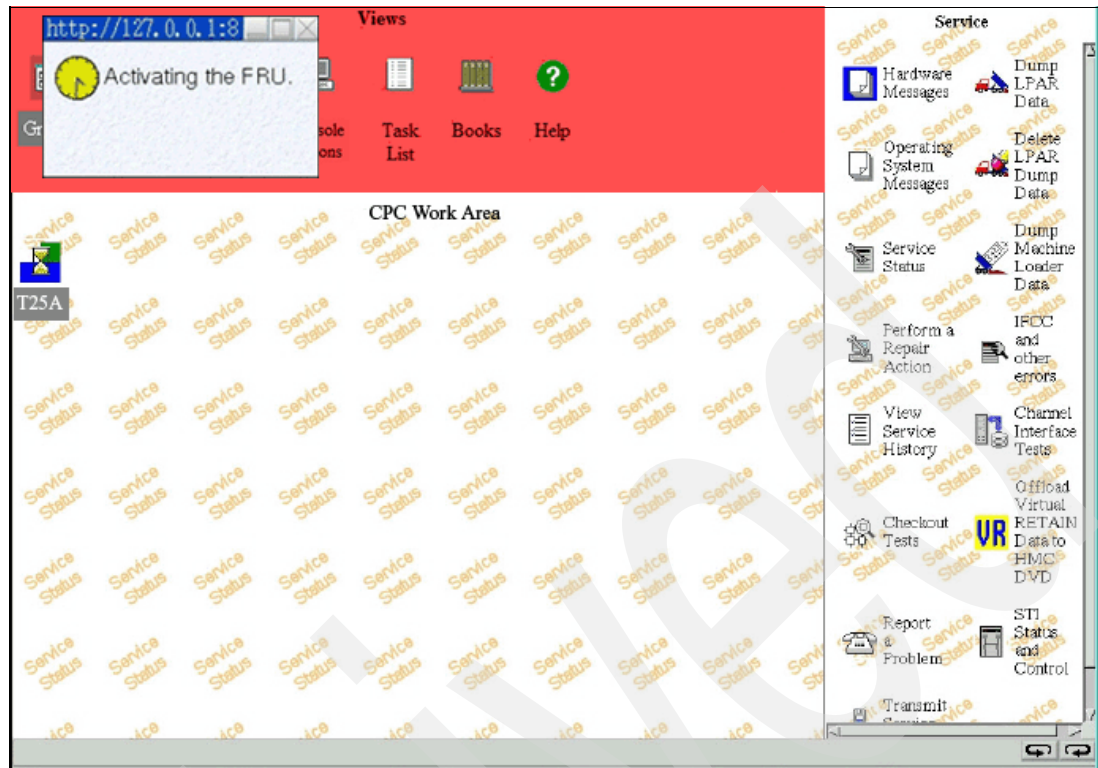


Figure 15-11 Activating the new FRU

After the FRU is activated, R&V verifies if the problem is fixed by waiting three minutes for any errors to occur. If no errors occur within this time, R&V assumes that the problem is fixed. During this time a busy message is displayed, as shown in Figure 15-12 on page 331.

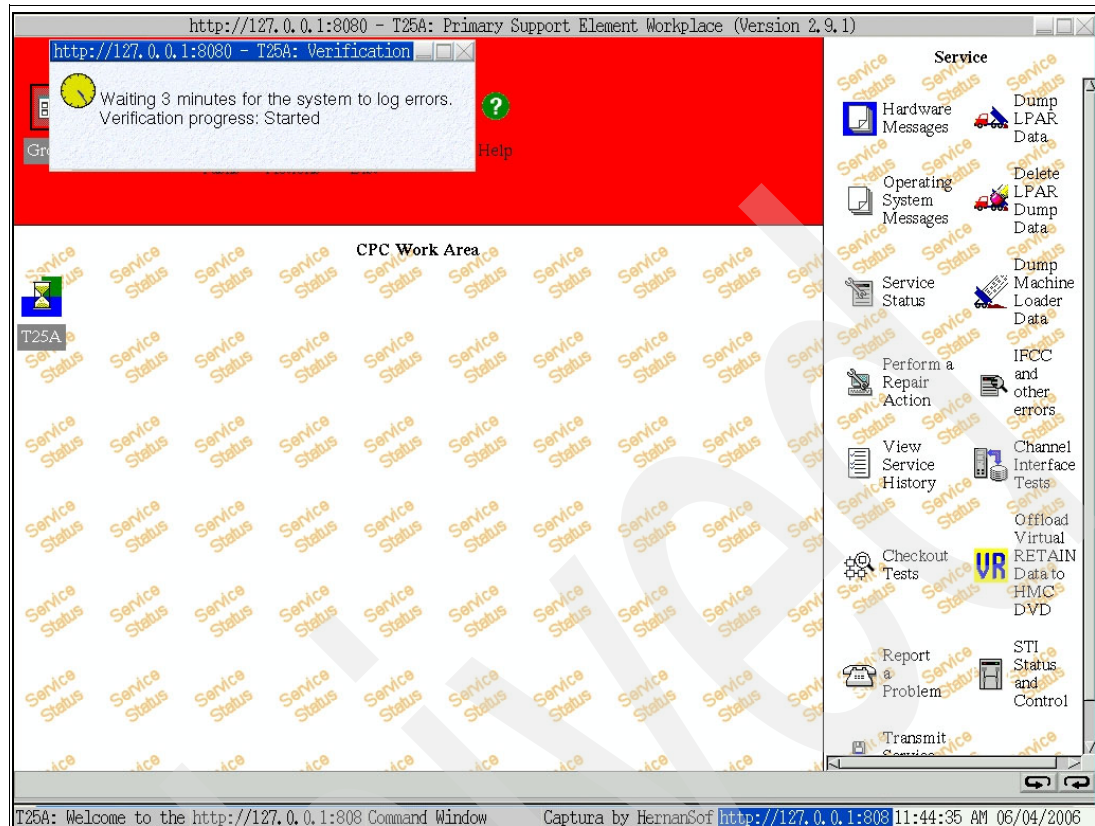


Figure 15-12 Verifying the problem is fixed

In this case, we assume that the problem was fixed by the new FRU. Figure 15-13 on page 332 shows the panel that is displayed confirming that the problem is fixed.

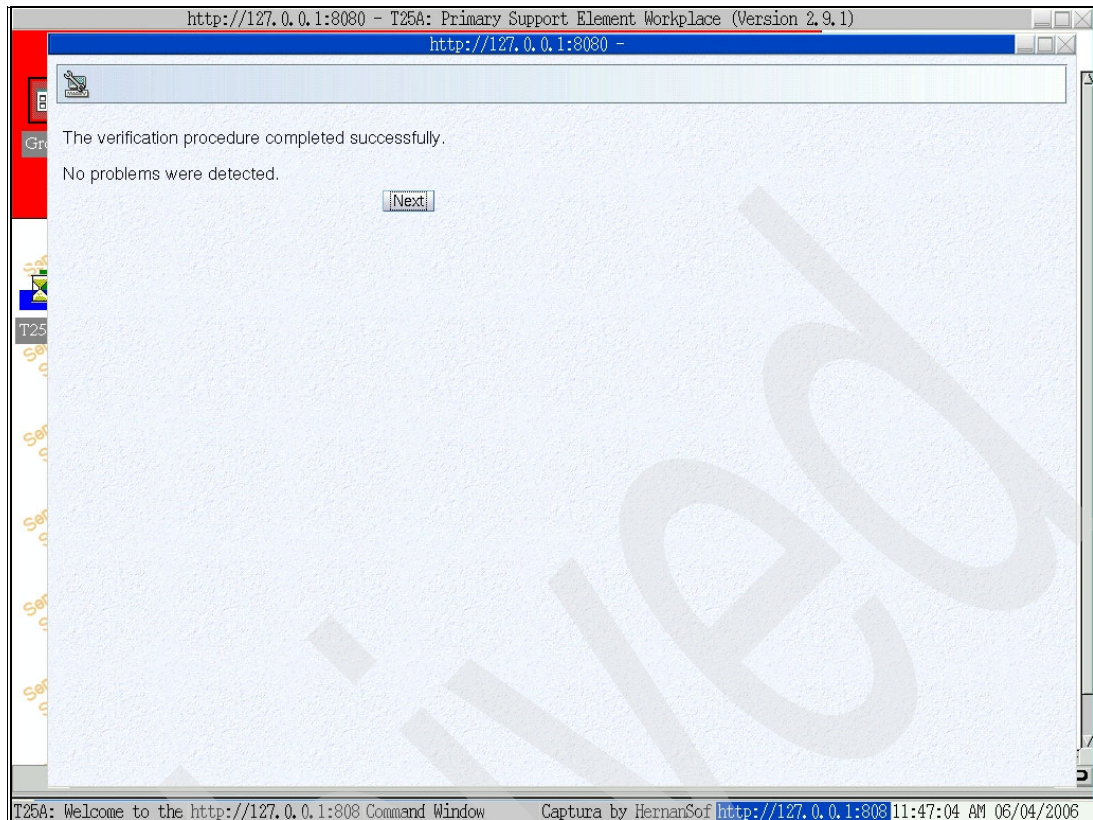


Figure 15-13 Verification completes successfully

Figure 15-14 on page 333 shows the panel that allows the SR to enter a description of the repair. This step is optional. Detailed data about the repair is automatically recorded in the logs in the SE.

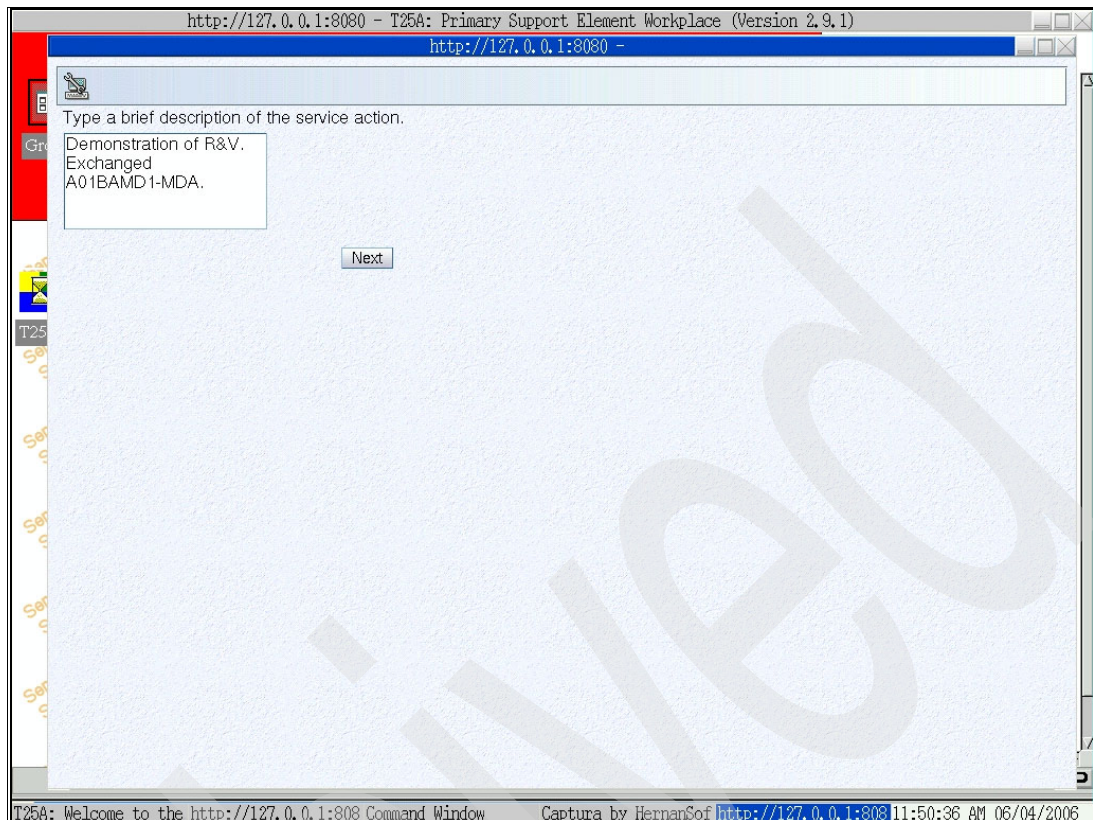


Figure 15-14 *Entering a description of the repair action*

At this point, the problem is closed. It is also possible that there are other open problems that are related to the recently closed problem, for example, a power failure while the system is running results in one open problem. If the customer attempts to power on the system, another error might be logged that results in another open problem. The panel in Figure 15-15 on page 334 guides the SR to check that any other problems, that are due to the same failure, are closed since they do not require further repair.

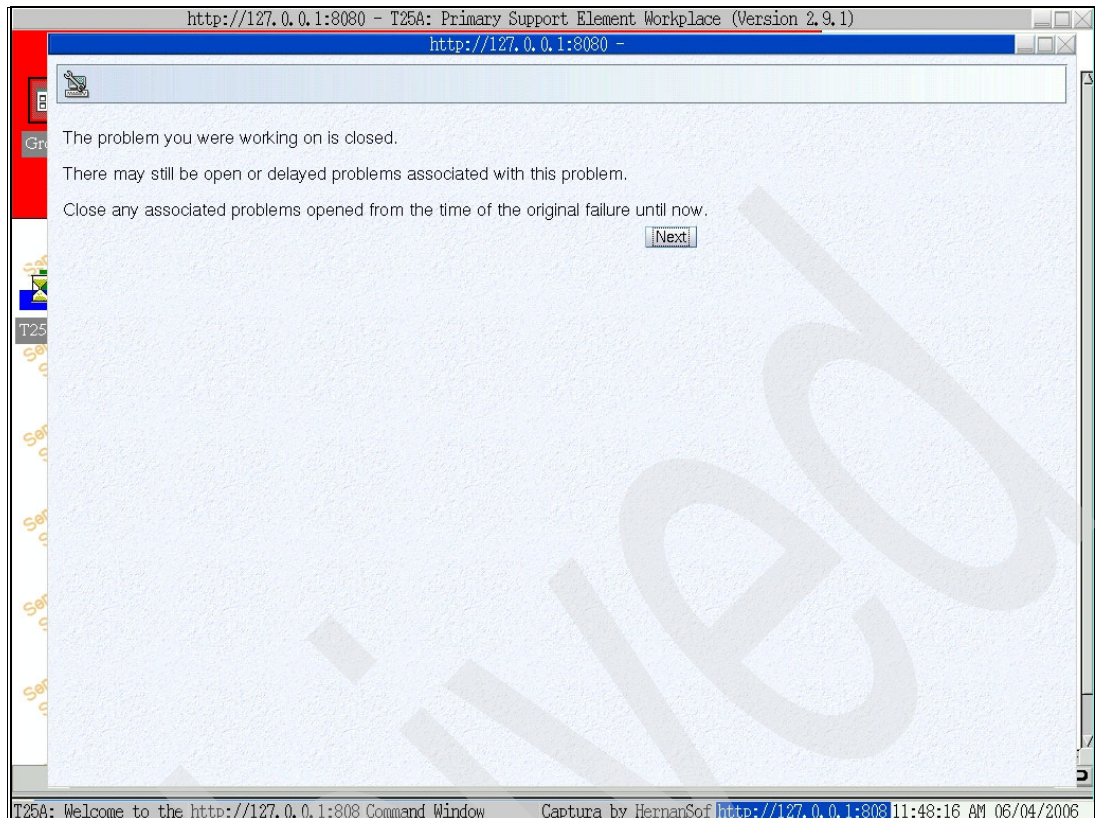


Figure 15-15 Problem closed

Finally, R&V checks for any open problems. In our example, there was only one open problem, and this problem is now repaired and closed. Figure 15-16 on page 335 shows the panel that is displayed that informs the SR that there are no open problems.

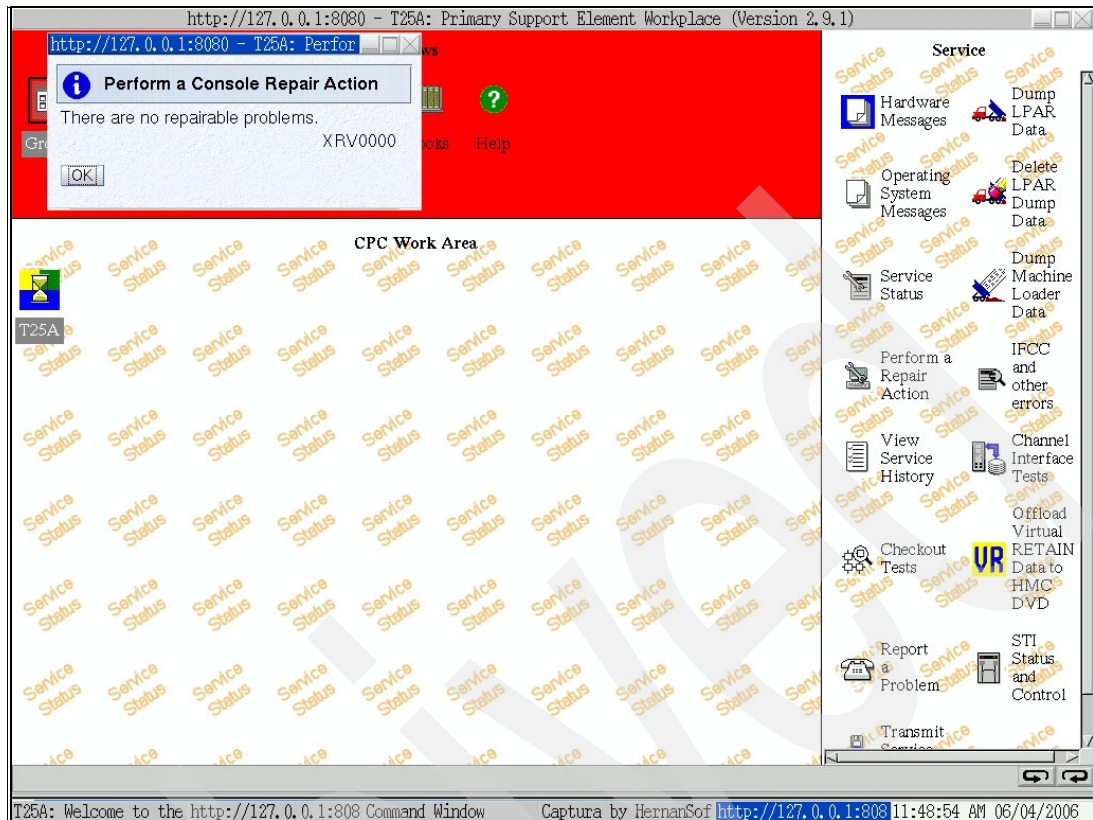


Figure 15-16 No open problems

15.5 Questions and discussions

1. The HMC panels for a repair action seem to progress in baby steps. Suggest some reasons for this.
2. How much time and effort do you think is involved in developing the material behind these panels?
3. Would many of the detailed drawings be better placed in books (in “hard copy”)?

Archived

Chapter keys

A.1 Chapter 1

1. *Multiprocessor* means several processors and that these processors are used by the operating system and applications. What does *multiprogramming* mean?

Multiprogramming refers to time sharing or switching among multiple threads or tasks that exist within the computer system. This can be done with a single processor or with multiple processors.

2. Usually, each read or write operation on a non-shared DASD device is one I/O operation. How many I/O operations are involved when the DASD is in a shared mode?

Each system that is sharing the DASD device can have an I/O operation started, but in older DASD devices only one of the processors can actually connect to the DASD device at a given instant. The other system users are queued. Modern DASD units can actually have several I/O operations in progress at the same instant, which involves cache and logical locking designs that involve only part of the data space on a disk.

3. What is the role of PR/SM?

PR/SM is the IBM-designed mechanism that allows and controls partitioning. The PR/SM facilities permit the definition of LPARs and so forth.

4. What changes are needed for z/VM applications to work in an LPAR?

No changes are needed. For normal application-level programs, and for almost all system-level programs, such as operating systems, the difference between a whole computer or an LPAR is undetectable.

A.2 Chapter 2

1. How does the HMC participate in control of operating systems, such as z/OS, z/VM, or Linux for System z?

In normal circumstances, an HMC is not involved in day-to-day control of individual operating systems (which are each running in their own LPAR). The HMC is involved in creating LPARs but these definitions are seldom changed in most situations. The HMC is

involved in starting each operating system through the process known as IPLing or LOADING the operating system. After they are started, operating systems are managed through their own operator console sessions; however, this is not completely true for Linux, which does not have a well-defined sense of an operator console.

2. Who should have access to the HMC?

Very few people. An HMC has logon security controls but, in practice, most HMCs are permanently active. A malevolent person (or someone who is merely curious) can crash all of the operating systems and applications running on all the CECs that are connected to that HMC.

3. What HMC functions are important to Application Programmers?

No HMC functions are directly related to Application Programmers or their applications. Indirect relations exist, for example, an HMC operator might enable additional channel paths to unusually busy devices or to bypass failed paths.

4. What is an HMC, in hardware terms? What programs or operating systems are involved?

AN HMC is a personal computer that runs a customized version of Linux. It is a *closed system*, meaning that user applications cannot be added to it.

5. What are key differences between an HMC and a SE?

Perhaps the most obvious difference is that an SE (or the two SEs in every CEC) are used only by that CEC. An HMC can connect to, and control, multiple CECs. It is possible to switch the HMC to single image mode in which it largely duplicates the console image and functions of the SE that is installed in a specific CEC.

SEs are physically inside a CEC, where they are not normally seen. They are usually closed, where their display panels cannot be seen, even with the CEC covers open. In contrast, an HMC is openly placed in an operator control area, typically with a large display that is always operational.

A.3 Chapter 3

1. What significant control functions are exclusive to the tree view? The classic view?

In principle, there are no differences between the actions that are available through the two views. The HMC operator can use either one.

2. What high-level markup language is much involved in HMC implementation?

XML is heavily used.

3. How difficult is it to expand an area of HMC panels?

For someone with a good grasp of the current HMC implementation, adding new panels (with their associated XML definitions) is relatively easy.

4. How much XML skill does an HMC user need? Why did we provide this overview?

The HMC user is completely unaware of the underlying implementation and requires no XML skills. The overview in this chapter provides some insight into the design of the HMC application.

A.4 Chapter 4

1. This chapter is about user management. Which users are we considering?

We only discuss HMC users who are usually senior operators or Systems Programmers. We do not discuss normal mainframe users.

2. We sometimes see elaborate password rules that might require at least one number, at least one special character, a mixture of upper case and lower case, prohibit more than two sequential characters that are adjacent on the keyboard, and so forth. What are some of the problems with these rules?

Care must be taken to not decrease the *key space* in a significant way because it makes brute force attacks easier. It is difficult to find a balance between (1) forcing users to avoid really trivial keys, and (2) maintaining the largest key space possible.

3. Oddly enough, many HMC systems have practically no encryption or firewalls or passwords involved. Why is this?

Many HMCs are in physically secure areas, on private LANs that are not accessible outside of that area.

4. Who establishes new HMC user IDs, sets passwords, determines privilege sets, and so forth? Is this the same person who performs these services for the general user community?

There is no self-defined answer for this. The person who performs these actions, in a sense, holds the keys to the keys and must be a trusted employee. The HMC user group is typically a small group and there might be only one security administrator for this group. It is important that the passwords for the administrator user ID be available in a secure storage location.

In any but the smallest installation, the administration of user IDs for the general user population would almost certainly be done by another group.

5. In a typical mainframe environment, would at least one HMC user be normally logged on to the HMC?

Probably not. The actions that are performed through an HMC (such as IPLing the operating system in an LPAR) are not often needed. In a stable situation, there can be days or weeks when HMC functions are not needed. The system operators (some of whose functions include HMC operation) are routinely more involved with the operating system consoles, such as MVS operator consoles.

6. Can the HMC console be combined with, for example, the z/OS operator console? How is security assigned in this situation?

The z/OS operator console, often known as the MVS operator console, currently requires a local 3270 session, which cannot be provided on an HMC console. There are ways to perform z/OS operator functions from an HMC terminal (using 3270 TSO sessions or the hardware console function), but these are very unlikely to be used in routine mainframe production environments. There is no overlap between z/OS operator console security (which is an interesting topic, in itself) and HMC functions other than the unusual case that we just mentioned where HMC communication functions (3270 emulation or the hardware console) are used.

A.5 Chapter 5

1. Cryptography discussions are often interesting. Are there any unbreakable encryption methods? If so, why are we so concerned about breaking ciphers?

In theory, one time pads, or the equivalent automated techniques, are unbreakable. They are impractical for HMC usage or for most common purposes. For practical purposes, a reasonable symmetric cipher (based on practical key lengths) is needed. In principle, such encryption schemes are breakable, but possibly only with very large efforts.

2. DES (56-bit) is still widely used. What is the practical degree of exposure in this?

In theory, 56-bit DES is breakable with today's fast computers, but it requires a concentrated, skilled effort. Many current LAN protocols involve temporary DES keys that are set through public key negotiations. Breaking a DES key is meaningful only for that one session. Extremely sensitive links, such as might be used by major banks or defense establishments, should worry about DES. More routine installations are likely to have many more potential security exposures to worry about before coming to potential DES exposures.

3. Why is public key cryptography not used for all communication?

Encryption and decryption with current public key schemes requires intense computation to handle only a few bytes of data. In practice, it is better to use public key cryptography to exchange only symmetric keys or something similar, such as identity certificates, and then use much faster symmetric methods for data exchange.

4. Remote access to the HMC functions appears to be very convenient. Why is that not every installation uses it?

Access to an HMC can completely disrupt all of the mainframes that are managed by that HMC. It is possible to provide secure remote access to the HMC if all the security controls and passwords are properly managed. Some installations might have doubts about these controls, for example, despite all rules to the contrary, some people still write down their passwords near their terminal. The HMC functions are so critical that some installations simply do not want the potential exposures created by remote users. These concerns might be a little paranoid, and the use of remote HMC operation is spreading.

A.6 Chapter 6

1. One of the figures contains a token ring LAN. Where are these used? What are their advantages? Disadvantages?

IBM developed token ring LANs, which have performance advantages over Ethernet LANs when many hosts are involved. Due to IBM management (or mismanagement) of token ring licensing issues, Ethernet became more popular especially after hub style Ethernet became available. Token rings were used for HMC LANs in earlier years, but are no longer used.

2. A significant number of HMC installations ignore all of the LAN security guidelines that we discussed here, but they are not considered insecure. How is this possible?

Many mainframe installations have their HMCs on a private LAN that exists only within their physical mainframe installation boundaries. There is no way for an outsider to connect to these LANs.

3. What is unique about IP addresses 192.168.xxx.xxx? IP addresses 10.xxx.xxx.xxx? Why are many of the examples using IP addresses 9.xxx.xxx.xxx?

The 192.168.xxx.xxx and 10.xxx.xxx.xxx are defined as non-routable. They will not pass through routers and, consequently, cannot be used for general connectivity. They are intended only for a local LAN.

IBM was assigned 9.xxx.xxx.xxx addresses, and many examples of IBM IP addresses use this range.

4. IPv6 has been available for some years now, but is not used much (yet). Why?

Some years ago it was observed that the world was quickly running out of IP addresses. This was quite true for two reasons:

- Internet and Web usage was growing very rapidly.
- 32-bit addressing, especially with the number of unusable or poorly used addresses due to existing structuring of 32-bit addresses, was clearly insufficient.

Many changes in LAN technology have occurred over the last two decades. One change was the advent of Network Address Translation (NAT) functions, which allows an ISP (for example) to assign a “real” IP address to a router and then have that router respond to many hosts that are assigned non-routable addresses (usually in the 192.168.xxx.xx range or the 10.x.x.x range). With some care, NAT functions can be cascaded, which greatly reduced the immediate need for more IP addresses for ISPs.

A.7 Chapter 7

1. Does the HMC call home only when there is a problem?

No. It can routinely send statistical information to IBM at other times.

2. Some installations cannot permit any outside system connections. How is this situation handled?

There are a number of ways to handle this. Your instructor can discuss some of these situations.

3. Customer data (from applications) is not sent through the RSF facility, but it might be possible to derive competitive information from the statistical data that is sent through RSF. How is this exposure handled?

All data sent through RSF is encrypted.

4. In a normal mainframe environment, might the HMC call home without the customer being aware of the action?

Yes. This includes the statistical data we previously mentioned. It also includes situations where a component failed and the mainframe transparently activated a spare element. The customer might be unaware of the failure until an IBM service person arrives (based on the call home report sent for the failure).

A.8 Chapter 8

1. How much of the information in this chapter is needed by Application Programmers?

In general, none.

2. Are all Systems Programmers expected to be completely familiar with HMC profiles and IOCDS preparations?

In a larger installation, this is a specialized area. Most Systems Programmers are somewhat familiar with it, but the actual setup is often left to one person or a smaller group.

3. Partitioning can be complex. What percentage of mainframe installations actually use it?

All IBM mainframes use it today. Until a few years ago it was possible to operate in *basic* mode without partitions, which was so seldom used that the function was discontinued.

4. How isolated is one partition from another partition? What are the data security implications?

The System Programmer who creates image profiles can allow a certain amount of cross-image communication, normally used to permit higher-level measurements and optimization adjustments by functions, such as the z/OS Workload Manager. If the relevant image profiles do not allow this communication, there is a very strong isolation between LPARs. However, it is common to allow multiple LPARs to see the same disk storage (shared DASD, which is not required, but it is very common. Of course, this allows the multiple LPARs to access the data on the disks.

5. With appropriate planning and manual operator intervention, real memory can be moved from one partition to another. What are some of the complexities involved? How often is this done?

This is not commonly done. The problem is that areas of memory might be fixed (or pinned, using Linux terminology, allowing the software to use the real addresses of that memory), meaning that those areas cannot be readily shifted to other pages of real memory. The most recent System z architecture has an additional level of memory address redirection that provides assistance in this area.

A.9 Chapter 9

1. The SNMP design for reporting seems primitive and confusing. Why do you think this came about?

There are probably many reasons. At the time the SNMP design was considered, circuits to implement the design were much less sophisticated than what would be used today. The SNMP reporting design was meant to be extended to small terminals and other devices, the cost of implementation was a significant factor. With today's very inexpensive one-chip microprocessors, this is not much of a factor, but it was a major factor at the time of SNMP design.

2. Java seems to mix well with SNMP. Were they designed for this?

Java is a much later development than SNMP. That they mix well is both a happy circumstance and a tribute to the adaptability of Java.

3. Why are automation and SNMP grouped together?

SNMP is intended to provide an automatic, under the covers, system of reporting and statistics gathering, which is largely unseen by anyone after it is operational.

4. How much of this material does the average System z Systems Programmer see? An Advanced Systems Programmer?

It would be highly unusual for even the most advanced System z Systems Programmer to become involved in SNMP reporting used by the HMC. It is included in this document to better explain some of the internal workings of the HMC. Some third-party vendors (and a

some advanced customers) do use these interfaces to create additional automation software products.

A.10 Chapter 10

1. What is special or unique about this problem reporting and handling design?

To some extent, this design is for a machine (the System z complex, including the HMC) to report to another machine (or multiplex complexes of machines) at IBM service locations. Significant System z failures are a very small part of the reporting data. Statistical information about soft failures (in which the System z automatically handled the failure) is important for future mainframe designs.

The design also allows separate service people, at different locations, and at different times to coordinate actions for a failure.

2. What is the link between the HMC and this problem handling design?

The HMC provides the data gathering point and the communication link between the System z and the IBM service complexes.

3. If System z machines seldom fail, why is such an elaborate problem reporting and handling scheme needed?

Again, statistical information about soft failures is important to IBM. These also provide look ahead trends to help predict when a component should be replaced before a hard failure occurs.

As an example, larger System z machines usually have spare processors. If a processor fails, and then fails all the build-in retry and recovery functions, a spare processor automatically replaces the failed processor. In most cases, Application Programs are unaware that this happened, and the System Operators are probably unaware that it happened. The HMC reporting system informs IBM of the event, allowing a decision to be made about replacing CPC elements.

It is not uncommon for System z service people to schedule appointments to replace System z components when the System z customer was unaware that anything had failed.

A.11 Chapter 11

1. Describe the different ways that an HMC or SE applies updated firmware.

Firmware is applied either disruptively or concurrently. A disruptive upgrade requires that the HMC or SE has planned downtime. Concurrent upgrade allows the system to run and perform tasks while being upgraded.

2. What is the reason for an alternate SE? How is it loaded? How is it kept in sync.

The alternate SE provides redundant hardware. If the primary SE fails, the alternate SE can take over until the primary SE is fixed. Also, the alternate SE allows for the use of concurrent driver upgrade. The primary SE is mirrored to the alternate SE one time per day, to make sure that if the alternate SE has to become the primary SE, it has the most up to date data.

3. What is cloning? What sub-tasks are performed by cloning?

Cloning is the process of upgrading one machine to the specific code level of another machine. A cloneable level for a machine must be created and then sent to the server, where another machine can retrieve that clone.

4. What is Concurrent Driver Upgrade? Explain the prerequisites for concurrent driver upgrade? What sub-tasks are performed by concurrent driver upgrade?

Concurrent Drive Upgrade is the process of upgrading a system (HMC or SE) without the need for planned downtime. In other words, the system can continue to run while fixes or new code is being applied. The system must be functioning properly, and the primary and alternate SE must be communicating with each other correctly.

A.12 Chapter 12

1. How many devices can you have on a parallel channel? How many devices can you have on a FICON channel?

The answers are 256 and 16384, but these are determined by the theoretical maximum addressing capacity. They might or might not be practical answers, depending on many other factors.

2. Where do you find the physical location (cage-slot-jack) of a PCHID?

This can be displayed by navigating to the PCHID Details panel, and then double-clicking a PCHID icon in the Channel Work area.

3. Where would you display the internal counters of an OSA channel?

These are displayed through the Advanced Facilities function in the Channel Operations sidebar.

4. When you configure off a CHPID, what state does it go to?

It goes to Standby state.

5. What is the name of the connector cable for the parallel channels?

Bus and Tag cables.

6. What are the most common channel types on a System z today?

ESCON and FICON channels are most common. Parallel channels are no longer available, although there are converter boxes for connecting parallel channel control units to ESCON channels.

7. Why is a LAN adapter (OSA) considered a System z channel?

An OSA is a LAN adapter on one end, but it has all of the channel functionality on the other end. That is, it works with channel commands (or their QDIO equivalent), has direct access to system memory, and so forth.

A.13 Chapter 13

1. Who might use SAD displays? Application Programmers, Systems Programmers, System Operators, IBM service people, executive management?

SAD displays are typically used by Operators and Systems Programmers. Usage is usually *ad hoc*, when investigating a suspected problem, rather than something that is done on a regular, scheduled basis. Other tools, such as z/OS RMF, are used for routine, long-term monitoring.

2. Why are SAD profiles important?

A great deal of information is available through SAD. It is necessary to define subsets that can be displayed in a reasonable way.

3. How busy would you expect a normal processor (or group of processors) to be in a typical System z production installation? How would this compare with typical Personal Computer servers? How busy would you expect a given channel (CHPID) to be?

Mainframes are typically run flat out with near 100% utilization of the processors. Various workload management routines ensure that the more important applications are given sufficient processor time when needed. In practice, PC servers are often run with perhaps less than 25% utilization in most cases (and 25% can be a high estimate), which is because these servers are typically dedicated to a single application, whereas the mainframe is usually running dozens of applications at the same time and can more effectively use its hardware.

Mainframe channels are a different story, because they can be a bottleneck if they are too busy. Traditionally, numbers larger than, say, 25% utilization (for a given channel) called for attention, which has changed somewhat for FICON channels because they can interleave many I/O operations at the same time.

4. SAD functions can collect a very large amount of data. How much does this slow down the System z processors?

System z machines have internal hardware that is dedicated to this type of data collection. SAD types of measurements have almost no effect on processor utilization.

5. How accurate is the SAD data? What is the time resolution of the data?

Much of the SAD data is collected on 15 second samples. In a steady-state environment, it should be quite accurate. However, the definition of steady-state can be complex for a large mainframe environment.

A.14 Chapter 14

1. In the earlier section on HMC and SE system management disciplines, capacity on demand is described as a function of multiple disciplines: configuration management, performance management, and serviceability. Why is this appropriate?

In a larger sense, system management disciplines include activities or events beyond the straightforward data processing functions, for example, a corporation might acquire another corporation and require additional computing capacity until the applications of the two corporations are meshed. Financial year-end processing is another type of temporary workload that is common. Mainframe customers can use Capacity on Demand for System z to respond to any change in demand for computing capacity, regardless of the system management discipline that leads to it.

2. Should a Systems Programmer be responsible both for ordering CoD records and for activating CoD upgrades, or should someone else be responsible for ordering CoD records? What are the advantages and disadvantages of separating these responsibilities? What are the advantages and disadvantages of not separating them?

Whether a Systems Programmer or someone else is responsible for ordering CoD records typically depends on the extent to which the Systems Programmer is accountable for managing IT expenses.

In smaller enterprises where Systems Programmers manage IT expenses, they can be responsible both for ordering CoD records and for activating CoD upgrades. A Systems Programmer has the advantage of ordering CoD records, of any type and at any time, to meet the immediate needs of the enterprise. But the disadvantage is that the enterprise might have inadequate oversight of the potential expenses that the Systems Programmer can incur by using the CoD records.

In larger enterprises, Systems Programmers typically are responsible only for activating CoD upgrades, while someone else managing IT expenses is responsible for ordering CoD records. The enterprise has the advantage of greater oversight of potential expenses, but might have a disadvantage in responding promptly when a Systems Programmer determines there is a need for additional capacity.

3. What are some ways a mainframe customer can activate On/Off CoD upgrades on multiple System z10 machines at the same time?

Use the Customize Scheduled Operations task from each machine's support element to schedule activating its upgrade at the same time as the other machines.

Use SNMP APIs for System z to develop a hardware systems management application that uses the APIs for Capacity on Demand to coordinate activating the upgrades.

For System z10 machines running z/OS R9 or later, use z/OS Capacity Provisioning to automate the activation of upgrades.

4. Research *utility computing* and *cloud computing* on the Internet. Compare and contrast on-demand computing with utility computing.

This is a discussion topic.

A.15 Chapter 15

1. The HMC panels for a repair action seem to progress in baby steps. Suggest some reasons for this.

Remember that these systems are installed world-wide and serviced by many people whose native language is not English. The small steps tend to avoid confusion that might occur with fewer, more complex panels. Also, many repair actions are extremely rare. The SR might have had intensive training in the related area several years earlier, but might not have not seen a real event in the meantime. Good prompting, through the HMC panels, serves as reminder material.

2. How much time and effort do you think is involved in developing the material behind these panels?

Lots and lots of time, especially for the detailed machine drawings that are needed for many actions. Thousands of these panels (and drawings) must be maintained and updated. Customer machines are not always at the latest hardware levels, and this requires the system to also maintain older information.

3. Would many of the detailed drawings be better placed in books (in hard copy)?

Maybe. Some people prefer hard copy. However, there are two strong points against it. For one thing, books might be out-of-date without the user being aware of this. Also, printing and distributing books (and updates) adds more expense to an already expensive undertaking.

Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 349. Note that some of the documents that we reference here might be available in softcopy only.

- ▶ *The IBM Mainframe Today: System z Strengths and Values*, IBM Redpaper REDP-4521
- ▶ *Introduction to the New Mainframe: Large-Scale Commercial Computing*, SG24-7175
- ▶ *IBM System z Connectivity Handbook*, SG24-5444
- ▶ *Introduction to the New Mainframe: z/VM Basics*, SG24-6366-01
- ▶ *IBM System z Strengths and Values*, SG24-7333-01
- ▶ *Introduction to the New Mainframe: z/VM Basics*, SG24-7316
- ▶ *IBM System z10 Enterprise Class Technical Guide*, SG24-7516-01
- ▶ *System z Virtualization Software Architect Automatic Partition Resource Manager for System i and iSeries*, SG247174

Other publications

These publications are also relevant as further information sources:

- ▶ *System z Input/Output Configuration Program User's Guide for ICP IOCP*, SB10-7037
- ▶ *System z Hardware Management Console Operations Guide*, SC28-6873
- ▶ *System z9 Support Element Operations Guide*, SC28-6979

Other materials

- ▶ *Mainframe Hardware Course - Mainframe's Physical Environment*
http://publib.boulder.ibm.com/infocenter/zos/basics/topic/com.ibm.zos.s.zcourses/zcourses_MFWenvironment.pdf
- ▶ *Mainframe Concepts*
http://publib.boulder.ibm.com/infocenter/zos/basics/topic/com.ibm.zos.s.zmainframe/zmainframe_book.pdf
- ▶ RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1
<http://www.rsa.com/rsalabs/node.asp?id=2152>
- ▶ *Introduction to SSL*
<http://docs.sun.com/source/816-6156-10/contents.htm>
- ▶ *Capacity on Demand advancements on the IBM System z10*
<http://www.research.ibm.com/journal/abstracts/rd/531/axnix.html>
- ▶ *Reliability, availability, and serviceability (RAS) of the IBM eServer z990*, IBM Journal of Research Volume 48
- ▶ *Concurrent driver upgrade: Method to eliminate scheduled system outages for new function releases*, IBM Journal of Research and Development IBM System z9 Vol 51, No. 1/2
- ▶ *z990 NetMessage-protocol-based processor to Support Element communication interface*, IBM Journal of Research and Development IBM System z9 Vol 48, No. 3/4
- ▶ *IBM zSeries: Premier Business Resiliency for the On Demand World*, GM13-0638-00
- ▶ *Advanced firmware verification using a code simulator for the IBM System z9*, IBM Journal of Research and Development IBM System z9 Vol 51, No. 1/2
- ▶ *System z Virtualization*, Romney White, IBM
- ▶ *System z Logical Partitioning design and development*, J.P. Kubala, IBM
- ▶ *Logical Partition Mode Physical Resource Management on the IBM eServer z990*, I. G. Siegel; B. A. Glendening; J. P. Kubala IBM

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived



Introduction to the System z Hardware Management Console

(0.5" spine)
0.475" <-> 0.873"
250 <-> 459 pages



Introduction to the System z Hardware Management Console



Large scale hardware systems management concepts

Design and implementation of a management controller

Practical operational techniques and scenarios

In this textbook, we provide students with the background knowledge and skills that are necessary to begin using the basic functions and features of the System z Hardware Management Console (HMC) and System z Support Elements (SE). This book is part of a series of textbooks that are designed to introduce students to mainframe concepts and to help prepare them for a career in large systems computing.

For optimal learning, we assume that the students are literate in personal computing and have some computer science or information systems background. Others who can benefit from this textbook include z/VM and z/OS professionals who want to expand their knowledge of other aspects of the mainframe computing environment.

After reading this textbook and working through the exercises, the student will have a basic understanding of:

- ▶ System z Hardware concept and the history of the mainframe.
- ▶ HMC and SE components.
- ▶ The HMC and SE user interface and user management.
- ▶ System z First Failure Data Capture and Serviceability functions.
- ▶ Monitoring the System z performance using the SAD.
- ▶ Implementing the networking capabilities of the HMC.
- ▶ Managing System z Partitions (LPAR)
- ▶ Partition resource movement and sharing.
- ▶ System z Customer Upgrade on Demand advanced functionality.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks